UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Vergelijkende studie van process discovery algoritmes

Dries Jarijch

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :
dr. Gert JANSSENSWILLEN

COPROMOTOR :
dr. Frank VANHOENSHOVEN

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

2023
2024

UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Vergelijkende studie van process discovery algoritmes

Dries Jarijch

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :
dr. Gert JANSSENSWILLEN

COPROMOTOR :
dr. Frank VANHOENSHOVEN

Comparative study of process discovery algorithms

Dries Jarijch

Abstract. Process discovery remains one of the most interesting, yet challenging, aspects of process mining. As a result from the existence of many different mining algorithms, it is unclear which miner should be used under certain conditions. This paper a comparison is made between the current state of the art process mining algorithms to determine their strengths and weaknesses. A literature study is performed to determine the best performing discovery algorithms. The best performing process miners are the Split miner and the Inductive miner. In addition, the recently developed BupaRminer is also used. Based on the results of the experiment, the Inductive miner outperforms both other miners in regard to fitness and precision. Split miner is able to discover much simpler models when compared to both other algorithms. These miners perform best on these metrics regardless of system complexity, noise in the log, or completeness of the log. The biggest weakness of the BupaRminer is with discovering loops when noise is introduced. For the Split miner, it is the ability to generate unsound models and its lack of precision.

Keywords: Process Discovery · BupaRminer · Inductive Miner · Split Miner

1 Introduction

Process discovery remains one of the most interesting, yet challenging, aspects of process mining [4]. The aim of process discovery algorithms is the discovery of process models based on observed data. The goal of discovered models is to reconstruct the true underlying process that allowed for the observed data to be recorded [4, 53]. In the past two decades, a lot of algorithms have been made [7, 8, 12, 14, 16, 17, 20, 22, 27, 31–37, 39, 43, 47, 49–52].

As a result from the existence of the many mining algorithms, a challenge arises. It is unclear which miner should be used under certain conditions. Additionally, in existing studies, datasets are often chosen ad hoc, which might be optimal for some miners, which can be to the detriment of others. In other studies, the basis for comparison is real-life datasets for which the underlying system is usually unknown [13]. Furthermore, many of these miners have parameters which are not commonly closely examined in literature.

This paper aims to provide a comparison between current state of the art process mining algorithms to determine their strengths and weaknesses. Based on recent literature, the best performing process miners are the Split miner and the Inductive miner [13]. In addition, the recently developed BupaRminer is also used. Artificial logs are simulated from artificially generated systems. New models are discovered from these logs and evaluated on different quality

dimensions. From this a conclusion is drawn on which miners perform best under which circumstance.

The paper is constructed as follows: Section 2 discusses related work. Section 3 elaborates on the research questions posed in this paper. Section 4 describes the methodology of the research. Section 5 shows the results, which are discussed in Section 6. Section 7 concludes the paper.

2 Related Work

In the following sections, an overview is given of existing literature regarding: commonly used evaluation metrics in Section 2.1, existing discovery algorithms in Section 2.2, and previously conducted comparative studies in Section 2.3. Based on this information, a selection is made for which algorithms and evaluation metrics will be used when performing the experiment. Evaluation metrics are discussed first to facilitate the discussion of the mining algorithms.

2.1 Quality Dimensions

There are several dimensions to evaluate the quality of a discovered model. This section discusses the most commonly used quality dimensions used in literature.

Fitness Fitness as a quality dimension is first proposed in [44], where it is defined as "how much behavior in a log is correctly captured (or can be reproduced) by a model". There are several different metrics associated with the fitness quality dimension:

- Completeness or Parsing Measure (CP): "which percentage of the traces in a log can also be generated by a model" [44]
- Continuous Parsing Measure (CPM): "the total number of correctly replayed traces and the total number of correctly replayed tasks." [44]
- Token-based fitness: "considers which problems (missing and remaining tokens) happened during the log replay." [45]
- $PF_{complete}$: Similar to Token-based replay, but it also takes into account trace frequencies when weighing the problems that occurred during the log replay.[38]
- Behavioral fitness: checks how much behavior is allowed by the original model that is not by the mined model [39]
- Negative event fitness (Behavioral recall): the proportion of the behaviour in the event log which can be replayed without forcing transitions to fire. While this is the same as the base fitness metric, the precision and generalisation version of negative event metrics are impacted [18].
- Structural fitness: structural fitness assess how many causality relations the original model has that are not in the mined model [39]
- Alignment-based fitness (F_a): "measures the degree to which every trace in the log can be aligned (with a small number of errors) with a trace produced by the model." [9]

Precision Precision and generalisation are defined in [44] as "how much more behavior is captured in the discovered model than was observed in the log". More specifically, Precision refers to the amount of behavior that is allowed by a discovered model, but is not present in the log. Precision metrics include:

- a'_B: "derives "sometimes" follows and precedes relations (reflecting alternative or parallel behavior) for tasks in a log and for tasks in a model, and compares these relations. The less relations derived from the model can also be derived from the log, the less precise is the model." [44]
- Behavioral precision (B_P): "checks how much behavior is allowed by the mined model that is not by the original model" [39]
- Negative event precision (Behavioral precision): Negative events are artificially introduced. These are events that should not be allowed by the model. Behavioral precision measures the ratio of positive events (all non-negative events) allowed by the model to all events allowed by the model [18].
- Structural precision: "structural precision assess how many causality relations the mined model has that are not in the original model" [39]
- Token-based precision: "different prefixes of the log are replayed (where possible) on the model. At the reached marking, the set of transitions that are enabled in the process model is compared with the set of activities that follow the prefix. The more the sets are different, the more the precision value is low." [41]
- Alignment-based precision (P_a): calculated by confronting model and log behaviors. Imprecision between the model and the log (i.e., situations where the model allows more behavior than the one reflected in the log) are detected and analyzed [10, 11].

Harmonic mean (F1-score): a way of combining precision and fitness into one metric [13]. It takes a precision and fitness metric and combines them in the following equation:

$$F_1 = \frac{2 * f * p}{f + p} \tag{1}$$

Generalisation Generalization is also first mentioned in [44]. It serves as an indicator of how much a discovered model might be over-fitting to the log. The following metrics belong to this quality dimensions:

- Negative event generalisation (Behavioral generalisation): this is the ratio of allowed generalised events to all generalised events. Generalised events are events which were not observed but at the same time not considered as negative [18].
- Alignment generalisation: starts from an aligned log. for each event, calculates the probability that the next time this state is visited, a new path will be observed. If the average probability over the log is low, then generalization is assumed to be high [5].

- K-fold cross-validation: Sections of a model are held out, then fitness is generated for the part held out against the discovered model. The mean value of the fitness values serves as an indicator of generalisation [13]
- Causal Footprint: "measures behavioral similarity based on two models' structures. It works by (i) mapping these models to their causal closure graphs, (ii) transforming these graphs to vectors in a multidimensional space, and (iii) measuring the cosine distance between these two vectors." [44]

Simplicity Simplicity was first referred to as complexity in [30]. More recent literature has adapted the name simplicity for this quality dimension [19]. It is defined as "how simple the process model is to understand for a human". Several commonly used metrics include:

- Coefficient of Connectivity (CNC): "The number of arcs divided by the number of nodes. In the context of a business process model, the number of arcs has to be divided by the number of activities, joins, and splits" [29]
- Control Flow Complexity (CFC): "the sum over all connectors weighted by their potential combinations of states after the split." [23]
- Size: "The number of nodes of the process model" [30]
- Diameter: "The diameter gives the length of the longest path from a start node to an end node in the process model" [40]
- Structuredness: "how far a process model can be built by nesting blocks of matching join and split connectors" [30]
- Average Degree: "ratio of the number of outgoing and incoming flows over the size of a model" [46]
- Average Connector Degree: "The number of nodes a connector is on average connected to." [30]

Soundness Measures whether a model violates one of the three soundness criteria defined in [3]. These criteria can be assessed by playing the token game on a petrinet:

- Option to complete: For every state M reachable from state i, there exists a firing sequence leading from state M to state o. From initiation to completion, every reachable state has a way of reaching completion.
- Proper completion: State o is the only state reachable from state i with at least one token in place o. Meaning that if a petrinet reaches completion, there are no lingering tokens in the net left over.
- No dead transitions: There are no transitions that can never fire.

Quality dimensions can be subdivided based on two characteristics. The first is their point of comparison. For the fitness, precision and generalisation dimension, this point of comparison is between a log (often a system log) and a discovered model. For simplicity and soundness, only the discovered model is used to determine the quality. The second characteristic is way of measuring.

Fitness, precision, and generalisation are often expressed as values between 0 and 1. Simplicity metrics often not bound between these values but are still numeric. Soundness is a metric that is either pass or fail. A model can be sound or not sound.

2.2 Discovery Algorithms

Table 1: Comparison of existing discovery algorithms.

Algorithm name	Year	Output	life-cycle components	Sound output
α-miner	2004	Petrinet	No	No
genetic miner	2005	Petrinet	No	No
heuristics miner	2006	BPMN-model	No	No
fuzzy miner	2007	MR-Object	No	No
genetic miner 2	2007	Petrinet	No	No
$\alpha++$-miner	2007	Petrinet	No	No
β-miner	2009	Petrinet	Yes	No
heuristics++ miner	2010	BPMN-model	Yes	No
flexible heuristics miner	2011	Petrinet	No	No
Evolutionary tree miner	2012	Process tree	No	Yes
Inductive miner	2013	Process tree	No	Yes
Inductive - infrequent	2014	Process tree	No	Yes
Inductive - incompleteness	2014	Process tree	No	Yes
α^{II}-miner	2016	Petrinet	No	No
Inductive - life cycle	2016	Process tree	Yes	Yes
Fodina	2017	BPMN-model	No	No
Split miner	2019	BPMN-model	No	No
eST-miner	2019	Petrinet	No	?
Inductive - Probabilistic	2021	Process tree	No	Yes
Split miner 2.0	2021	BPMN-model	Yes	No
Improved eST-miner	2023	Petrinet	No	?
BupaRminer	2023	BPMN-model	Yes	Yes
eST-miner variant	2024	Petrinet	No	?

Alpha Miner There are many existing process miners that have been developed in the past two decades. The first notable one is the α-miner proposed in [8]. As the first miner, it still had some major shortcomings. The main one was not being able to handle short-loops, but also not being able to discover non-free-choice processes and unsound models [8].

A significant improvement was made upon the α-miner in [51] where the $\alpha++$-miner is proposed. This version of the α-miner allows for the rediscovery of

most, though not all, sound models that include short-loops. The main advantage of this miner is being able to discover non-free-choice models [51].

An extention of the original α-miner was also developed in [52] called the β-miner. This version allows for discovering models based on logs with life-cycle components [52]. This is something the base α-miner is not capable of doing.

The most recent adaption of the α-miner is the α^{II}-miner developed in [35]. It looks at eventually-follows relations between events when discovering models. This change to the α-miner targets parallel business processes. [35].

Most versions of the α-miner have a strong focus on simplicity, but perform poorly for both fitness and precision metrics [20].

