Data Science Techniques Used in Process Mining for Removing Noise

Kalle Heinonen

Master's Thesis
Master of Engineering - Big Data Analytics
2023

MASTER'S THESIS	
Arcada	
Degree Programme:	Master of Engineering - Big Data Analytics
Identification number:	9206
Author:	Kalle Heinonen
Title:	Data Science Techniques Used in Process Mining for Removing Noise
Supervisor (Arcada):	Leonardo Espinosa-Leal
Commissioned by:	

Abstract:

Process mining supports organisations to understand and improve existing processes by extracting event log data from IT systems and visualising it. This study explores the main challenges associated with event log data processing. One of the most significant challenges is the presence of noise activities, which are infrequent and do not accurately represent the typical behaviour of the process. A process model, known as a Petri net, is generated using process mining techniques to enable business stakeholders to analyse and verify processes. In this thesis, the methods that are effective at handling noise in event log data are investigated. An experiment was conducted using two real-life event logs to evaluate which algorithm is best suited to handle noisy data. Two Petri nets were generated using Integer Linear Programming (ILP) and Inductive Miner algorithms. Generalisation and complexity were compared in addition to quality metrics such as F-score. The results indicate that the Inductive Miner algorithm generates a Petri net with a good F-score and suitable complexity metrics. When dealing with a more complex event log with a higher number of events and more noise, the ILP Miner produced a slightly better F-score by replicating more transitions in the model. The algorithms perform well for use cases where precise results are not essential. However, in industries such as medicine or fraud detection, where accuracy is critical, it is recommended that the excluded traces are checked by an expert to ensure that vital information is not omitted in a discovered model.

Keywords:	discovery algorithms, inductive miner, integer linear programming miner, noise reduction, petri net, process mining, process modelling
Number of pages:	33
Language:	English
Date of acceptance:	18.05.2023

CONTENTS

Figures

Tables

Abbreviations

IT	Information technology
BPMN	Business Process Modelling Notation
ID	Identification
WF	Workflow
CFC	Control-flow complexity
ACD	Average connector degree
CNC	Coefficient of Network connectivity
APD	Automated Process Discovery
ILP	Integer Linear Programming Miner
IEEE	Institute of Electrical and Electronic Engineers
BPI	Business Process Intelligence

1 INTRODUCTION

1.1 Background

With the current market challenges of component shortages, rising labour and material costs, and logistics disruptions, companies are seeking ways to enhance the efficiency of their processes (Espinosa-Leal et al., 2020). One of the key obstacles in achieving this is gaining a clear understanding of the current state of business processes. This is where process mining comes in, helping organisations to create a digital representation of their processes. With this information, data-driven root-cause analyses can be performed to identify inefficiencies.

Process mining is a set of techniques that aim to discover, monitor, and improve real-world processes by extracting event log data from IT systems. By understanding current processes, organisations can identify areas for improvement, detect variations, and investigate root causes. In the past, processes were analysed by manual means, such as interviews with employees. Such an approach has been missing a link between company processes and IT systems. This gap has been filled by process mining which links data to operational processes and uses a set of data mining techniques to visualise and analyse company processes. (van der Aalst, 2016)

Process mining techniques are built on classical data mining techniques which link processes to the data science field. Data mining is defined as “the analyses of (often large) datasets to find unsuspected relationships and to summarise the data in novel ways that are both understandable and useful to the data owner” (Hand et al., 2001). In this study, process mining techniques such as Petri net and α-algorithm are covered (van der Aalst, 2016). These techniques are used to get a better understanding of the process performance.

Process mining and data mining techniques have been studied for many years by various researchers. One of the most experienced researchers in this area is Wil van der Aalst, a

Dutch computer scientist. He has been contributing to the area of process mining with his publications (van der Aalst, 2016), (van der Aalst et al., 2004), (van der Aalst et al., 2011), (Leemans et al., 2013), (Günther & van der Aalst, 2007), (Adriansyah et al., 2012) and (Rozinat & van der Aalst, 2008). Currently, he is acting as Chief Scientist at Celonis company bringing practical applications of process mining into businesses. This study follows the ideas presented in the studies of process mining "Process Mining: Data Science in Action" (van der Aalst, 2016) and "Process Mining in Action: Principles, Use Cases and Outlook" (Reinkemeyer, 2020). The research will be complemented by data mining literature to cover data mining techniques in detail.

1.2 Research aim

Incorporating process mining in business poses several data-related obstacles. Overcoming these challenges is crucial for leveraging process mining generated models to enhance processes. (van der Aalst et al. 2009) The research aims to examine the challenges associated with data quality in process mining and explore potential solutions.

The study will focus on two main research questions:

- What are the primary challenges related to data quality in process mining?
- What kind of techniques exists in process mining for removing noise?

First, the study will cover the literature on process mining and the main challenges related to quality in process mining will be identified. Then one of the challenges will be selected for in-depth research. In the final part of this thesis, a comparison of two existing noise filtering algorithms will be made using ProM software. A conclusion will be drawn on which is the most suitable algorithm to tackle the noise.

2 LITERATURE REVIEW

2.1 Process mining and data science overview

Process mining is a discipline that acts as the bridge between data science and process science. The goal of process mining is to discover, monitor and improve real processes by extracting data from event logs available from IT systems (van der Aalst, 2016). These insights are used to identify process improvements, check compliance, compare process variants, and discover inefficiencies such as rework and bottlenecks.

Data science is "a set of fundamental principles that support and guide the principled extraction of information and knowledge from data" (Provost & Fawcett, 2013). One of the concepts that is closely related to data science is data mining. Data mining can be defined as "the analysis of (often large) datasets to find unsuspected relationships and to summarise the data in novel ways that are both understandable and useful to the data owner" (Hand et al., 2001).

One of the areas of data science is machine learning. Machine learning focuses on the question of how to build programs that automatically improve with experience. Machine learning refers to "algorithms that give computers the capability to learn without being explicitly programmed (learning from experience)" (Samuel, 1959). Process mining adds process perspective to data science, as data mining and machine learning techniques do not consider end-to-end process view.

Process science refers to the broader discipline that combines knowledge from information technology and management sciences to improve and run operational processes. One of the key concepts in process science is business process management, the discipline combining design execution, control, and measurement of business processes (van der Aalst, 2016). Process mining complements process science with a data-driven approach. Process science focuses on process models, whereas process mining focuses on the insights that can be achieved from event data. Process science is emphasising process

models such as Petri nets and Business Process Model and Notation (BPMN). Petri nets and BPMN are both graphical notations used to model business processes, workflows, and other types of systems (van der Aalst, 2016). BPMN is a standard notation used to represent business processes and workflows in a clear and standardised way. It uses a set of symbols and shapes to represent different types of activities, events, and flow elements, as well as swimlanes to group activities by roles or departments (Acosta-Velásquez et al., 2022).

Alternatively, Petri nets are a mathematical modelling language used to describe and analyse systems with concurrency, synchronisation, and other complex behaviours. Petri nets use a set of nodes and arcs. (van der Aalst, 2016)

2.2 Process modelling

To explain how process mining supports business process management, the phases involved in the business process management cycle are covered in the research. Process mining supports a business process management life cycle consisting of the following phases. (Wil et al., 2003)

- **Vision**: The vision is created based on the changes an organisation would like to implement.
- **Diagnosis**: An analysis of the selected processes is carried out to obtain an understanding of the current performance. The performance is evaluated based on key performance indicators such as throughput time and customer satisfaction.
- **Process redesign**: KPIs are set for the selected scope of the redesign targets. The requirements and the necessary supporting technology and information systems are defined.
- **System design and construction**: Because of redesign, an outline of the technology components and their interaction process is created. The architecture of the process is developed at this phase.

- **Transfer and implementation**: The newly redesigned process is handed over to the organisation. The changes to the process (including roles, processes, and system functionality) are communicated to the organisation.
- **Evaluation**: The continuous phase of monitoring starts for the developed process.

Process mining uses event logs to conduct three types of analyses (van der Aalst, 2016). The first type is called **process discovery**. Process mining takes an event log and produces a model. The result of the process discovery is a process flow illustrated in Figure 1. The process flow points out the inefficiencies in the process. It is also used as the basis for all process mining analyses including conformance checking.

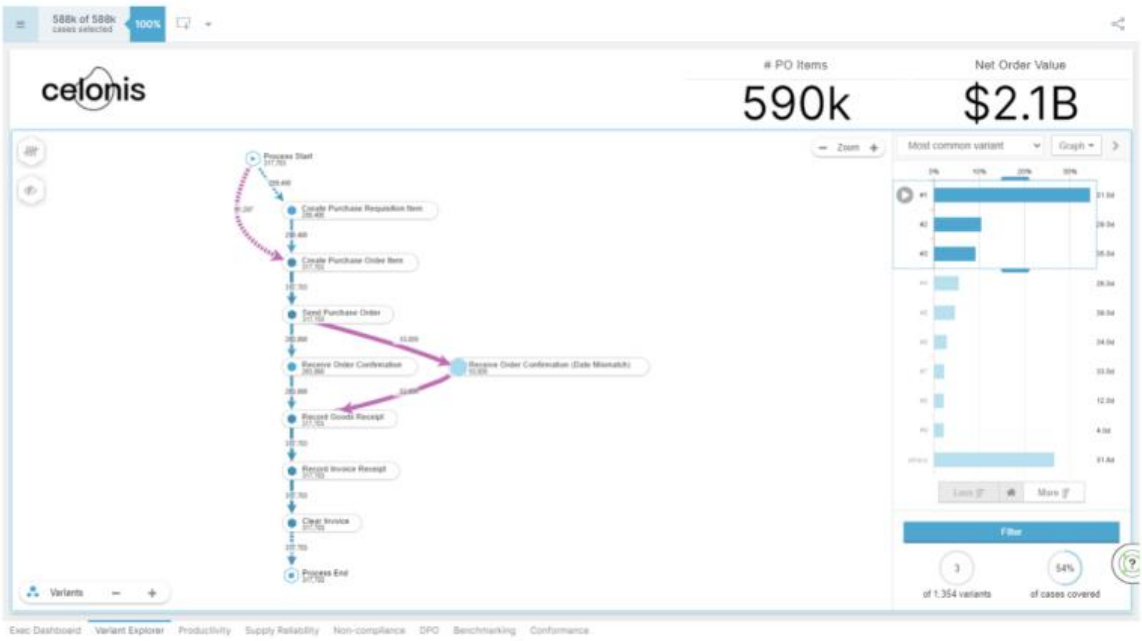

Figure 1. Process flow (Celonis, 2023)

The second type of process mining analyses is **conformance checking** where a process model produced in the process discovery phase is compared with the model from an event log. Conformance checking allows the detection of deviations and the measurement of their impact on the process performance. Figure 2 illustrates the results of conformance checking.

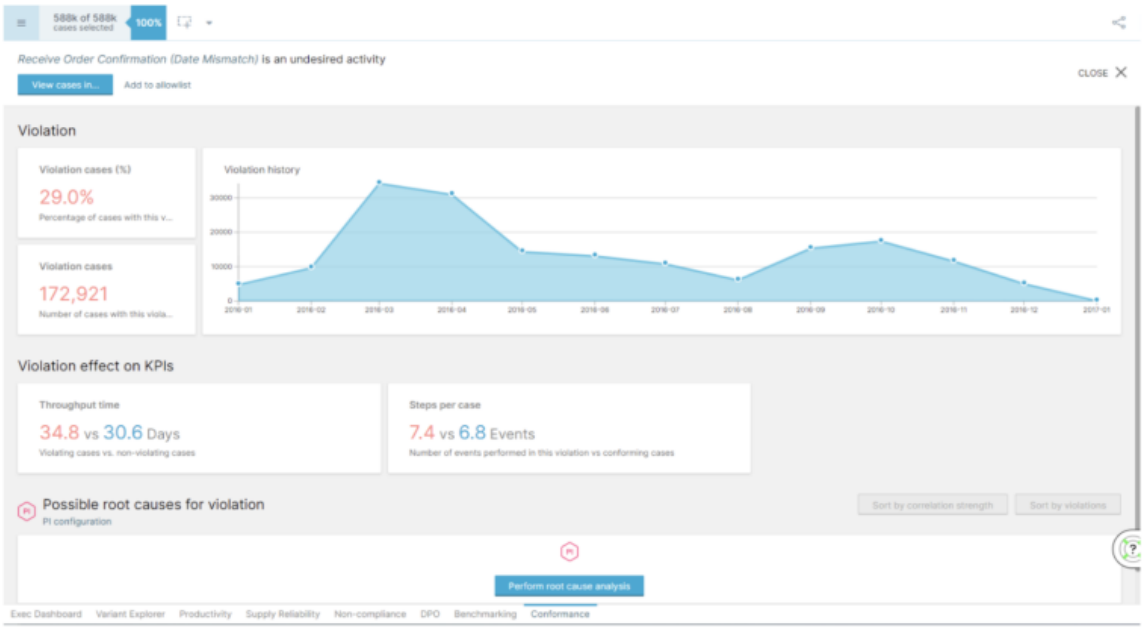

Figure 2. Conformance checking (Celonis, 2023)

The third type of process mining analyses is called **process enhancement.** The main purpose of this step is to improve the existing process model using the information obtained from process mining discovery and conformance checking. For example, in Figure 3 the automation rate of the process is present, and the most frequent manual activities are displayed. One of possible the actions is to review the reasons for manual changes and automate the manual activities.

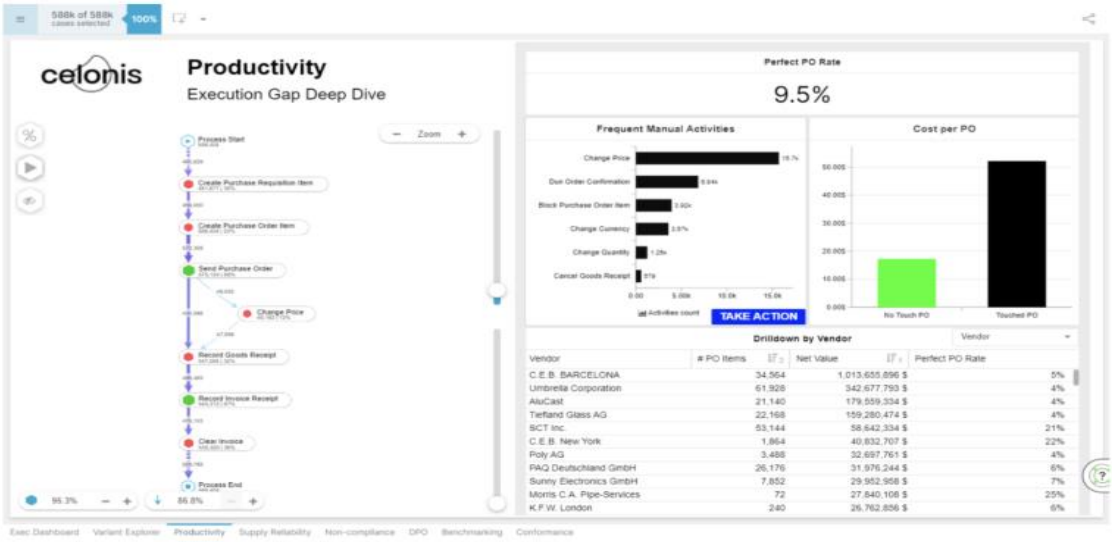

Figure 3. Process enhancement (Celonis, 2023)

2.3 Petri net

Process mining uses event log data to automatically discover a process model. A process model is expressed in terms of a Petri net. Petri nets are the oldest and best-investigated process modelling language allowing the modelling of concurrency (van der Aalst, 2016). A Petri net consists of places, transitions, and a flow relation between them. Places act as used reached milestones and transitions as individual tasks. Places are shown as circles and transitions are shown as rectangles. The arc describes the relationship between different activities. The state of a Petri net is determined by the distribution of tokens within the network and referred to as marking. The network structure is governed by a firing rule. A transition is enabled to fire if its input places are marked, and thus tokens can flow through the network (van Hee & Reijers, 2000). The process constructions are called blocks which are presented in Figure 4.