Heuristics Miner The heuristics miner was first proposed in [49] as a business oriented approach to process discovery. This is achieved by giving the option to express only the main behavior of a process rather than all behavior. This proves itself to be a particularly flexible and noise resistant process miner [49]. Finally, it has a unique focus on ease of interpretation [17].

Like the α-miner, the heuristics miner also does not inherently work with life-cycle components in logs. The heuristics++ miner proposed in [22] is capable of discovering models with these life-cycle components [22].

An improvement was made on the original heuristics miner, called the flexible heuristics miner. The basic ideas of the previous versions were not exploited to their full potential, which this version accounts for. In addition to being more noise resistant, the flexible heuristics miner is now also adept at mining processes from less structured environments [50]. In [17] several shortcomings were still identified for the (flexible) heuristics miner for particularly complex logs: tasks can be unconnected, duplicate tasks cannot be mined, and long distance dependencies are sometimes overly sensitive [17].

Fodina is the most recent variant of the heuristics miner, developed in [17]. This miner has proven itself more robust than all previous versions. Furthermore, it shows the capability to mine duplicate tasks, which no other variant is currently able to do [17].

Genetic Miners Currently, three major genetic mining algorithms exist. The first was proposed in [7]. It discovers models over several iterations, using the best fitting models as input for the next iteration. Its main benefit is its resistance noise and incompleteness [7].

Another genetic miner was developed in [39]. The focus for this genetic miner is to account for: non-free-choice constructs, invisible tasks and duplicate tasks [39].

The Evolutionary Tree miner (EVT), proposed in [20], uses a genetic algorithm at its core. Rather than focusing on one to two performance metrics at a time, it attempts to strike a balance between the main four: fitness, precision, simplicity, and generalisation. This is an atypical approach to process discovery. A significant importance is given to user guidance. This allows for certain met-

rics to be prioritised over others, but also requires more user input as a result [20].

The biggest drawback of most genetic models are their long run-times [33].

Fuzzy miner As of yet, the fuzzy miner has only seen one version, described in [27]. The paper critiques the genetic mining algorithm proposed in [7]. Genetic algorithms are particularly adept at handling noise and incompleteness, but are limited by their focus on discovering a true underlying process. The fuzzy miner has a focus on optimizing process discovery for processes from real-life logs and unveiling hidden knowledge by avoiding spaghetti processes. It assumes real-life logs are not entirely reliable and there is not one true model. The fuzzy miner preserves highly significant behavior, and abstracts less significant behavior, this makes it ideal for discovering real-life processes [27].

Inductive Miner The base version of the Inductive miner is proposed in [31]. It points out that most other, older discovery techniques (α-miner, heuristics miner, fuzzy miner, and genetic miner) have little guarantee regarding fitness and soundness. The Inductive miner was among the first algorithms to have this focus on fitness and soundness [31]. As a result, it is currently among the top performing miners in process discovery [13].

The Inductive miner - infrequent (IMi) was developed in [33]. The main focus is on speed and being able to filter out infrequent behavior. Following the Pareto principle, it is used to discover 80% of the most frequent behavior seen in logs. Some of the high fitness inherent to the base version of the Inductive miner is sacrificed in favor of precision and some simplicity [33]. Infrequent behavior is considered to be one of the two biggest problems for process discovery algorithms [32].

Inductive miner - incomplete (IMin) addresses the other big problem discovery algorithms face: rediscovering processes from incomplete logs. This is done by introducing new relations between activities, which were not observed in the log. Then the probabilities of these new relations are estimated, and kept if deemed likely enough. Sufficiently large logs are still required to discover adequate models while maintaining simplicity [32].

All other implementations of the Inductive mining algorithm that have been discussed so far, assume that events in logs are atomic. The Inductive miner - lifecycle (IMlc) is able to use life-cycle components when discovering models from logs. Some limitations are still identified: this version has difficulty detecting hidden relations between events and with expressing multiple instances of the same activity. Hidden relations are the result of a non-free-choice construct where the the execution of one activity is only possible if another activity was performed previously in the process. [34].

The probabilistic Inductive miner (PIM) presented in [16] is intended to strike a balance between IMi and IMin [33, 32]. Both miners individually only tackle one of the big problems that discovery algorithms tend to face. While outperforming most other algorithms it was compared to (see Section 2.3), the

probabilistic Inductive miner sacrifices on time by taking hundreds of times longer to complete [16].

Split miner In [12], two issues are identified with regards to existing mining algorithms: 1) Some algorithms often produce spaghetti-like models which are hard to interpret or 2) they discovery models with poor fitness or precision. Instead a proposal for the Split miner is made. The Split miner algorithm balances precision, fitness, and generalisation. The discovered models are comparable in simplicity to the Inductive miner and evolutionary tree miner [31, 20]. Models are also always sound and deadlock-free. It claims to be the first algorithm capable of doing this [12].

Split miner 2.0, discussed in [14], is the most recent version of Split miner with two main additional features: 1) it is able to use life-cycle components when discovering models and 2) it has the ability to differentiate between inclusive and exclusive choices (OR and XOR) [14].

eST miner The eST-miner, proposed in [36], takes a unique approach to process discovery. By using language based regions [15], much more detailed and complex processes can be discovered. As a result, discovered processes are much more complex. This algorithm performs particularly well in ways of computation times and when discovering non-free choice constructs [36].

The eST miner in criticised in [43] for often discovering imprecise substructures, such as one-loops. The improved eST miner proposed in [43] maintains the high efficiency of the original eST miner, while reducing the number of these imprecise substructures. Additionally, it improves simplicity by removing implicit connections. These are connections which do not impact behavior when removed [43].

In [37] another issue is identified: deadlocks. The base eST-miner is likely to generate a deadlock where the end of a process cannot be reached. This problem is addressed by the variant of the eST miner proposed in this paper. It greatly improves the handling of places connected to rarely occurring activities. In doing so, it reliably prevents deadlocks from appearing [37].

BupaRminer BupaRminer is proposed in [47]. The focus is on discovering sound process models. There is no requirement for fine-tuning hyperparameters, which increases the ease of use for novice users. It is inherently noise resistant and inherently incorporates working with life-cycle components. It operates in a two-staged approach. First, the algorithm estimates likely relationships between pairs of activities. The result can be seen as a declarative model. Second, these relationships are aggregated into a sound BPMN model. [47].

2.3 Other comparative studies

Table 2: Overview of other comparative studies.

Author	Date	Most prominent miners	Datasets	Quality measures
De Weerdt [24]	2012	α-miner [8] Heuristics miner [49] Genetic miner [39] Fuzzy miner [27] $\alpha++$-miner [51] β-miner [52]	20 artificial logs also used in [38] 8 real-life event logs from information systems in: a university, a manufacturing company, a telecom company, and an insurance company	Fitness Completeness Parsing measure Successfully executed traces Behavioral recall Behavioral appropriateness Advanced behavioral appropriateness Soundness Behavioral specificity Behavioral precision ETC precision The harmonic mean (F-score)
Esmita [25]	2014	Heuristics miner [49] Fuzzy miner [27] Genetic miner [7]	No datasets used	No measures used
Augusto [13]	2019	Inductive miner [33] Split miner [12] Heuristics miner [49] Fodina [17] EVT [20]	Twelve logs from 4TU Centre for Research Data [1] Twelve proprietary logs from several international companies	Alignment based fitness Alignment based precision The harmonic mean (F-score) K-fold cross-validation Size Control-flow complexity Structuredness Soundness
Peng [42]	2021	α-miner [8] Heuristics miner [49] Inductive miner [33]	Twelve logs from 4TU Centre for Research Data [1]	Behavioral Fitness Structural Fitness Behavioral Precision Structural Precision The F1 score Running time
Augusto [14]	2021	IM - life-cycle [34] Split miner [12] Split miner 2.0 [14]	Eleven proprietary logs from several international companies	Alignment-based Fitness Alignment-based Precision Size Control-flow Complexity (CFC) Structuredness Soundness
Brons [16]	2021	Inductive miner [33] Split miner [12] EVT [20] PIM [16]	Fourteen logs from 4TU Centre for Research Data [1] Two proprietary logs	Fitness Precision The F1 score Size Control-flow Complexity Structuredness Soundness

The first notable comparative study is the one conducted in [24]. For all algorithms, the standard settings was chosen. While choosing more optimal parameters could improve results, it would also require prior knowledge, which is seldom available. For artificial event logs, the genetic miner Genetic miner [39] performed outstandingly compared to other miners, while both the heuristics miner and the α-miner (including its variants) [8, 49, 51, 52] underperform with regards to their fitness metrics. For real-life event logs, the heuristics miner outperforms the other algorithms on the different precision metrics used. It is also reported to have the fastest running times. The base α-miner and its veriants were once again among the worst performing miners.

A later comparison performed in [25] compares the heuristics miner [49], the fuzzy miner [27], and the genetic miner [7]. It does not appear to use any basis for evaluating each of the different miners. It does provide guidelines for the context in which each miner should be used, but provides no further proof [25].

A later benchmark study performed in [13] compares 35 different miners, the most prominent of which include: the Inductive miner [33], the Split miner [12], the heuristics miner [49], fodina [17], and the evolutionary tree miner [20].

In the benchmark study, two forms of evaluation are used: 1) with default hyperparameters, and 2) with hyperparameter optimalisation. Notably, for the second evaluation, EVT was left out due to time constraints. in both scenarios all miners were consistenly outperformed by [13]: 1) Inductive miner on the basis of fitness, 2) Evolutionary tree miner on the basis of precision, and 3) Split miner on the basis of the harmonic mean. This paper was criticized in [16] for having three shortcomings: Logs are prefiltered to reduce complexity, logs were too small, and certain common process types are missing.

A comparative study in [42] compares the performance of the α-miner [8], heuristic miner [49], and Inductive miner [31]. In addition to the base version of the algorithms, several of its derivatives are also used, although only two of these derivatives were cited (IMi [33] and IMin [32]). In addition, several types of noise are introduced to the datasets: 1) Missing head noise, 2) Missing tail noise, 3) Missing episode noise, 4) Perturbed order noise, 5) Double activity noise, 6) Alien activity noise, and 7) Change name noise.

The heuristics miner performs best on the F1 score. The Inductive miner outperforms the rest on all fitness metrics. The α-miner performs best on running time. This remains consistent when the miners are exposed to noise.

[14] compares Split miner 2.0 (where it is first introduced) with the Inductive miner-life-cycle, and the base Split miner. The base Split miner was the only miner to discover unsound models. The Inductive miner scores the highest on fitness. Split miner 2.0 usually performs best on precision, but is sometimes matched by the base Split miner. Split miner 2.0 consistently performs best on all simplicity metrics.

The paper in which the Probabistic Inductive miner [16] is proposed, it is also compared to EVT [20], Inductive miner [33], and Split miner [13]. For the comparison of the PIM, a more simplictic version of PIM was also implemented

for the comparison (PIM_{30}) to have lower run-times. This is done on the basis of the following metrics:

PIM consistently sacrifices fitness for increased precision, even out-performing EVT. PIM also scores best on average for all simplicity metrics. Split miner achieves the highest fitness, while the hightest F1-score is achieved by the Inductive miner [16].

3 Problem Statement

The purpose of this paper will be to compare the current state of the art process mining algorithms. In order to achieve this, two focal points are defined. First, a focus on noise resistance. This will measure how well the process miners can handle the presence of noise and incompleteness in logs. Second, a focus on system variation. This will compare how the different process miners perform under varying conditions such as noise, completeness, and complexity. To evaluate these problems, two research questions are defined:

1. RQ1: Which miners perform best when exposed to different levels of noise and completeness.
2. RQ2: Which miners perform best when exposed to data from systems with differing levels of complexity.

4 Methodology

The methodology in this paper is based on the framework presented in Weber (2011) [48]. For comparing process mining algorithms, the following six steps are given by this framework:

1. Generate a ground truth system
2. Calculate number of paths
3. Simulate logs, including a very large, complete log
4. Discover models using different process mining algorithms
5. Measure the quality of the discovered models
6. Statistical analysis using measurements from different simulations

4.1 Generating ground truth models

For the generation of ground truth systems, the Purpose-Guided Log Generation framework proposed in [21] is used. This framework provides a tool (PLG2) that allows for the generation of systems based on different parameters:

Within the context of this paper, three levels of complexity for system generation are defined, as can be seen in Table 4. For each of these systems, four different configurations of inputs are used in PLG2: without loops or skips, a configuration containing loops, a configuration containing skips, and a configuration containing both loops and skips. The percentages of each of these patterns

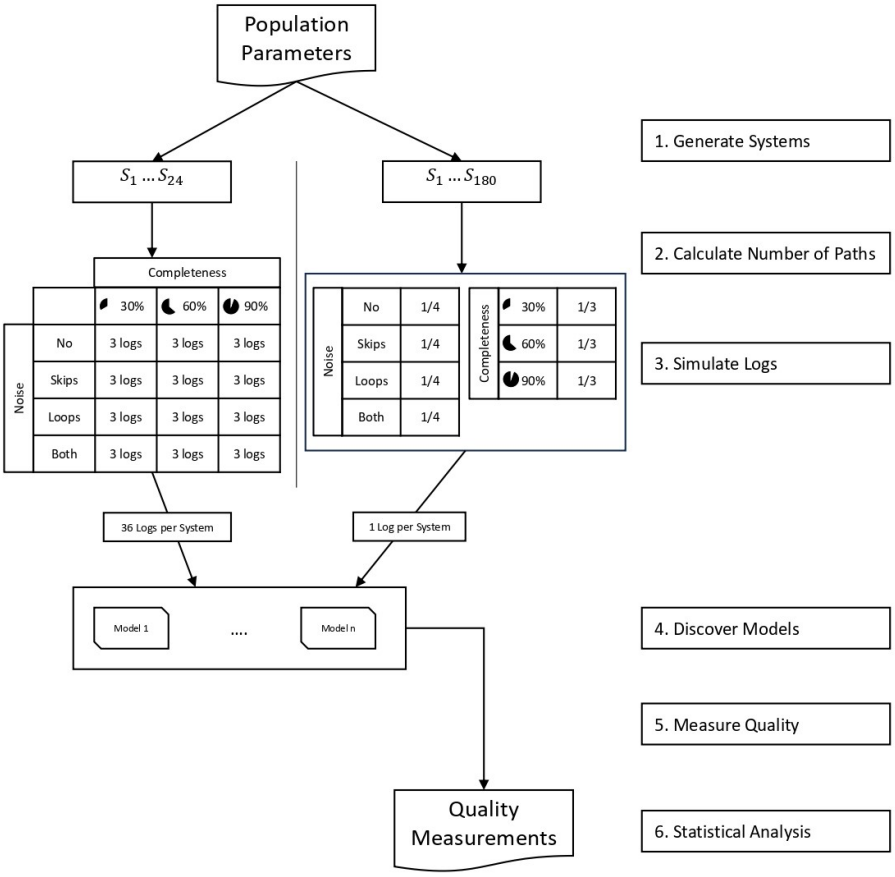

Fig. 1: Diagram representing the methodology.

Table 3: Levels of complexity.

Parameter	Explanation
Maximal Depth	The maximal number of generated nested patterns
Maximal number of AND branches	When an AND-pattern is generated, this is the maximal number of outgoing branches
Maximal number of XOR branches	When an XOR-pattern is generated, this is the maximal number of outgoing branches
Single activity weight	The weight of a single activity pattern being generated
Sequences weight	The weight of a sequence pattern being generated
AND split weight	The weight of an AND split pattern being generated
XOR split weight	The weight of an XOR split pattern being generated
Skip weight	The weight of a pattern generation being skipped
Loop weight	The weight of a loop pattern being generated

Table 4: Levels of complexity.

Complexity	Maximal depth	Maximal number of AND branches	Maximal number of XOR branches
Low complexity	2	3	4
Medium complexity	4	3	4
High complexity	5	3	4

for the configurations are described in Table 5. PLG2 uses a system of weights for each pattern to determine the likelihoods of that pattern being generated, proportionate to other patterns. These weights are values between 0 and 1. For example: if a loop pattern has a weight of 0.5, while an AND pattern has a weight of 1, the AND pattern will be twice as likely to be generated. To achieve the percentages shown in Table 5, all weights are summed for a total weight, then each individual weight is divided by the total weight.

Table 5: Weights of PLG2 Parameters.

Configurations	Single Activities	Sequence	AND Splits	XOR Splits	Skips	Loops
standard	8.33%	58.33%	8.33%	25.00%	0.00%	0.00%
with loops	6.67%	46.67%	6.67%	20.00%	0.00%	20.00%
with skips	6.67%	46.67%	6.67%	20.00%	20.00%	0.00%
with loops and skips	5.56%	38.89%	5.56%	16.67%	16.67%	16.67%

In order to answer RQ1, two systems are generated for each configuration. This will provide a sufficiently large number of systems to examine the effects of noise and incompleteness. For answering RQ2, 15 systems are generated for each configuration. Generating this many systems allows for sufficient variety in systems. This leads to 24 (3 complexity levels x 4 configurations x 2 systems) systems for RQ1 and 180 (3 complexity levels x 4 configurations x 15 systems) systems for RQ2 in total.

4.2 Calculating number of paths

Using pm4py, the ground truth models generated in Section4.1 are converted to process trees. The ground truth models are block-structured. These process trees are then used to calculate the number of possible execution paths. For loop operators, it is assumed they cannot be iterated over more than three times. This ensures a finite number of possible execution paths [28]. The number of paths is used as the input for a coupon collector problem [26]. This will be discussed in the next section.

4.3 Simulating logs

For this Section, PLG2 is used to generate logs from systems. Various types of completeness and noise are used for the logs. The completeness of a log is the percentage of unique paths present in the log compared to the number of unique paths the system allows. PLG2 is used to generate a set number of cases from a log. To calculate the number of cases needed to achieve a certain level of completeness, a coupon collector problem is used [26]. Consider the following variables:

- t = the total number of paths for a process [28]
- i = the i-th new path to be observed if $i-1$ unique paths have already been seen
- $E(C_i)$ = the expected number of cases needed in a log to achieve i number of unique paths
- $E(c_i)$ = the expected additional number of cases needed to find the i-th newest path. (meaning $C_i = C_{i-1} + c_i$)

If each path has an equal chance to be selected as a case for the log, the probability of a new path i to be discovered can be seen in Equation 2. The expected number of additional traces can be achieved by using Equation 3. To achieve a log with i unique paths, Equation 4 can be applied.

$$P(i) = \frac{t-i+1}{t} \tag{2}$$

$$E(c_i) = \frac{t}{t-i+1} \tag{3}$$

$$E(C_i) = E(c_1) + E(c_2) + ... + E(c_i) \tag{4}$$

For a system with 100 unique paths, achieving a log with 30% completeness requires 30 unique paths present in the log. To calculate the expected number of cases required in the log, $E(C_{50})$ is calculated using Equation 3 and Equation 4. This gives the result in Equation 5. This result is rounded up to achieve 69 required cases.

$$\begin{aligned} E(C_i) &= E(c_1) + E(c_2) + ... + E(c_{30}) \\ &= \frac{100}{100-1+1} + \frac{100}{100-2+1} + ... + \frac{100}{100-50+1} \\ &= \frac{100}{100} + \frac{100}{99} + ... + \frac{100}{51} \\ &= 68.82 \end{aligned} \tag{5}$$

PLG2 also provides the option to introduce various types of noise to a log. The different types of noise that are introduced are the same as the ones mentioned in [42]: 1) Activity name noise, 2) Missing event noise, and 3) Double events noise (See also Table 6). This noise is applied on the activity level. To calculate noise on a trace level, Equation 6 is used, where:

- n_{noise} represents: trace level noise. This means that a trace has an n_noise chance to contain any noise.
- S represents: the number of activities in a trace.
- a_{noise} represents: the activity level noise given to PLG2. Meaning there is an a_{noise} chance of an activity being affected by noise.

According to this formula, the likelihood of every activity in a trace being noise free is calculated. This can then be rewritten to Equation 7. The required trace level noise ($n_{noise} = 0.1$) can then be used to determine a_{noise}. PLG2 does not allow for the length of a trace to be taken into account when determining noise. As a result, the number of activities a system has is taken as the S value.

$$n_{noise} = 1 - (1 - a_{noise})^S \tag{6}$$

$$a_{noise} = 1 - \sqrt[S]{1 - n_{noise}} \tag{7}$$

Research Question 1 When generating logs from systems used to answer RQ1, different combinations of noise and incompleteness are applied to the logs. For every combination, visible in Table 6, 3 logs are generated. This is done for each of the 24 systems. This generates a total of 864 (24 systems x 36 configurations) logs.

Research Question 2 For the systems that are used to answer RQ2, a random level of incompleteness and noise is designated. This can range between the values visible in table 6 (except 100% completeness). Every system will only generate one log, this results in 180 logs (180 systems x 1 configuration).

Table 6: RQ1: number of logs.

	Completeness		
Noise Type	30%	60%	90%
No noise	3 logs	3 logs	3 logs
Activity names	3 logs	3 logs	3 logs
Missing events	3 logs	3 logs	3 logs
Double events	3 logs	3 logs	3 logs

In addition, for each system in RQ1 and RQ2, one log with 100% completeness (a system log) is generated. This log is used in step 5 of the methodology to measure the alignment based metrics. This gives an additional 204 (24 systems + 180 systems) system logs.

4.4 Discovering models

Based on the literature presented in Section 2, a selection of 3 mining algorithms is made. The selected algorithms selected are: bupaRminer, inductive miner, split miner. These miners currently stand out through their performance or novelty [13, 47]. Their respective implementations are included:

- bupaRminer implemented in R [47]

– Inductive miner implemented in Python (using pm4py) [6]
– Split miner implemented in java [14]

The eST miner does not have a publicly available code implementation, and the Probabilistic Inductive miner is not open source. As a result, these algorithms are not selected despite their performance and novelty.