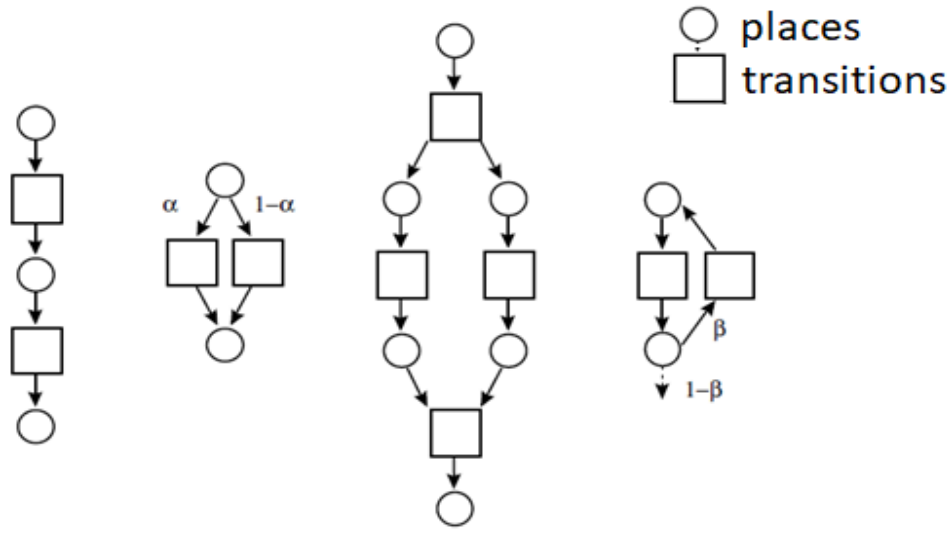

Figure 4. Types of blocks (van Hee & Reijers, 2000)

The first block represents the process where two tasks are in sequential order. The second task can be only completed when the first task is carried out. The second block represents a process where either of the activities is carried out. The third block illustrates that the two tasks can be executed simultaneously. In the last block, there is a repetition of a task.

Figure 5 is an example of a process model that is discovered by the α-algorithm with several traces *{(a,b,d,e,h), (a,d,c,e,g), (a,c,d,e,f,b,d,e,g), (a,d,b,e,h), (a,c,d,e,f,d,c,e,f,c,d,e,h), (a,c,d,e,g)}*.

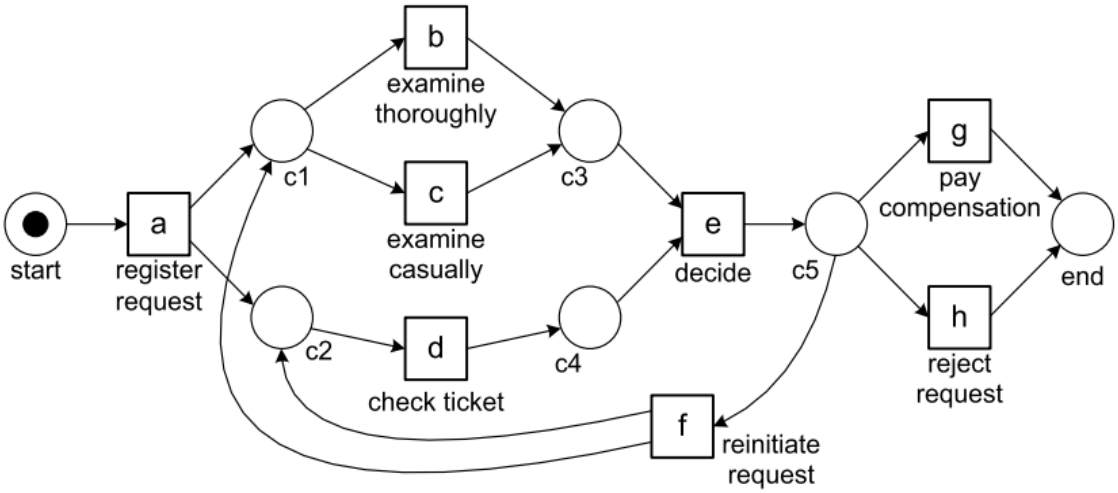

Figure 5. Process model discovered by α-algorithm based on set of traces (van der Aalst et al., 2004)

In process mining an event log data is used to construct a Petri net. The event log data contains the event name, the timestamp when the event has happened, and the case ID to which the event belongs. An example of event log data is presented in Table 1, where Order 365 is a case ID, Order creation is an event name and 30-12-2021:11.02 is a time stamp.

Order ID	Activity	Timestamp	Country
365	Order creation	30-12-2021:11.02	Finland
365	Order shipped	15-03-2022:12.10	Finland
534	Order packed	15-12-2021:14.15	Germany
895	Order creation	10-02-2022:09.08	Germany

Table 1. Example of event log data

The event log data can be enriched with additional information. Additional event log data may contain information such as resources, location, and monetary information. Depending on process mining techniques only part of the information will be used in analyses.

Process mining aims to generalise the behaviour contained in the event logs to show the most likely underlying model. One of the challenges in process mining is to find a balance between overfitting and underfitting models. An overfitting model is too specific showing rare behaviours, and an underfitting model is illustrating too generic behaviours (van der Aalst, 2016). Process mining can facilitate the construction of better models in less time. Process discovery algorithms like the α-algorithm can automatically generate a process model. The α-algorithm produces a Petri net. The α-algorithm is one of the possible process discovery algorithms. To find a balance between overfitting and underfitting models, more advanced algorithms are needed. The models also help to deal with quality-related challenges such as incompleteness (e.g. logs containing only a small portion of possible behaviours because of several alternative traces) and noise (containing rare behaviour that should not be automatically taken in the model).

Workflow (WF) nets are a natural subclass of Petri nets used for modelling and analysing operational processes. Workflow nets model the creation and the completion of the cases flowing through the event log. Creation of the model is started by putting a token in a unique start place and completed by reaching a unique sink plan or end of the process. The Petri net presented in Figure 5 is a Workflow net. (van der Aalst, 2016)

2.4 α-algorithm

Process mining can facilitate the construction of better models in less time. Process discovery algorithms like the α-algorithm can automatically generate a process model. The α-algorithm produces a Petri net. The algorithm represents the generic idea used by many process mining algorithms. α-algorithm starts with an empty Petri net and iteratively adds transitions and places to the net based on a set of structural constraints. The algorithm uses ordering relations between events in the event log to derive a process model. Next,

it constructs a footprint matrix and converts the footprint matrix into a Petri net. The resulting Petri net is generally compact and may contain complex behaviour.

The α-algorithm has problems with noise, infrequent or incomplete behaviour, and complex routing constructs (van der Aalst, 2016). The event log is assumed to be noise free and complete when using an α-algorithm. The algorithm does not consider the frequency or relation making it sensitive to noise. The idea of this algorithm has been used in developing more complex and robust techniques. Later, Alves de Medeiros et al. developed the α+ algorithm, which can detect short loops (de Medeiros et al., 2004). Wen et al. proposed additional extensions to the α-algorithm: the β-algorithm to detect concurrency and the α++ algorithm uses non-local information in an event log to discover non-free choice constructs (Wen et al., 2007).

Input for the α-algorithm is a simple event log L over A, i.e. $L \in B(A*)$. The event log is referred to as L. *A* stands for activities in the event log. The capital letters, e.g. A and B are sets of activities. Inside these are individual activities where no capitalisation is used, e.g. $a, b, c \in A$. The output of the α-algorithm is a marked Petri net.