In order to account for differences in performance caused by parameters not being optimised, the Inductive miner will be applied to discover two models for every log. The parameter used by the Inductive miner is the noise threshold. The noise threshold refers to the amount of noise filtered out of logs when discovering a model. 0.1 and 0.2 are the values chosen for the two models the Inductive miner discovers.

For Split miner 2.0, the concurrency threshold is 0.05. The concurrency threshold is defined as the minimum percentage of times that the two activities' life-cycles are required to overlap to assume the two activities concurrent. Since the PLG2 generated logs do not have life-cycles, this variable should not be relevant. For every generated log, four models are thus discovered. This results in a total of 4176 discovered models.

4.5 Measuring the quality of the discovered models

As stated previously, for checking conformance the following three quality dimensions are used: Fitness, Precision, and Simplicity.

To measure the fitness and precision quality dimension, the alignment-based fitness (F_a) and alignment-based precision (P_a) metrics are calculated for each discovered model as implemented in pm4py. The harmonic mean (F_1 score) is also calculated using the precision and fitness metric. The logs used for alignment is the system log. These metrics are seen as state-of-the-art metrics to measure their respective quality dimension. To measure simplicity, a combination of metrics will be used, which include: Control flow complexity (CFC), Model Size, Average Degree, Average Connector Degree, and Coefficient of Connectivity (CNC). For further explanation of these metrics, refer back to Section 2.1.

4.6 Statistical analysis

The analysis is divided for each Research question. For both Research questions the first step is to analyse the generated ground truth models. For each level of complexity, the distribution of several characteristics are discussed. These include the number of activities, paths and types of patterns generated. The second step is to analyse the simulated logs. The following metrics are used to evaluate the logs as described in [2]:

– Magnitude: the number of events
– Support: the number of cases
– Variety: the number or activities

- Level of Detail: the mean variety of all cases
- Structure: the inverse relative amount of directly-follows relations in the log, compared to the maximal number of directly-follows relations possible

The last step is to compare the quality dimensions of the discovered models. For RQ1, the focus is on analysing the metrics based on the types of noise and the levels of completeness. For RQ2, the performance of the miners is compared based on the complexity of the systems, and the types of patterns generated within those systems.

5 Results

5.1 Research Question 1

Ground truth models While generating ground truth models, three levels of complexity were envisioned. PLG2 randomly generates systems based on given parameters. Depending on the types of patterns generated, the number of possible paths for a system can still vary drastically. Any system could, for example, only generate sequence patterns. This results in it having a single path. This would not be considered a complex system. As a result, actual complexity of the systems is not determined by their input parameters, but instead based on the number of paths a system has. The eight systems with the highest number of paths are defined as high complexity systems, the eight systems with the lowest number of paths are considered as low complexity, and the remaining eight are medium complexity. An overview of the different characteristics of the systems can be found in Table 7. Unsurprisingly, the low complexity systems tend to score lower on all characteristics. Medium complexity systems have a higher average and maximum number of activities. There is also a higher average number of XOR patterns. However, there are also less loop patterns on average. Loop patterns increase the number of paths that a system has more than XOR patterns.

Simulated logs For RQ1 a total of 864 normal logs and 24 system logs were simulated. The logs are described in Table 8. All levels of complexity show a very wide range of magnitude and support when comparing the minimum and maximum values. There also appears to be some overlap for these values between complexity levels. For the interquantile ranges the values are quite wide, yet there is no overlap. Variety of the log is closely linked to the number of activities of the original system. The slight deviation between these values is likely caused by the noise and incompleteness that is introduced. Level of detail and Structure both increase significantly as logs become more complex.

Discovered models The Inductive miner was able to discover sound models for every log and every level of complexity. BupaRminer discovered no unsound models, but was unable to discover any model for 4 of the low complexity logs.

Table 7: RQ1: System characteristics.

Complexity	Characteristic	Minimum	Average	Maximum	sd
Low Complexity	Paths	3	6	13	3.02
	Activities	6	9	13	2.27
	XOR patterns	0	1.38	3	0.916
	AND patterns	0	0.25	1	0.463
	Loop patterns	0	0.25	1	0.463
Medium Complexity	Paths	28	186	540	198
	Activities	11	40.4	111	30.3
	XOR patterns	0	5.75	19	5.65
	AND patterns	0	1.38	6	2.07
	Loop patterns	0	0.625	3	1.19
High Complexity	Paths	702	1660	3312	858
	Activities	30	46	62	13.1
	XOR patterns	0	5.12	11	3.56
	AND patterns	1	1.38	4	1.06
	Loop patterns	0	2.75	6	2.25

Table 8: RQ1: Log characteristics.

Complexity	Characteristic	Minimum	Q1	Average	Median	Q3	Maximum	sd
Low Complexity	Magnitude	5.0	16.0	45.9	30.0	52.2	273	48.2
	Support	1.0	3.0	7.62	5.5	12.0	29.0	6.21
	Variety	3.0	6.0	7.79	8.0	9.0	13.0	1.86
	Level of Detail	3.0	4.49	5.62	5.0	6.4	15.4	1.64
	Structure	0.667	0.917	0.941	0.953	0.963	0.982	0.036
Medium Complexity	Magnitude	143	636.0	4351.0	1727.0	5520.0	27288.0	6277
	Support	11.0	45.2	222.0	104.0	223.0	1239.0	308.0
	Variety	11.0	27.8	40.3	31.0	39.8	112.0	28.3
	Level of Detail	11.0	13.2	18.1	15.2	20.9	35.4	7.37
	Structure	0.975	0.996	0.995	0.997	0.998	1.000	0.007
High Complexity	Magnitude	5455	18166.0	55138.0	35304.0	69461.0	225147.0	54045.0
	Support	251.0	670.0	1978.0	1424.0	2792.0	7624.0	1789.0
	Variety	30.0	36.5	46.2	48.0	58.0	65.0	12.3
	Level of Detail	15.5	22.8	27.7	28.2	29.7	46.6	8.50
	Structure	0.997	0.998	0.998	0.999	0.999	0.999	0.001

The cause was the inability to detect a loop block. Split miner was able to discover models for all logs, 7 of which were unsound. Those models were not taken into account when calculating the fitness and precision metrics.

Table 9 shows the distribution of the metrics used to measure fitness, precision and simplicity for the different miners. The fitness, precision, and F_1 scores are also visualised per miner in Figure 2, 3, and 4 respectively. Table 10 describes the average effect of different types of noise on fitness, precision, and the F_1 score. Figure 5, 6, and 7 show boxplot distributions of the fitness, precision, and F_1 score per noise type. Table 11 shows the effect of completeness on the averge precision, fitness, and F_1 score metrics. Figures 8, 9, and 10 show boxplot distributions of the fitness, precision, and F_1 score respectively per level of completeness.

The Inductive miner on average performed best on both precision and fitness metrics for all incompleteness levels and all noise types. The boxplots show great consistency in this regard also. The Split miner greatly outperforms all other algorithms on the basis of simplicity metrics. The BupaRminer usually sits in between the other miners, only performing the worst on CNC and CFC metrics. The difference between the performances of the two Inductive miners is extremely minor. For all miners, there was a great significant impact of completeness on fitness. The distribution of fitness becomes tighter as completeness increases. The only significant impact of noise was the effect of Activity name noise on precision. Figure 6 shows a wider distribution for the precision metric when Activity name noise is introduced except when Split miner is used.

5.2 Research Question 2

Ground truth models Three levels of complexity are once again defined according to the number of paths each system has. For RQ2, 180 systems were generated. As a result, each complexity level contains 60 systems instead of eight. An overview of the characteristics per complexity level can be seen in Table 12. The value of all characteristics increase as the level of complexity rises. The only remarkable aspect is the maximum number of activities for medium complexity is higher than high complexity. This was also the case for RQ1.

Simulated logs Each system was used to generate one log with a random level of completeness and a random type of noise. This selection was done randomly and resulted in the distributions seen in Table 13 and 14. The level of completeness is slightly skewed towards 30% completeness. Activity name noise is also less present when compared to other types of noise within the logs.

The same log metrics from [2] are once again examined in Table 15. Magnitude and Support increase significantly as the level of complexity increases. Similarly, Variety also increases with complexity although for the lower two complexity levels never reaches the same number of activities as seen in their respective systems (See also Table 12). The level of detail also still increases as with the other logs, but the difference is not as large as before the same can

Table 9: RQ1: Results.

Metric	Miner	Q1	Average	Median	Q3	sd
F_a	BM	0.952	0.963	**1.000**	1.000	0.068
	IM0.2	1.000	**0.978**	**1.000**	1.000	0.063
	IM0.1	1.000	**0.978**	**1.000**	1.000	0.063
	SM	0.921	0.943	0.981	1.000	0.077
P_a	BM	0.881	0.925	0.947	0.998	0.091
	IM0.2	0.978	**0.977**	**0.990**	1.000	0.046
	IM0.1	0.978	**0.977**	**0.990**	1.000	0.046
	SM	0.701	0.727	0.725	0.757	0.043
F_1 score	BM	0.932	0.940	0.956	0.975	0.066
	IM0.2	0.978	**0.975**	**0.991**	0.996	0.044
	IM0.1	0.978	**0.975**	**0.991**	0.996	0.044
	SM	0.813	0.818	0.824	0.837	0.034
CFC	BM	5.00	55.6	17.0	26.0	272
	IM0.2	4.00	14.5	13.0	18.0	13.3
	IM0.1	4.00	14.6	13.0	18.0	13.4
	SM	4.00	**13.1**	**12.0**	18.0	12.5
Size	BM	16.0	49.8	49.0	68.0	37.5
	IM0.2	16.0	47.3	44.0	62.0	37.3
	IM0.1	16.0	47.3	44.0	62.0	37.4
	SM	14.0	**44.9**	**43.0**	62.0	34.9
Average Degree	BM	2.27	2.32	2.35	2.41	0.160
	IM0.2	2.24	2.27	2.30	2.36	0.142
	IM0.1	2.24	2.27	2.30	2.36	0.142
	SM	2.14	**2.22**	**2.29**	2.33	0.181
Average Connector Degree	BM	3.00	3.31	3.29	3.50	0.283
	IM0.2	3.00	3.30	3.33	3.43	0.259
	IM0.1	3.00	3.30	3.33	3.43	0.259
	SM	2.60	**2.88**	**3.05**	3.31	0.658
CNC	BM	1.13	1.16	1.18	1.20	0.080
	IM0.2	1.12	1.14	**1.15**	1.18	0.071
	IM0.1	1.12	1.14	**1.15**	1.18	0.071
	SM	1.07	**1.11**	**1.15**	1.17	0.091

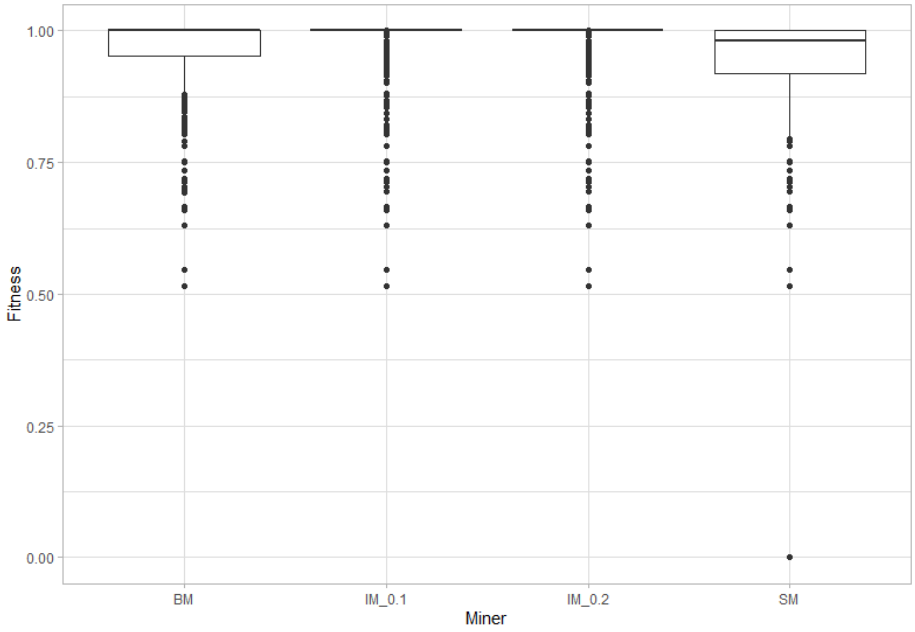

Fig. 2: Boxplots of Alignment Fitness for each miner.

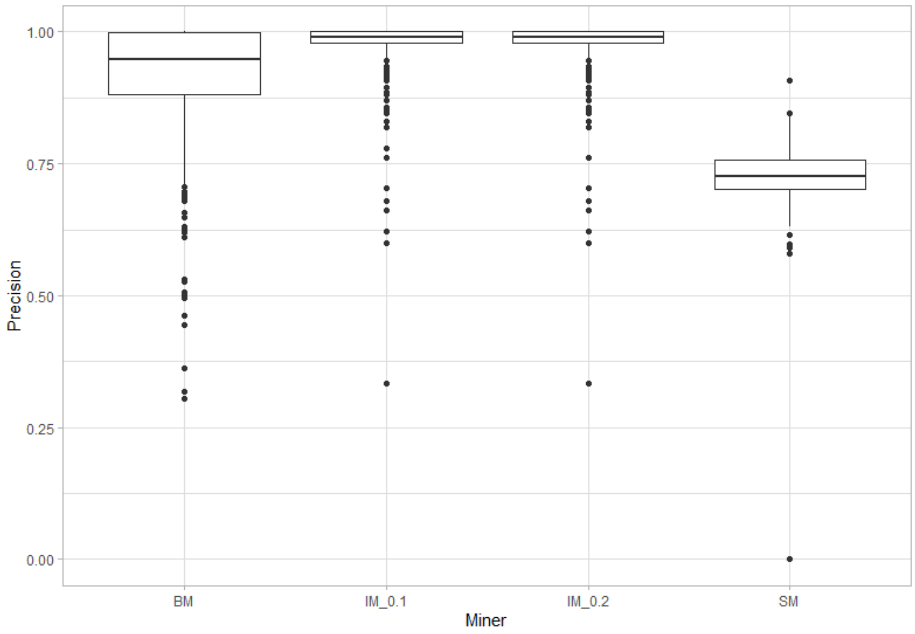

Fig. 3: Boxplots of Alignment Precision for each miner.

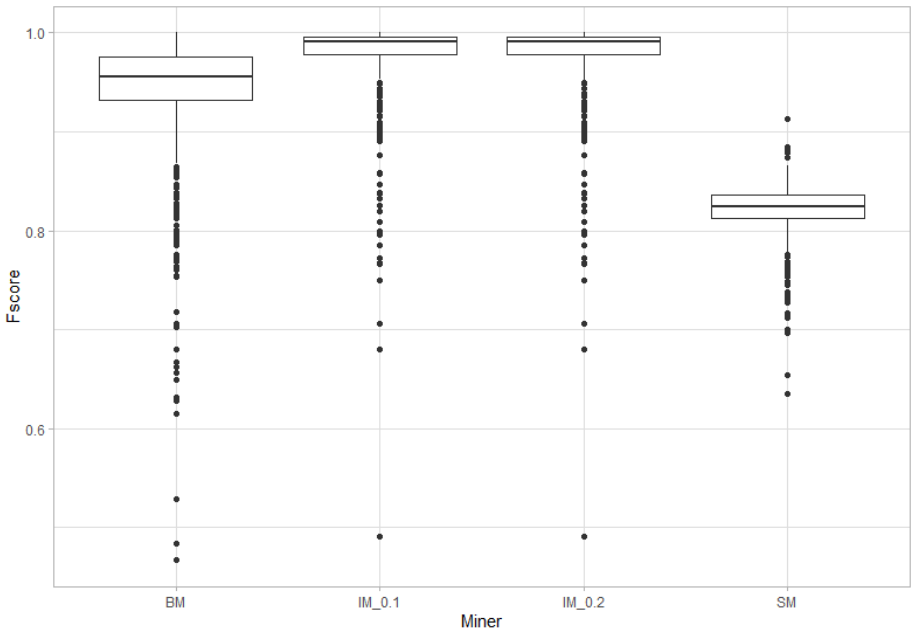

Fig. 4: Boxplots of Fscore for each miner.

Table 10: RQ1: impact of noise.

Metric	Miner	No noise	Activity name noise	Missing activity noise	Double activity noise
F_a	BM	0.965	0.960	0.960	0.969
	IM0.2	**0.979**	**0.976**	**0.973**	**0.982**
	IM0.1	**0.979**	**0.976**	**0.973**	**0.982**
	SM	0.940	0.944	0.942	0.946
P_a	BM	0.934	0.903	0.934	0.930
	IM0.2	**0.983**	**0.962**	**0.983**	**0.981**
	IM0.1	**0.983**	0.961	**0.983**	**0.981**
	SM	0.727	0.725	0.727	0.730
F_1 score	BM	0.946	0.924	0.943	0.946
	IM0.2	**0.980**	**0.966**	**0.976**	**0.981**
	IM0.1	**0.980**	0.965	**0.976**	**0.981**
	SM	0.817	0.817	0.817	0.821

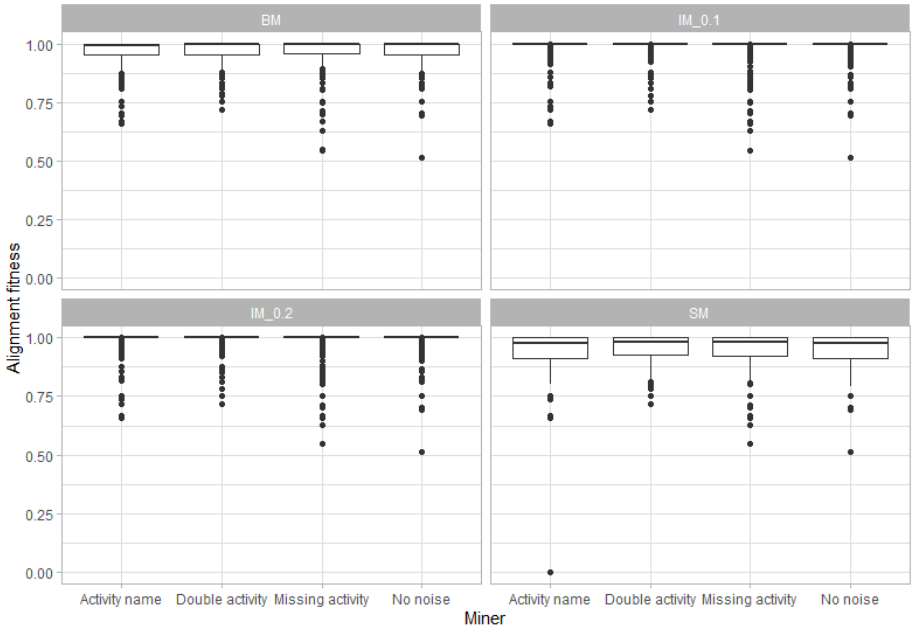

Fig. 5: Boxplots of the effect of noise on fitness.

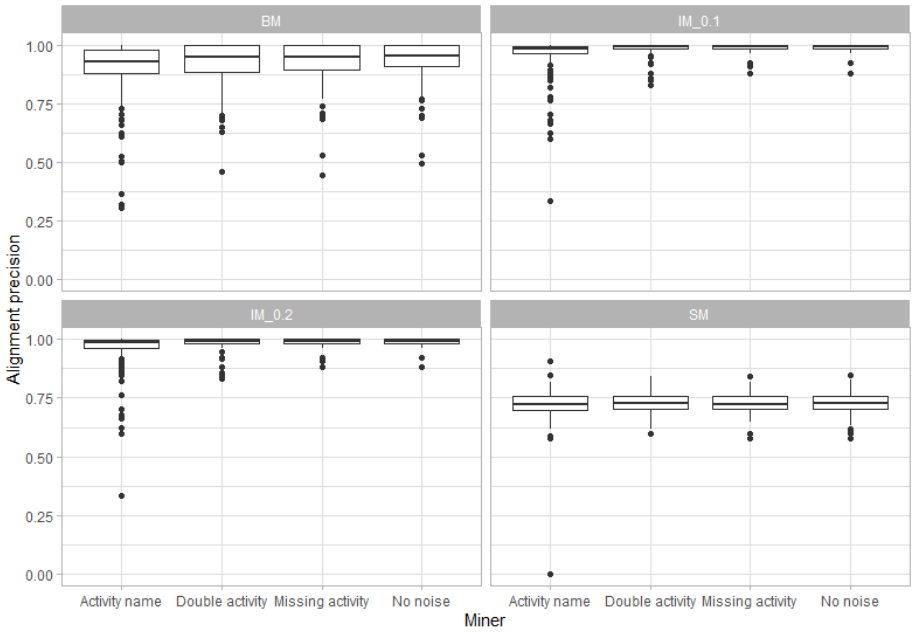

Fig. 6: Boxplots of the effect of noise on precision.