The α-algorithm identifies patterns within the event log, as illustrated in Figure 6. When an activity *a* is followed by *b* but *b* is never followed by *a*, the algorithm infers a causal relationship between *a* and *b*. As a result, the Petri net will contain a place that connects *a* to *b*. Four relations capture patterns from an event log. (de Medeiros et al., 2004)

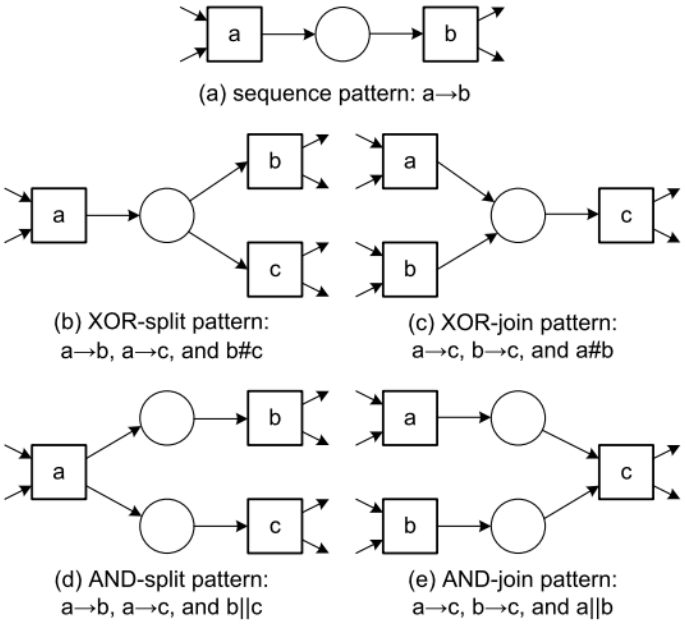

Figure 6. Typical process patterns and footprints in event log (van der Aalst, 2016)

Let L be an event log over A, i.e. L ∈ B ($A*$). Let $a, b \in A$.

- **Directly follows relation:** $a >_L b$, a is directly followed by b if and only if there is a trace $\sigma = (t_1,t_2,t_3,...,t_n)$ and $i \in \{1,...,n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$.
- **Sequence relation:** $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not> _L a$.
- **Non-direct relation**: $a \#_L b$ if and only if $a \not> _L b$ and $b \not> _L a$.
- **Parallel relation:** $a \parallel_L b$ if and only if $a >_L b$ and $b >_L a$.

Let's consider a log L1 = $[(a,b,c,d)^3, (a,c,b,d)^2, (a,e,d)]$. For this event log, the following log-based ordering relations can be found:

- $>_{L1} = (a,b),(a,c),(a,e),(b,c),(c,b),(b,d),(c,d),(e,d)$
- $\rightarrow_{L1} = (a,b),(a,c),(a,e),(b,d),(c,d),(e,d)$
- $\#_{L1} = (a,a),(a,d),(b,b),(b,e),(c,c),(c,e),(d,a),(d,d),(e,b),(e,c),(e,e)$
- $\parallel_{L1} = (b,c),(c,b)$

The footprint of the log is present in Figure 7.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Figure 7. Footprint of L1: $a \#_L a$, $a \rightarrow_{L1}$ b, $a \rightarrow_{L1}$ c, etc. (van der Aalst, 2016)

These relations are used to discover patterns in process models. For instance, if *a* and *b* occur in sequence, the event log will indicate an $a \rightarrow_L b$ relationship. If after *a* there is a choice between *b* and *c*, the log will show $a \rightarrow_L b$, $a \rightarrow_L c$, and $b \#_L c$ because *a* can be followed by *b* and *c*, but *b* will not be followed by *c* and other way around. If $a \rightarrow_L c$, $b \rightarrow_L c$, and $a \#_L b$, then this implies that after the occurrence of either *a* or *b*, *c* should happen. If $a \rightarrow_L b$, $a \rightarrow_L c$, and $b \|_L c$, then it appears that after *a*, both *b* and *c* can be executed in parallel (AND-split pattern). If $a \rightarrow_L c$, b $\rightarrow_L c$, and $a \|_L b$, then the log suggests that *c* needs to synchronise *a* and *b* (AND-join pattern). The α-algorithm generates the Workflow net based on information provided in L.

2.5 Main challenges

Despite the existing process mining techniques and tools, there are still challenges that need to be solved. For example, process discovery is one of the biggest challenges in process mining. It is difficult to construct a process model based on event logs that are incomplete and noisy.

In process mining, **representational bias** should be considered. There are graph-based notations that allow models to contain deadlocks, that are defined as disconnected parts. α-algorithm may discover Workflow nets that have a deadlock. Approaches using process trees do not have this problem, but they have difficulties in expressing certain process constructs and fail to duplicate activities when needed. (van der Aalst et al., 2011)

Another challenge is called **concept drift,** meaning that processes change while being observed. Existing process discovery approaches do not take these changes into account. It could be beneficial to detect process changes and visualise these changes. (van der Aalst, 2016)

There are always trade-offs between the quality criteria of process models (van der Aalst, 2016). Finding a solution where the following four criteria balance each other is challenging and requires continuous experimentation. When selecting a suitable algorithm for noise removal the following criteria should be considered.

- **Fitness:** The model should represent the behaviour observed in the event log.
- **Precision:** The discovered model should not contain any behaviour completely unrelated to what was present in the event log.
- **Generalisation:** The discovered model should provide a generic overview of the behaviour observed in the model.
- **Simplicity:** The model should be as simple as possible.

2.6 Quality of event log

In the discovery of a process model, it is assumed that the model contains a representative sample of behaviour. However, there are challenges such as noise and incompleteness that can prevent this from happening. Noise and incompleteness refer closely to the quality of an event log. Noise means that the event log contains rare infrequent behaviour which is not representative of the typical behaviour of the process. Incompleteness represents an issue where an event log contains too few events to discover the underlying control flow structures.

2.6.1 Noise

Real-world logs are often noisy because they may contain duplicated, inconsistent, and incorrectly logged sequences of events. Such problems can result from IT system issues, data entry problems, data transmission, and streaming problems, or other technology-

related limitations. Noisy exceptional or infrequent behaviour should not be included in the discovered model. It is often very hard to extract useful information from the behaviour which occurs rarely. However, noise does not mean the data in the event log is incorrect. Instead, there are always some infrequent occurrences or outliers. Such inputs usually lead to the creation of exceptional paths in the process model.

Identifying issues as soon as possible is important when extracting event log data from various sources. Sometimes it can be difficult to distinguish between noise and desired infrequent behaviour. It is often hard for a discovery algorithm to distinguish incorrect logging from exceptional events. This can be tackled by human judgment while manual processing of the event log data. There are also discovery algorithms that help in reducing noise in the data such as Fuzzy Mining and Heuristic Mining (van der Aalst, 2016).

The metrics of support and confidence are useful when using association rules in the form of $X \Rightarrow Y$. The support of a rule $X \Rightarrow Y$ indicates the applicability of the rule, i.e. the instances where both preceding and subsequent instances are true. The confidence metric measures the reliability of the rule. For example, a rule about flakes $\wedge$ coffee with milk having a support of 0.2 and confidence of 0.9 means that 20% of customers buy flakes, coffee, and milk at the same time, whereas 90% of the customers that buy flakes and coffee also buy milk. Rules with low confidence can be considered noise.

This is how the rules can be applied to α-algorithm. The α-algorithm begins with the establishment of the relation >L. This relation is defined as follows: $a >_L b$ holds true if and only if there exists a trace in the event log L where activity *a* is directly followed by activity *b*. The support of $a >_L b$ can be determined by the number of times the pattern *(...,a,b,...)* appears in the log. $a >_L b$ has a reasonable support if the pattern *(...,a,b,...)* appears 1000 times. Moreover, if *a* appears 1500 times and *b* 1000 times, then $a >_L b$ has good confidence. However, if the pattern appears frequently but *a* and *b* appear more frequently on their own, the confidence is low. Therefore, removing noisy $a >_L b$ rules can result in a more accurate log, which is better input for the α-algorithm. Considering

noise, an 80/20 model is favoured as it allows the process model to describe 80% of the behaviour observed in the log.