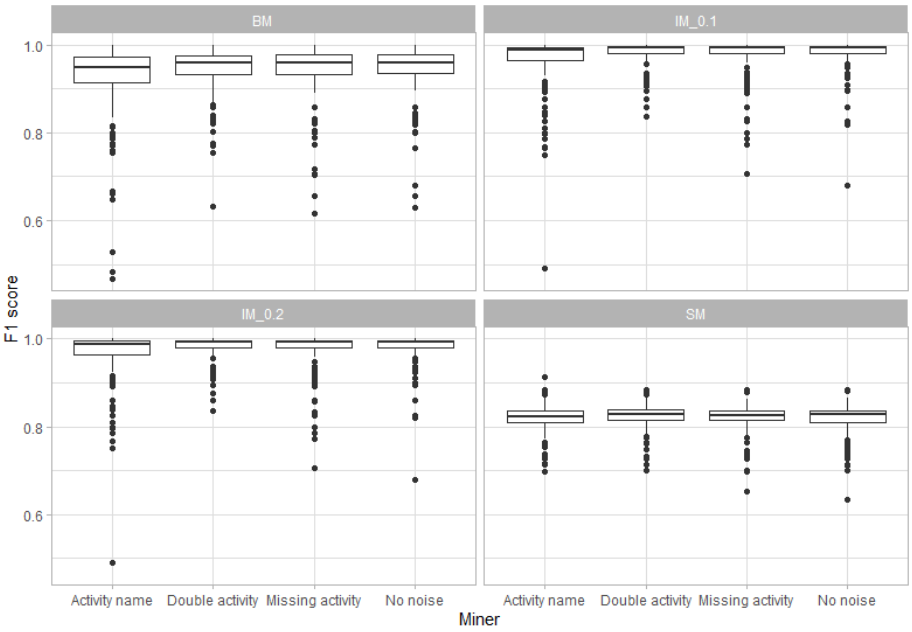

Fig. 7: Boxplots of the effect of noise on the F_1 score.

Table 11: RQ1: impact of incompleteness.

Metric	Miner	30%	60%	90%
F_a	BM	0.943	0.963	0.984
	IM0.2	**0.957**	**0.979**	**0.997**
	IM0.1	**0.957**	**0.979**	**0.997**
	SM	0.934	0.938	0.957
P_a	BM	0.932	0.927	0.916
	IM0.2	**0.982**	**0.978**	**0.971**
	IM0.1	**0.982**	**0.978**	**0.971**
	SM	0.733	0.726	0.723
F_1 score	BM	0.933	0.941	0.946
	IM0.2	**0.967**	**0.977**	**0.982**
	IM0.1	**0.967**	**0.977**	**0.982**
	SM	0.818	0.815	0.822

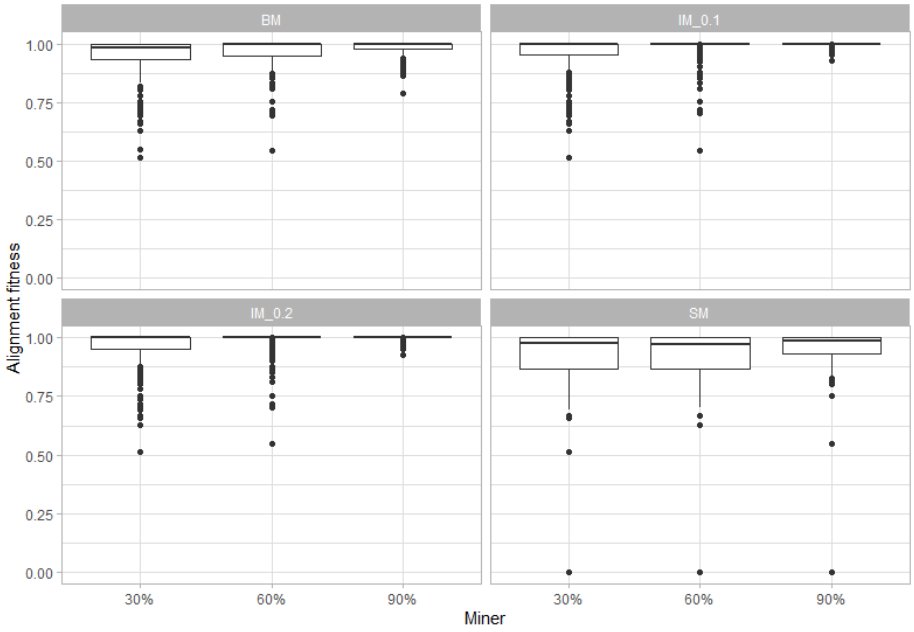

Fig. 8: Boxplots of the effect of completeness on fitness.

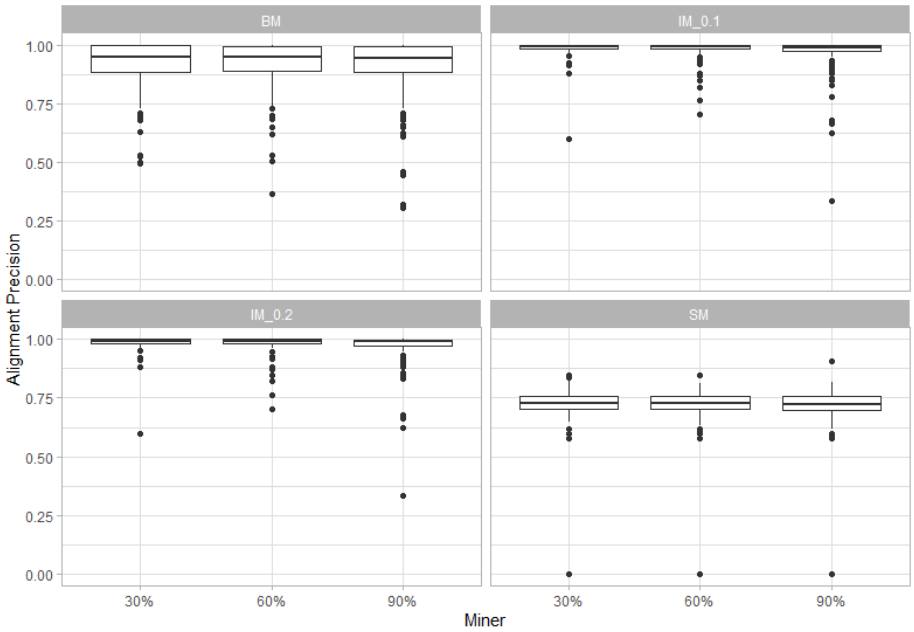

Fig. 9: Boxplots of the effect of completeness on precision.

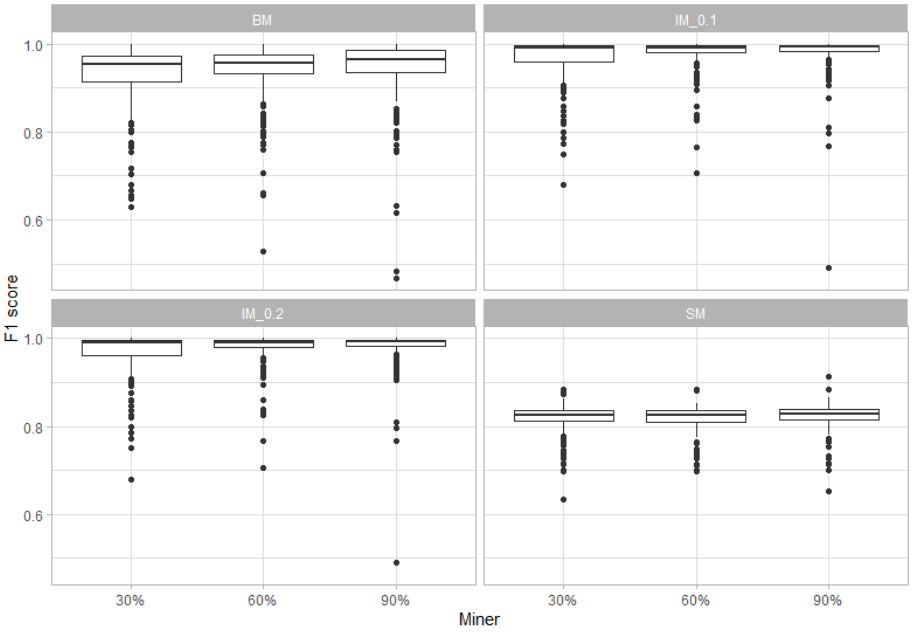

Fig. 10: Boxplots of the effect of completeness on the F_1 score.

Table 12: RQ2: System Characteristics.

Complexity	Characteristic	Minimum	Average	Maximum	sd
Low Complexity	Paths	1	3.65	11	2.35
	Activities	2	7.8	32	4.79
	XOR patterns	0	0.883	5	1.01
	AND patterns	0	0.167	1	0.418
	Loop patterns	0	0.317	2	0.537
Medium Complexity	Paths	12	51	135	34.5
	Activities	8	31.2	113	17.0
	XOR patterns	0	4.43	22	3.50
	AND patterns	0	1.05	3	0.946
	Loop patterns	0	0.817	5	1.30
High Complexity	Paths	144	1206	4381	1278
	Activities	14	47.2	100	22.0
	XOR patterns	1	6.37	17	3.61
	AND patterns	0	1.63	6	1.28
	Loop patterns	0	1.80	9	2.13

be said for the standard deviation. Structure has a very low minimum for low complexity, but this difference is not observed in higher complexity levels.

Table 13: Distribution of completeness over the different logs

Completeness	Number of Logs
30%	69
60%	53
90%	58

Table 14: Distribution of noise over the different logs

Noise type	Number of Logs
No noise	48
Activity names	37
Missing events	46
Double events	49

Discovered models The Inductive miner and Split miner were able to discover a sound model for each of the 180 systems. BupaRminer was not able to discover a model for 5 systems. The issue was once again caused by not being able to detect loop blocks. As complexity rises, fitness increases while precision decreases. As a result, the F_1 score remains consistent for the different levels of complexity. As can be seen in the correlation graphs in Figures 11, 12, 13, and 14, There is very likely a correlation between complexity (the number of paths) and precision. The correlation between number of paths and fitness is below 0.2 for all correlation graphs. The correlation between the two values is insignificant.

Generally the Inductive miner outperforms both other miners on fitness and precision. It also has less variety in performance for those metrics. For all simplicity metrics, the Split miner outperforms the other miners. The only competitor is the Inductive miner at higher complexity levels on account for having a lower standard deviation.

There is very little difference as a result of the parameter change of the Inductive miner.

Two types of correlations were examined. The first is the Pearson method, which examines if there are linear correlations between variables. No significant linear correlation could be detected. The second technique is the Spearman method. It examines non-linear correlations bewteen variables. As can be seen in Figures 11, 12, 13, and 14, there is still almost no significant correlation between most of the metrics when compared to fitness and precision. For

Table 15: RQ2: Log characteristics.

Complexity	Characteristic	Minimum	Q1	Average	Median	Q3	Maximum	sq
Low Complexity	Magnitude	2	5	25.4	16	31.5	118	27.2
	Support	1	1	4.23	3	6	19	4.24
	Variety	2	4.75	6.98	6	8	19	3.62
	Level of Detail	2	4	5.74	5	6	23	3.10
	Structure	0.25	0.863	0.886	0.917	0.953	0.992	0.114
Medium Complexity	Magnitude	37	204	861	500	1061	5304	981
	Support	5	18	56	37.5	73.5	312	63.4
	Variety	8	21	28	27	33.5	89	14.4
	Level of Detail	6.17	9.60	14.2	13.7	18.3	25.7	4.95
	Structure	0.953	0.994	0.991	0.996	0.997	1.000	0.013
High Complexity	Magnitude	911	3049	30243	15138	40543	231334	43962
	Support	53	161	1427	796	1519	10085	2025
	Variety	14	28.8	47.2	47	61.2	100	22
	Level of Detail	7.22	16.8	21.8	21.4	26.7	35.5	6.57
	Structure	0.985	0.996	0.997	0.999	0.999	1.000	0.003

BupaRminer there is a negative correlation between precision and the number of Loop-patterns in a system. Inductive miner also has a correlation between precision and the number of Loop-patterns, but to a lesser degree.

Table 16: RQ2: low complexity results.

Metric	Miner	Q1	Average	Median	Q3	sd
F_a	BM	0.932	0.941	0.971	1.000	0.106
	IM0.2	0.939	**0.943**	**0.980**	1.000	0.107
	IM0.1	0.939	**0.943**	**0.980**	1.000	0.107
	SM	0.900	0.924	0.958	1.000	0.108
P_a	BM	0.923	0.943	**1.000**	1.000	0.140
	IM0.2	0.964	**0.962**	**1.000**	1.000	0.137
	IM0.1	0.964	**0.962**	**1.000**	1.000	0.137
	SM	0.700	0.716	0.723	0.750	0.101
F_1 score	BM	0.920	0.934	0.967	1.000	0.137
	IM0.2	0.950	**0.945**	**0.980**	0.992	0.137
	IM0.1	0.950	**0.945**	**0.980**	0.992	0.137
	SM	0.805	0.800	0.823	0.829	0.111
CFC	BM	0	2.52	2	4	2.72
	IM0.2	0	2.05	2	3.25	2.06
	IM0.1	0	2.05	2	3.25	2.06
	SM	0	**1.62**	**0**	3	1.94
Size	BM	7	11.5	10	16	5.48
	IM0.2	7	11.0	10	14	5.17
	IM0.1	7	11.0	10	14	5.17
	SM	6.75	**10.4**	**8.5**	14	4.97
Average Degree	BM	1.71	1.98	2	2.17	0.253
	IM0.2	1.71	1.95	2	2.16	0.236
	IM0.1	1.71	1.95	2	2.16	0.236
	SM	1.70	**1.89**	**1.78**	2.12	0.228
Average Connector Degree	BM	3	3.22	3	3.5	0.341
	IM0.2	3	3.22	3	3.5	0.349
	IM0.1	3	3.22	3	3.5	0.349
	SM	1	**1.81**	**1**	2.6	0.874
CNC	BM	0.857	0.988	1.000	1.08	0.127
	IM0.2	0.857	0.975	1.000	1.08	0.118
	IM0.1	0.857	0.975	1.000	1.08	0.118
	SM	0.851	**0.946**	**0.888**	1.06	0.114

Table 17: RQ2: medium complexity results.

Metric	Miner	Q1	Average	Median	Q3	sd
F_a	BM	0.950	0.948	**1.000**	1.000	0.085
	IM0.2	0.999	**0.971**	**1.000**	1.000	0.070
	IM0.1	0.999	**0.971**	**1.000**	1.000	0.070
	SM	0.895	0.878	0.963	0.990	0.247
P_a	BM	0.826	0.885	0.939	0.977	0.132
	IM0.2	0.950	**0.940**	**0.971**	0.988	0.108
	IM0.1	0.950	**0.940**	**0.971**	0.988	0.108
	SM	0.688	0.673	0.712	0.744	0.184
F_1 score	BM	0.857	0.906	0.934	0.974	0.084
	IM0.2	0.947	**0.949**	**0.979**	0.987	0.077
	IM0.1	0.947	**0.949**	**0.979**	0.987	0.077
	SM	0.809	0.814	0.818	0.827	0.027
CFC	BM	8	15.1	13	19	8.60
	IM0.2	7	12.0	11	15	6.40
	IM0.1	7	12.0	11	15.5	6.42
	SM	6	**11.2**	**10**	14.2	6.30
Size	BM	33	43.9	40	51	20.8
	IM0.2	32.5	42.2	39	49.5	19.2
	IM0.1	32.5	42.2	39	49.5	19.2
	SM	29.8	**40.4**	**38**	47.5	19.0
Average Degree	BM	2.27	2.33	2.33	2.4	0.105
	IM0.2	2.24	2.29	2.30	2.35	0.088
	IM0.1	2.24	2.29	2.30	2.36	0.088
	SM	2.19	**2.25**	**2.25**	2.32	0.093
Average Connector Degree	BM	3.17	3.33	3.29	3.5	0.253
	IM0.2	3	3.26	3.29	3.4	0.226
	IM0.1	3	3.26	3.27	3.4	0.226
	SM	2.82	**3.03**	**3**	3.22	0.265
CNC	BM	1.14	1.17	1.17	1.20	0.052
	IM0.2	1.12	1.14	1.15	1.18	0.044
	IM0.1	1.12	1.14	1.15	1.18	0.044
	SM	1.10	**1.12**	**1.12**	1.16	0.047

Table 18: RQ2: high complexity results.

Metric	Miner	Q1	Average	Median	Q3	sd
F_a	BM	0.978	0.977	**1.000**	1.000	0.044
	IM0.2	1.000	**1.000**	**1.000**	1.000	0.001
	IM0.1	1.000	**1.000**	**1.000**	1.000	0.000
	SM	0.960	0.934	0.986	1.000	0.185
P_a	BM	0.835	0.842	0.892	0.931	0.152
	IM0.2	0.924	**0.931**	**0.938**	0.958	0.045
	IM0.1	0.924	**0.931**	**0.938**	0.958	0.046
	SM	0.665	0.665	0.692	0.711	0.141
F_1 score	BM	0.894	0.895	0.926	0.957	0.114
	IM0.2	0.960	**0.964**	**0.968**	0.978	0.026
	IM0.1	0.960	0.963	**0.968**	0.978	0.026
	SM	0.796	0.801	0.810	0.824	0.060
CFC	BM	17	30.6	24	35.8	21.6
	IM0.2	14	22.4	20	28	12.1
	IM0.1	14	22.5	20	28	12.2
	SM	12.8	**20.2**	**17.5**	22	11.5
Size	BM	47.2	74.6	69.5	86.2	33.9
	IM0.2	43.8	71.0	68	88.2	32.7
	IM0.1	43.8	71.0	68	88.2	32.8
	SM	43	**67.1**	**63**	83	31.4
Average Degree	BM	2.36	2.40	2.41	2.44	0.077
	IM0.2	2.31	2.35	2.36	2.39	0.065
	IM0.1	2.31	2.35	2.36	2.39	0.065
	SM	2.27	**2.31**	**2.31**	2.36	0.069
Average Connector Degree	BM	3.17	3.29	3.29	3.37	0.177
	IM0.2	3.18	3.28	3.26	3.39	0.151
	IM0.1	3.18	3.28	3.26	3.39	0.151
	SM	3	**3.18**	**3.18**	3.31	0.197
CNC	BM	1.18	1.20	1.20	1.22	0.038
	IM0.2	1.15	1.18	1.18	1.20	0.033
	IM0.1	1.15	1.18	1.18	1.20	0.032
	SM	1.14	**1.16**	**1.16**	1.18	0.034

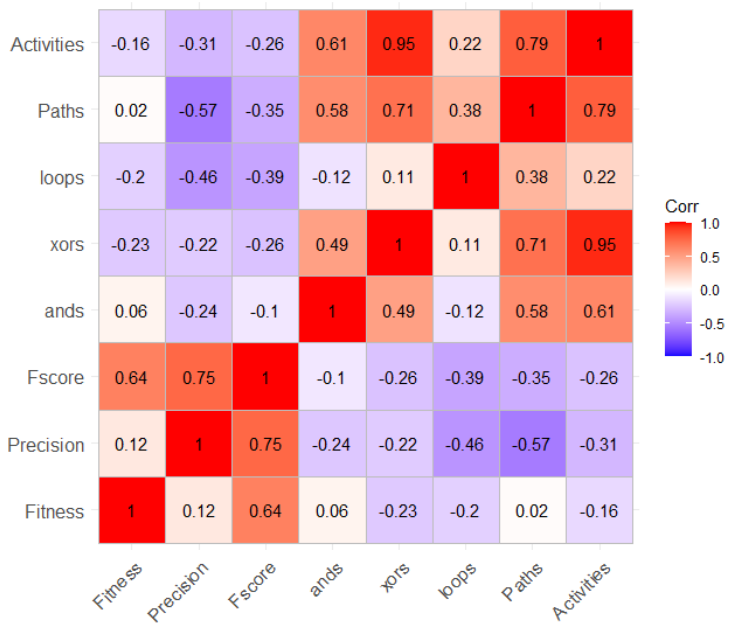

Fig. 11: Spearman correlation graph for BupaRminer.

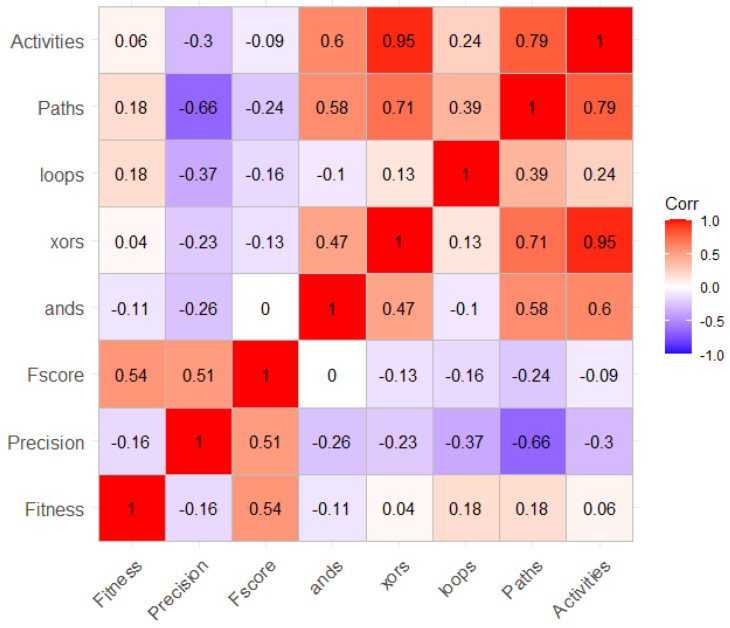

Fig. 12: Spearman correlation graph for Inductive miner (0.2 noise threshold).