2.6.2 Incompleteness

Incompleteness refers to a situation where there is too little data to derive the process model (Wang, 2022). The process model allows for an exponential number of different traces. An event log might not contain every possible trace, because some are rarer than others. In a complex example with multiple possible paths, the log may contain only a fraction of these. This can be caused by the fact that certain events might be missed due to recording errors. The α-algorithm assumes a relatively weak notion of completeness to avoid this problem. Most process mining algorithms currently in use operate under the assumption that the log is complete, implying that the direct successor relationship, causal dependency, and overall log are all complete. The α-algorithm uses a local completeness notion based on >L, i.e. if there are two activities *a* and *b* and *a* can be directly followed by *b*, then this should be observed in the log at least once (van der Aalst, 2016). Often, weaker completeness notions are needed.

To highlight the importance of completeness, let's consider a process comprising 10 activities that can be executed simultaneously, along with a corresponding log containing information about 10,000 cases. The total number of possible interleavings in the model, considering the 10 concurrent activities, is 10! = 3,628,800. Therefore, it is not possible for each interleaving to be present in the log since there are fewer cases than potential traces. In a process where 10 activities can be executed in parallel, local completeness can reduce the required number of observations dramatically. For example, for the α-algorithm only $10 \times (10 - 1) = 90$, rather than 3,628,800 different observations are needed to construct the model.

Some algorithms assume that the event log contains all possible traces. A very strong completeness assumption easily results in an overfitting model. Weak completeness assumption results in underfitting models.

2.7 Quality of discovered model

The concept of process model quality was first introduced by Rozinat and van der Aalst in terms of fitness and precision (Rozinat & van der Aalst, 2008). The quality of the discovered model is measured based on four dimensions: recall, precision, generalisation, and simplicity.

Fitness or **recall** measures how well the model represents the behaviour observed in the event log. A recall measurement of 0 indicates the inability to reproduce the behaviour from the event log, whereas a value of 1 indicates the opposite.

Precision measures the capability of a model to reproduce only the behaviour seen in the event log. The identified model must not include any behaviour that is entirely disconnected from what was observed in the event log. The value 0 indicates that the model can reproduce the behaviour never observed in the log, whereas a value of 1 indicates that the model only reproduces the behaviour observed in the log. The measurement is calculated based on the ratio between the number of executed actions and the number of possible actions. The key to understanding the difference between fitness and precision is that fitness measures only how well the model represents the behaviour in the event log whereas precision measures the capability to reproduce only the behaviour observed in the event log. (Adriansyah et al., 2012).

F-score combines measures of fitness and precision as a measure of accuracy.

$$F\ score = 2 * \frac{\text{fitness} * \text{precision}}{\text{recall} + \text{precision}} \quad (1)$$

Generalisation means that the discovered model should provide a generic overview of the behaviour observed in the model. It can be thought of as the opposite of precision, providing a measurement of the model's capability to reproduce behaviour never observed in the log. The measurement is calculated using 10-fold cross-validation, a method commonly used in data mining. (Kohavi, 1995)

Simplicity indicates that the model should be as simple as possible. There are different ways how simplicity can be measured. (Wen et al., 2009)

- Size: # of places, # transitions, and # of arcs.
- Density: Proportion between the actual number of arcs and the maximum potential number of arcs in a model that has the same quantity of nodes.

A good balance between overfitting and underfitting is important for process discovery. An optimal model should generalise and at the same time to not restrict behaviour observed in the event log. A model that does not generalise easily becomes overfitting allowing only a specific pattern in the log. An underfitting model has the opposite problem. It is generalising the data too much in the log and has poor precision. For example, the Petri net shown in Figure 5 generalises as it allows more patterns than seen in the training set. The behaviour (a,h) where the request is registered and immediately rejected is not present. This does not necessarily imply that it is not possible. However, having a model where (a,h) is present although it is not in the event log would result in underfitting. This dilemma is caused by the lack of negative examples in the event log.

Noise impacts the quality dimensions differently. The recall might not be reliable as having a too high value of it does not guarantee that the discovered model is an exact representation of reality. Precision tends to be lower because noise introduces new connections in the model that should not be there. Generalisation is higher due to the increased number of connections. Simplicity tends to be lower because the model becomes complex containing more activities and connections.

2.8 Discovery algorithms for coping with noise

Some algorithms cope with noise better than α-algorithm. Algorithms such as Heuristics Miner, Fodina, and Inductive Miner are often used with noisy event logs. (van der Aalst, 2016) These algorithms are called Automated Process Discovery techniques (APD). APD techniques filter out noisy behaviour during the process discovery phase. Such techniques can produce a process model that contains the most frequent traces, thus reducing the

complexity of the model. Most APD techniques cannot identify duplicate activities which appear in different parts of the process model.

Heuristics Miner algorithm discovers a model using the α-relationship. This algorithm uses the frequency of binary relations among tasks and detects direct dependency, concurrency, and non-directly connectedness. By considering all task pair dependencies and their frequency, this technique identifies short loops and non-free choice structures from noisy logs. By calculating the dependency measure between activities, this approach filters out noise by eliminating causal dependencies that fall below a specified threshold. Sometimes this technique generates a model with incorrect behaviour as soundness and fitness are not guaranteed. The soundness is not guaranteed when the final place is marked but one token is left behind.

Despite having noise-tolerant capabilities, noise has a significant impact on the quality of the models produced by these algorithms. For instance, Heuristics Miner which employs a technique for disambiguating event dependencies due to noise can have a 45% drop in accuracy when the level of noise is 3% of the total log size. To limit the impact of noise the Heuristics Miner has a frequency-based metric. Given two labels a and b, $a \Rightarrow b = (|a>b| - |b>a|) / (|a>b| + |b>a| + 1)$. This metric is used to verify if || relationship has been correctly identified and if the value of ⇒ is above the given threshold. || relationship will be replaced by → relationship. A similar approach is used in the Fodina algorithm. (Weijters & Ribeiro, 2011)

Alves de Medeiros et al. proposed a Genetic algorithm that can detect non-local relationships that are not explicit in the event log because of a global search ability based on the use of fitness function using a recall and precision measure to find the best-matched models. Genetic Miner can detect a non-local pattern. (de Medeiros et al., 2007)

Inductive Miner is a discovery algorithm based on a divide-and-conquer approach. This algorithm generates a directly follows graph. It identifies a cut (i.e. ×, →, ∧) in the graph along which the log is split. It then detects an operator. This operation is repeated until

more cuts are identified or no more cuts can be identified. From each sub-log, it builds a sub-model. The process model then is built based on these sub-models. While the model's soundness is assured, its precision is notably low, and the model tends to be overly generalised. To cope with the noise, the algorithm applies filters like Heuristics Miner, which removes edges from the directly follows graph. It uses an eventually follows graph to remove edges that the first filter did not remove. (Leemans et al., 2013) As limitations, the dependencies are removed only if they are ambiguous (e.g. replacing || dependency with → dependency). It does not remove dependencies that are simply infrequent. Dependencies are only removed from the dependency graph, leaving the log unaffected, influencing the result of the discovery.

Fuzzy Miner is another algorithm applying noise filtering a-posteriori directly on the model discovered. The algorithm discovers behaviour models from the event log based on correlation and significance, producing a fuzzy net where each node and edge is associated with a value of correlation and significance. After mining the user provides significance and correlation thresholds that are used for filtering. These two thresholds can simplify the model by preserving significant behaviour, aggregating less significant but correlated behaviour (by clustering of nodes and edges), and abstracting less significant and less correlated behaviour (by removal). One of the downsides is that a fuzzy net only provides an abstract representation of the process behaviour extracted from the log due to its intentionally underspecified semantics which leaves room for interpretation. This technique produces a model without executable semantics. Its soundness and fitness are not guaranteed. (Günther & van der Aalst, 2007)

Integer Linear Programming Miner (ILP) handles noise by integrating noise into the discovered model. This algorithm translates relations observed in the logs into an Integer Linear Programming problem, where the solution is a Petri net capable of reproducing all behaviour present in the log. The negative effect of this approach is to generate so-called "flower" models which suffer from very low precision. (van der Werf et al., 2009)

3. RESEARCH METHODOLOGY

3.1 Research datasets

Process data is considered a critical asset of a business as it contains valuable data on company operations, for example, delivery data of products including lead times of different processes, sales figures, and locations. There are not many companies that publish their data as datasets for research purposes. Therefore, publicly available datasets were used in this research. The data is available at http://www.processmining.org/event-data.html or Process Mining Event Logs - IEEE Task Force on Process Mining (tf-pm.org). The selected datasets are included in a collection of datasets originally utilised in a Business Process Intelligence Challenge organised annually. The data is published for the annual competition where participants are to utilise the data to uncover insights about a certain process. For this purpose, the anonymised datasets are made publicly available by different companies sponsoring the event.

For the research two individual event logs called BPI 2012 and BPI 2014 were used. The datasets represent different processes and industries. Table 2 indicates the characteristics of the event logs in terms of the number of case IDs, number of events, and per cent of noise.

Log	# of case IDs	# of events	% Noise
Dataset 1 BPI 2012	13087	148192	25%
Dataset 2 BPI 2014	42790	392320	57%

Table 2. Characteristics of selected event logs

The first dataset relates to a loan application process. The event log data is from a Dutch financial institute. The event log consists of event classes such as offer, final decision, and suspicion of fraud. The second dataset is from Rabobank Netherlands Group ICT

department and includes record details from an ITIL Service Management tool called HP Service Manager. The event log contains incident cases.

3.2 Proposed method

Many trial Petri nets were created on different models to sort out a suitable model for handling noisy and incomplete data. An emphasis was put on Automated Process Discovery techniques that address data quality issues and affect the accuracy and comprehensibility of the models. Eventually, Inductive Miner and ILP Miner were selected to process the datasets and to discover the Petri nets. The basic α-algorithm was not selected because of its poor characteristics in handling event logs with noise as described in chapter 2.4. ProM software was used to generate the Petri nets and evaluate the data quality. ProM is a framework that includes plugins for supporting various process mining techniques (ProM Tools, 2023). Being implemented in Java, ProM is platform-independent and free to download. The ProM program was chosen because of its accessibility and wide adoption among process mining professionals (Claes & Poels, 2013). Anyone can contribute to the development of the tool by developing new process mining plugins.

After uploading the event log data, a filter log function was used in the Simple Heuristics plugin of ProM to remove infrequent labels with a threshold value of 90%. The filter log function in Simple Heuristics plugin removes all traces that do not start or end with a particular event. It can also remove all events related to a specific process task by calculating the frequencies of event occurrence. Furthermore, the filter log function using prefix close language eliminates all traces that are not a prefix of another prefix in the log by using a frequency threshold defined by the user.

The first filter applied was an event-type filter, that allowed the user to select multiple types of events, tasks, or audit trail entries for consideration while mining the log. All event types were kept. The second filter was applied to keep only traces or cases that start with indicative tasks. A threshold was adjusted to 90%, meaning that the most frequent start events were selected until at least 90% of the traces were covered. The third filter

was an end event filter, which kept only traces or cases that ended with the indicated tasks. The range was again set to 90% to select the most frequent end events.

The final filter revealed all unselected events from the log. The threshold was set to 100% to select the most frequent events. An example of a filter selection process is indicated in Figure 8.

Figure 8. Filter for selecting end event threshold

After using the filter log function in Simple Heuristics plugin, ILP Miner and Inductive Miner were applied to discover Petri nets. Finally, the quality of discovered algorithms was measured using quality metrics. ProM has a specific quality metrics plugin to compute projected fitness and precision. It also has a plugin to show Petri net metrics for calculating generalisation and complexity grades. The whole experiment process is illustrated in Figure 9.

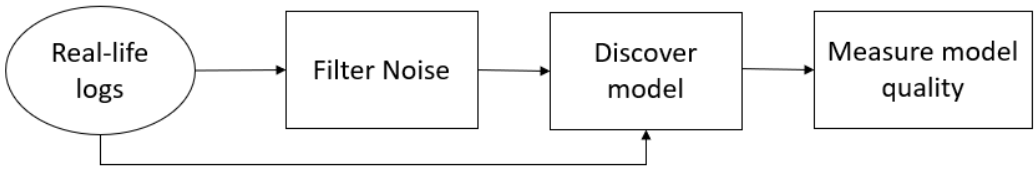

Figure 9. Experiment setup for real-life logs

3.3 Results

As the experiment result, the Inductive Miner model is the best algorithm for handling noise when the log has fewer event activities and less noise. The Petri net produced by this algorithm exhibits a good simplicity level, meaning it has fewer arcs and places, and has a much higher F-score result of 0.86. However, when event logs are more complex and contain more events, the ILP Miner algorithm produces slightly better results with an F-score of 0.67 compared to a score of 0.56 using Inductive Miner. Apparently, the ILP Miner algorithm can replay more relations than the Inductive Miner algorithm. Additionally, the clarity of the Petri net is better when comparing visually generated models.

The performance of the algorithms to generate the Petri nets was acceptable up until 15 minutes. The generated Petri nets by the ILP Miner algorithm are presented in Figures 10 and 11, and by the Inductive Miner algorithm in Figures 12 and 13.

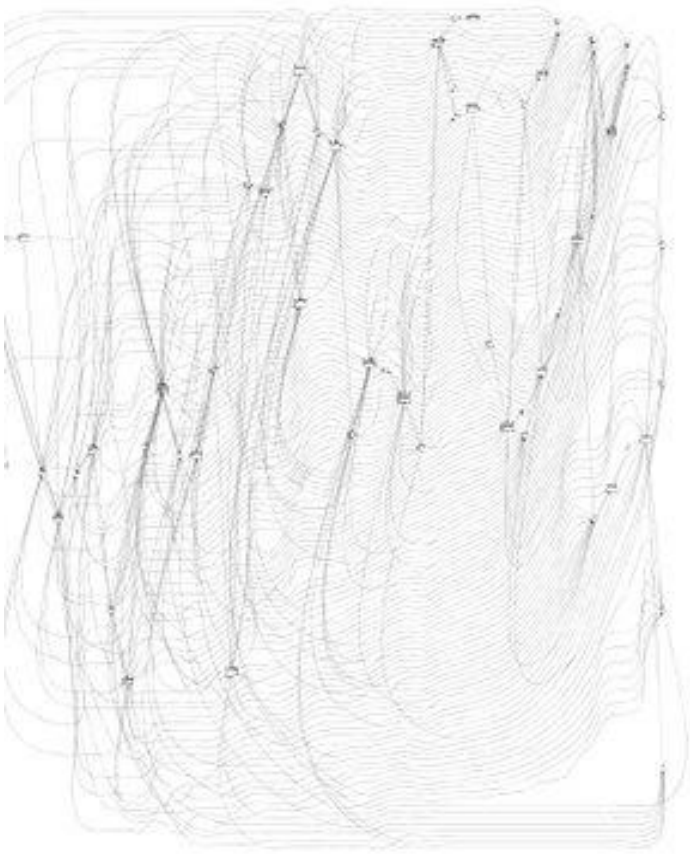

Figure 10. Petri net generated by ILP Miner algorithm of BPI 2012 event log

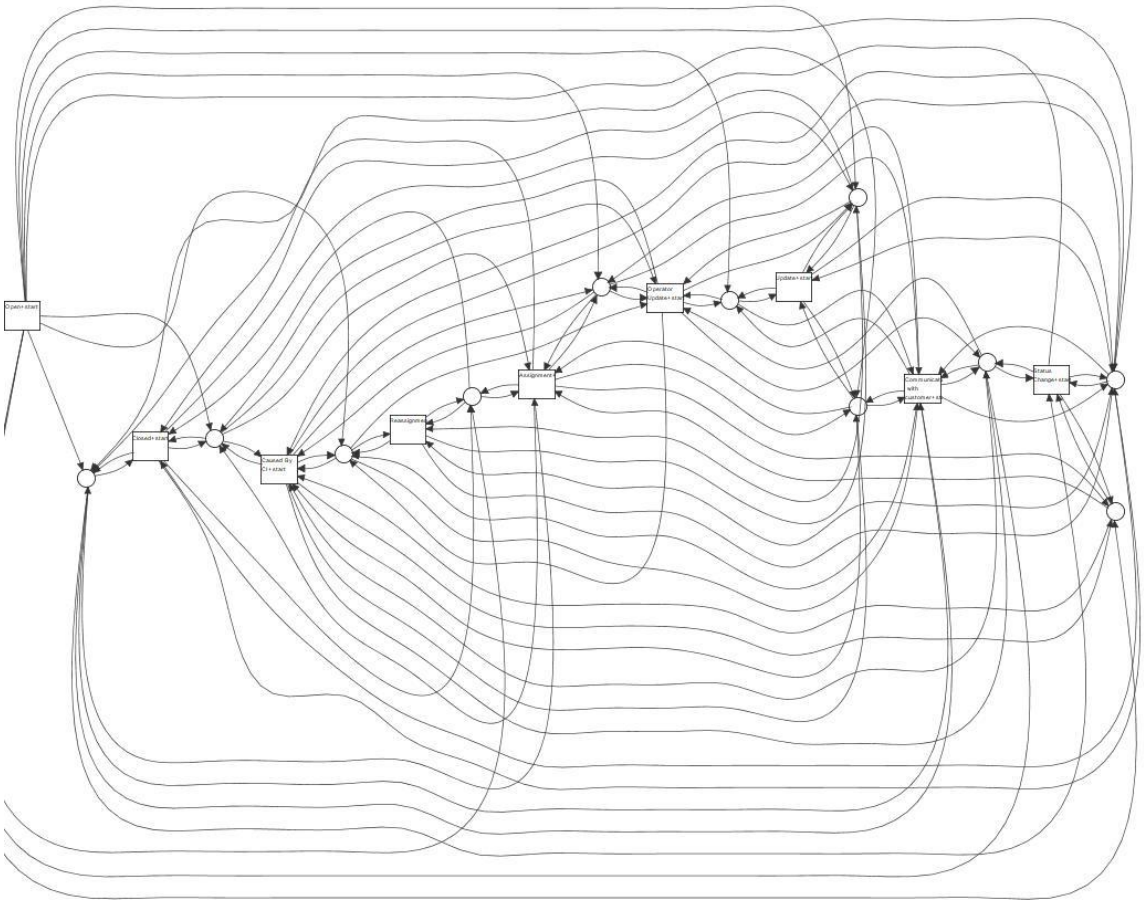

Figure 11. Petri net generated by ILP Miner algorithm of BPI 2014 event log

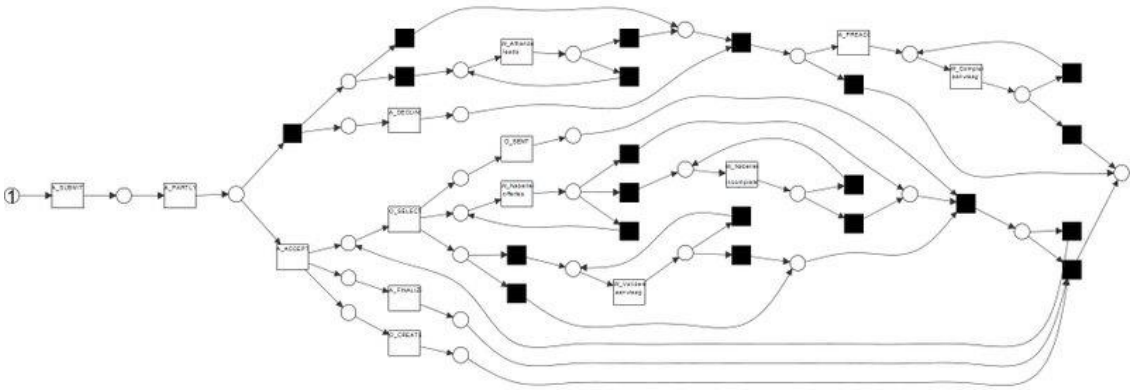

Figure 12. Petri net generated by Inductive Miner algorithm of BPI 2012 event log

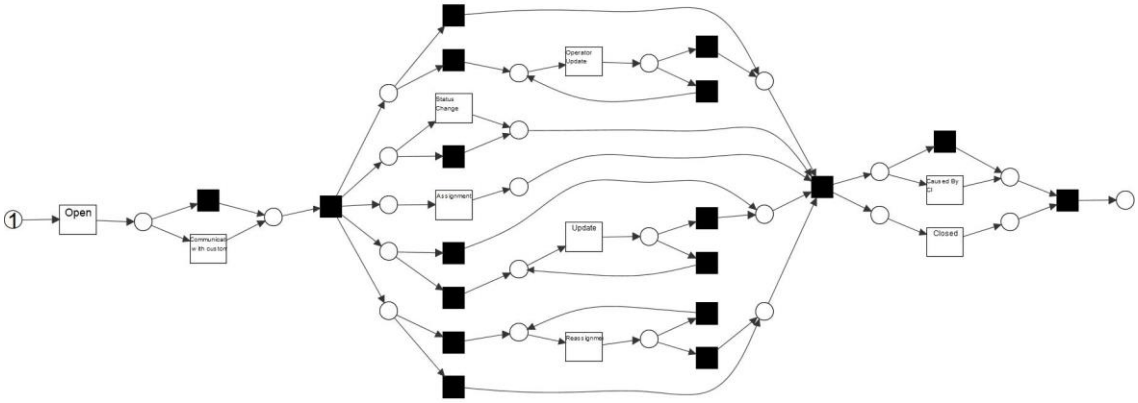

Figure 13. Petri net generated by Inductive Miner algorithm of BPI 2014 event log

The results of the quality metrics are presented in Table 3.

	Algorithm	ILP Miner	Inductive Miner
BPI2012	F-score	0.35	0.86
	Number of arcs	332	80
	Number of places	36	30
	Number of transitions	23	35
	Density	0.2	0.038
BPI2014	F-score	0.67	0.56
	Number of arcs	97	64
	Number of places	11	24
	Number of transitions	9	27
	Density	0.489	0.049

Table 3. F-score and complexity metrics

3.4 Limitations

The outlier detection model is a simplified way of sanitising the log before applying discovery model algorithms. One of the biggest disadvantages of the selected approach is that the method removes the traces that contain either noise or infrequent but correct behaviour. The noise does not mean incorrect data in the event log. The event log may also contain infrequent behaviour or outliers. Such behaviour usually leads to the creation of exceptional paths in the process. It is possible that the selected approach may remove traces that contain critical and useful information. Not always this might be a critical issue, but in specific applications like medical treatment or compliance investigations, such information could prove to be meaningful. To improve the accuracy of the model even further the classification rules may be applied to the log to detect noisy traces. For this, manual input is needed for the classification algorithm to generate rules. In addition, rules generated can be checked manually and corrected by experts.

Another limitation of this research is that only two algorithms were selected for comparison due to long processing times. To produce a Petri net using other algorithms such as Fodina or Heuristics Miner proved to be challenging due to the limited computing power available. A computer with better processing capability could help to reduce the processing times.

4. CONCLUSIONS

4.1 Discussion

Process mining has become an effective tool for companies to analyse their processes. This study aimed to identify the main challenges related to data quality in process mining. The main challenges were identified as representational bias, concept drift, noise, and incompleteness. Noise is one of the main challenges when it comes to working with real-life logs. Logs contain errors or infrequent behaviour which prevents generating a good quality process model. The quality of the model is measured by quality metrics such as precision, fitness, complexity, and generalisation. Noise level must be reduced to generate a process model that can be used to discover, check compliance, and monitor processes in business.

This study aimed to identify the existing approaches for handling noise and compare them. The existing solutions for handling noise have been improved since the first developed α-algorithm that is assuming that the event log is noise free. Algorithms such as Heuristics Miner, Fuzzy Miner, Integer Linear Programming (ILP) Miner, and Inductive Miner have been developed to tackle noise.

The initial aim of this study was to test and compare multiple major process mining algorithms in removing noise. However, when running the algorithms, the processing capacity of the computer to generate process models soon became a limiting factor. It took several days for the computer to run the algorithms, and eventually not generate any results. Due to such limitations, only the Inductive Miner and the ILP Miner algorithms were selected for comparison as these algorithms were able to generate results. Event log data was filtered using the Simple Heuristics plugin. However, there was no measurement of how clean the data became after applying the filter. A more sophisticated approach is needed to indicate how clean the data is before applying algorithms to the event log.

Handling noise in the generation of Petri nets is a critical aspect of process mining since Petri nets are a fundamental component in checking process compliance. The study of available algorithms and their performance on real-life event logs revealed that the ILP Miner algorithm produces a slightly better F-score when the log is complex and has more noise. This is because the ILP Miner algorithm is better able to replicate connections between event activities. However, the Petri net produced by the ILP Miner algorithm is much more complex in terms of complexity metrics, making it more difficult for business users to understand.

Alternatively, for simpler logs with fewer events, cases, and less noise, the Inductive Miner algorithm outperforms the ILP Miner algorithm by a higher F-score and better performance in complexity metrics. In areas where outliers are crucial, such as compliance checks related to fraud or medical use cases, removed activities should be additionally checked by an expert to ensure that important but infrequent activities are not filtered out.

4.2 Future research

As future research, it is recommended to expand the algorithmic comparison by incorporating additional algorithms such as Fodina and Heuristics Miner. Furthermore, it could be beneficial to explore ways in which the algorithms can be combined with human review to enhance results and prevent the exclusion of crucial logs that contribute to understanding the process model. Additionally, investigating various algorithms' impact on the quality parameters of the produced Petri nets could be valuable.

Future research could also focus on other quality issues that have not been widely studied such as concept drift. As companies operate in dynamic environments, the processes change constantly and an effective way to visualise changes through time to process model is needed.

REFERENCES

Acosta-Velásquez, R. D., León-Pulido, J., García-Pérez, A., Fajardo-Moreno, W. S., & Espinosa-Leal, L. (2022). Contemporary Management Practice Applying the Dynamic Absorptive Capacity Measurement Model (PM^4AC) for Improved Business Sustainability. Sustainability, 14(17), 11036. https://doi.org/10.3390/su141711036

Adriansyah, A. A., Munoz-Gama, J., Carmona, J., van Dongen, B. F., & van der Aalst, W. M. P. (2012). Alignment Based Precision Checking. Springer Berlin Heidelberg eBooks, 137-149. https://doi.org/10.1007/978-3-642-36285-9_15

Claes, J., & Poels, G. (2013). Process mining and the prom framework: An exploratory survey. Business Process Management Workshops, 187-198. https://doi.org/10.1007/978-3-642-36285-9_19

de Medeiros, A. K. A., van Dongen, B., van der Aalst, W. M. P., & Weijters, T. (2004). Process Mining: Extending the α-algorithm to Mine Short Loops. TU Eindhoven, 113. https://doi.org/10.1007/756-6-842-83454_52

de Medeiros, A. K. A., Weijters, A. J., & van der Aalst, W. M. P. (2007). Genetic process mining: An experimental evaluation. Data Mining and Knowledge Discovery, 14(2), 245-304. https://doi.org/10.1007/s10618-006-0061-7

Espinosa-Leal, L., Chapman, A., & Westerlund, M. (2020). Autonomous industrial management via reinforcement learning. Journal of intelligent & Fuzzy systems, 39(6), 8427-8439. https://doi.org/10.3233/JIFS-189161

Grubbs, F. E. (1969). Procedures for Detecting Outlying Observations in Samples. Technometrics, 11(1), 1–21. https://doi.org/10.1080/00401706.1969.10490657

Günther, C. M., & van der Aalst, W. M. P. (2007). Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. Lecture Notes in Computer Science, 328-343. https://doi.org/10.1007/978-3-540-75183-0_24

Hand, D. J., Mannila, H., & Smyth, P. (2001). Principles of data mining (1st ed.). The MIT Press.

How does process mining work? (2023, May). Celonis. https://www.celonis.com/process-mining/how-does-process-mining-work

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. International Joint Conference on Artificial Intelligence, 2, 1137–1143. https://ijcai.org/Proceedings/95-2/Papers/016.pdf

Leemans, S. S., Fahland, D., & van der Aalst, W. M. P. (2013). Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. Business Process Management Workshops, 66-78. https://doi.org/10.1007/978-3-319-06257-0_6

ProM Tools (2023, May). ProM. https://promtools.org

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. Big Data, 1(1), 51-59. https://doi.org/10.1089/big.2013.1508

Reinkemeyer, L. (2020). Process Mining in Action: Principles, Use Cases and Outlook (1st ed.). Springer Nature.

Rozinat, A. A., Veloso, M., & van der Aalst, W. M. P. (2008). Evaluating the quality of discovered process models. European Conference on Principles of Data Mining and Knowledge Discovery, 45–52. https://pure.tue.nl/ws/files/3045384/Metis219033

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210-229. https://doi.org/10.1147/rd.33.0210

van der Aalst, W., Buijs, J., & van Dongen, B. (2012). Towards improving the representational bias of process mining. Lecture Notes in Business Information Processing, 39-54. https://doi.org/10.1007/978-3-642-34044-4_3

van der Aalst, W. M. P. (2016). Process Mining: Data Science in Action (2nd ed.). Springer.

van der Aalst, W. M. P., Desel, J., & Oberweis, A. (2003). Business Process Management: Models, Techniques, and Empirical Studies (1st ed.). Springer.

van der Aalst, W. M. P., van Dongen, B. F., Günther, C. W., Rozinat, A., Verbeek, E., & Weijters, T. (2009). ProM: The process mining toolkit. BPM (Demos), 489(31), 2. https://doi.org/10.1007/934-3-532-68455-875

van der Aalst, W. M. P., Weijters, T., & Maruster, L. (2004). Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16(9), 1128–1142. https://doi.org/10.1109/tkde.2004.47

van der Werf, J. M., van Dongen, B. F., Hurkens, C. A. J., & Serebrenik, A. (2009). Process Discovery Using Integer Linear Programming. Lecture Notes in Computer Science, 368–387. https://doi.org/10.1007/978-3-540-68746-7_24

van Hee, K. M. & Reijers, H. A. (2000). Using Formal Analysis Techniques in Business Process Redesign. Lecture Notes in Computer Science, 142–160. https://doi.org/10.1007/3-540-45594-9_10

Wang, L., Fang, X., & Shao, C. (2022). Discovery of business process models from incomplete logs. Electronics, 11(19), 3179. https://doi.org/10.3390/electronics11193179

Weijters, A. J., & Ribeiro, J. (2011). Flexible Heuristics Miner (FHM). Computational Intelligence and Data Mining. https://doi.org/10.1109/cidm.2011.5949453

Wen, L., van der Aalst, W. M. P., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery, 15(2), 145–180. https://doi.org/10.1007/s10618-007-0065-y

Wen, L., Wang, J., van der Aalst, W. M. P., Huang, B., & Sun, J. (2009). A novel approach for process mining based on event types. Journal of Intelligent Information Systems, 32(2), 163–190. https://doi.org/10.1007/s10844-007-0052-1

.