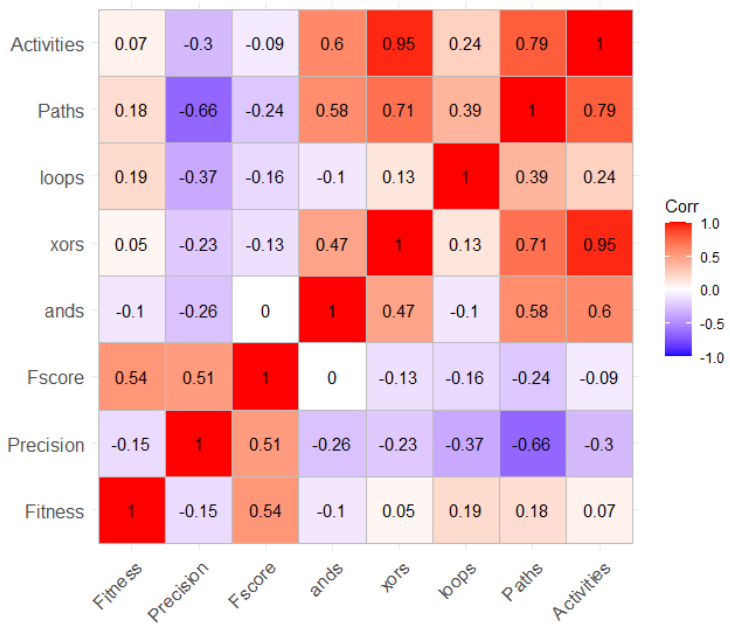

Fig. 13: Spearman correlation graph for Inductive miner (0.1 noise threshold).

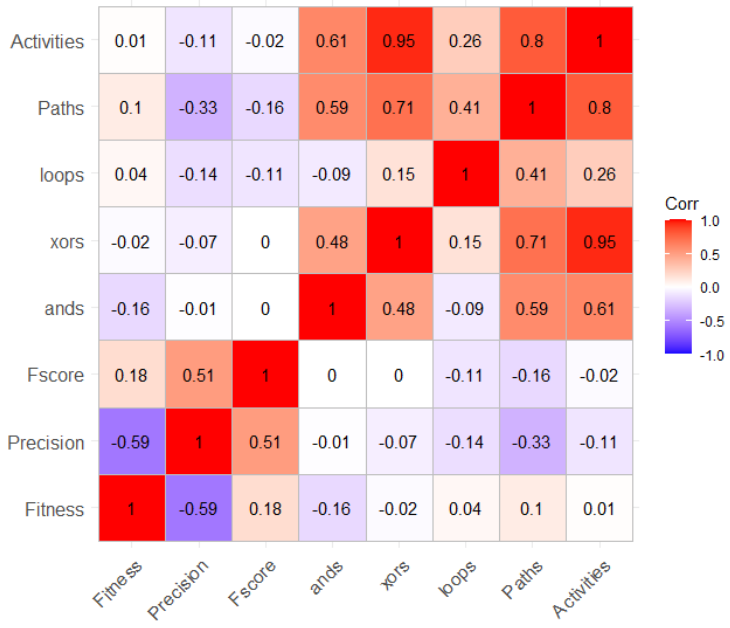

Fig. 14: Spearman correlation graph for Split miner.

6 Discussion

When observing the performance of the different miners, both versions of the Inductive miner clearly outperform both other miners in regard to fitness and precision. Changing the parameter of the Inductive miner only has a very minor negative impact on its performance. BuparRminer performs slightly worse than the Inductive miner on these measures. Split miner tends to have acceptable fitness scores, but falls behind significantly when comparing precision.

Split miner does excel in all simplicity metrics when compared to both other algorithms. Inductive miner and BupaRminer perform worse in this regard. As the level of complexity rises for systems, the difference between Inductive miner and BupaRminer becomes greater.

The results of RQ1, described in Table 9, show that the average fitness and precision of the Inductive miner are 0.978 an 0.977 respectively. The F_1 score is on average 0.975. When considering noise, shown in Table 10, the average rises to 0.979 for fitness and 0.983 for precision for noise free logs. Surprisingly, the presence of Double activity noise further increases the average fitness and precision to 0.982 and 0.981. The only form of noise that has a negative impact on the average precision is Activity name noise, this is also applicable for BupaRminer and Split miner. Completeness clearly has a positive impact on the fitness for all miners. When considering completeness of the logs, described in Table 11, for the Inductive miner, the fitness values are 0.957, 0.977, and 0.997 for 30%, 60%, and 90% respectively. The opposite is true for precision with the Inductive miner having a precision of 0.982, 0.978, and 0.971 for the respective completeness levels. Split miner is the only discovery algorithm for which the F_1 score reaches its lowest average at 60% completeness, with it being 0.815. Split miner performs best on all simplicity metrics. The Split miner only performs worse on its standard deviation of Average degree, Average connector degree, and CNC.

The results of RQ2 are described in Tables 16, 17, and 18. The positive effect of complexity on fitness is clearly visible. The inductive miner is once again the highest performing miner. It reaches an average fitness of 0.943 for low complexity, but increases to an average fitness of 1.000 at highest complexity. The opposite is true for precision. The Inductive miner still performs best, with its highest being 0.962 at low complexity. The lowest value is 0.931 for high complexity. Split miner still performs best on all simplicity metrics, but consistently scores worse on the standard deviation of Average degree, Average connector degree, and CNC. The correlation graphs from Figures 11, 12, 13, and 14 show that BupaRminer has the most negative correlations with system characteristics. The most negative correlation is -0.46 between the number of loop patterns and precision. Do note these correlations are not linear. Split miner has the least correlation with system characteristics. The most significant is also a negative correlation (-0.14) between the number of loop patterns and precision. This does suggest that as the number of loops in a system keeps increasing, Split miner might eventually perform better on precision than the other miners.

The biggest weakness of the BupaRminer is with discovering loop blocks when noise is introduced. It is the only miner that was not able to discover a model for several of the logs provided. The biggest weakness of the Split miner is the ability to generate unsound models and its lack of precision, especially for complex systems. The Inductive miner has only the minor weakness of discovering slightly more complex models, but is otherwise resistant to noisy, incomplete and complex systems.

7 Conclusion

This paper first gives an overview of existing quality dimensions, discovery algorithms, and existing comparative studies. From this overview, a selection is made for the types of miners and quality dimensions, with corresponding metrics, that are to be used. For the algorithms, the BupaRminer, Inductive miner and Split miner are chosen. Fitness, precision, F_1 score and simplicity are chosen as the quality dimensions. Two research questions are defined: RQ1: which miners perform best when exposed to different levels of noise and completeness, and RQ2: which miners perform best when exposed to data from systems with differing levels of complexity. An experiment with two focus points is conducted. First, systems are generated for both focus points of the experiment. Second, logs are simulated from the systems. Third, new models are discovered from the logs. Lastly, models are evaluated on the basis of the previously chosen quality dimensions.

From the analysis, it can be concluded that the Inductive miner performs best on fitness, precision, and F_1 score regardless of system complexity, noise in the log, or completeness of the log. The Split miner performs best on all Simplicity metrics regardless of system complexity, noise in the log, or completeness of the log.

The experiment was limited by several factors. Due to time constraints, the amount of systems and logs that could be generated was limited. As the number of systems and logs grow, so does the number of models that have to be discovered. Discovering more models was not a possibility withing the given time frame. There was also no opportunity to optimise the parameter of the Inductive miner. This was mentioned as a shortcoming for other comparative studies and was not fixed in this paper. Computational power also placed a limit on the complexity of systems that could be generated. Both time and availability of code played a role in the selection of discovery algorithms. As mentioned, The ETM has run-times which are described as excessive. The eST miner and PiM do not have publicly available code implementations. Finally, the implementation of noise could not be properly determined based on average trace length. As a result, the formula given in the paper is not fully correctly implemented.

Future research could focus on using a wider selection of discovery algorithms. This could provide more specific cases in which certain miners perform better than others. The amount of data could be larger to verify the results found in this paper. In this paper, only the alignment based metrics for fitness and

precision were used. Future research could examine more metrics for each quality dimension.

References

1. Business Process Intelligence (BPI) (search),
2. Process mining in flexible environments. Technische Universiteit Eindhoven (2009). https://doi.org/10.6100/IR644335, https://research.tue.nl/en/publications/process-mining-in-flexible-environments(dc50e490-e9ce-4c20-bcfe-5c9e992daa54).html
3. van der Aalst, W.M.P.: Verification of Workflow nets. pp. 407–426 (Jan 1997)
4. van der Aalst, W.M.P.: Foundations of Process Discovery. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, pp. 37–75. Lecture Notes in Business Information Processing, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_2, https://doi.org/10.1007/978-3-031-08848-3_2
5. van der Aalst, W.M.P., Adriansyah, A., Dongen, B.: Replaying History on Process Models for Conformance Checking and Performance Analysis. WIREs Data Mining and Knowledge Discovery **2**, 182–192 (Mar 2012). https://doi.org/10.1002/widm.1045
6. van der Aalst, W.M.P., Berti, A.: Discovering Object-Centric Petri Nets (Oct 2020)
7. van der Aalst, W.M.P., Medeiros, A., Weijters, A.: Genetic Process Mining. vol. 14, pp. 48–69 (Jun 2005). https://doi.org/10.1007/11494744_5
8. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering **16**(9), 1128–1142 (Sep 2004). https://doi.org/10.1109/TKDE.2004.47, http://ieeexplore.ieee.org/document/1316839/
9. Adriansyah, A., van Dongen, B., van der Aalst, W.: Conformance Checking Using Cost-Based Fitness Analysis. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference. pp. 55–64 (Aug 2011). https://doi.org/10.1109/EDOC.2011.12, https://ieeexplore.ieee.org/document/6037560, iSSN: 1541-7719
10. Adriansyah, A., Munoz-Gama, J., Carmona, J., Dongen, B., Aalst, W.: Alignment Based Precision Checking. vol. 132 (Sep 2012). https://doi.org/10.1007/978-3-642-36285-9_15
11. Adriansyah, A., Munoz-Gama, J., Carmona, J., Dongen, B., Aalst, W.: Measuring precision of modeled behavior. Information Systems and e-Business Management **13** (Jan 2014). https://doi.org/10.1007/s10257-014-0234-7
12. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems **59** (May 2019). https://doi.org/10.1007/s10115-018-1214-x
13. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Transactions on Knowledge and Data Engineering **31**(4), 686–705 (Apr 2019). https://doi.org/10.1109/TKDE.2018.2841877, https://ieeexplore.ieee.org/document/8368306, conference Name: IEEE Transactions on Knowledge and Data Engineering
14. Augusto, A., Dumas, M., La Rosa, M.: Automated Discovery of Process Models with True Concurrency and Inclusive Choices (May 2021). https://doi.org/10.48550/arXiv.2105.06016, http://arxiv.org/abs/2105.06016, arXiv:2105.06016 [cs]

15. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining Based on Regions of Languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) Business Process Management, vol. 4714, pp. 375–383. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_27, http://link.springer.com/10.1007/978-3-540-75183-0_27, series Title: Lecture Notes in Computer Science
16. Brons, D., Scheepens, R., Fahland, D.: Striking a new Balance in Accuracy and Simplicity with the Probabilistic Inductive Miner. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 32–39 (Oct 2021). https://doi.org/10.1109/ICPM53251.2021.9576864, https://ieeexplore.ieee.org/abstract/document/9576864?casa_token = 1txmYQDvc2oAAAAA:DUV7PytQXep-So6r5mksJQs8MuOPLJGLd1 z0ciDq0eJ9Qx5ugOvlCY9v67Fd6sghrbwL1HveXn9Wlg
17. vanden Broucke, S.K.L.M., De Weerdt, J.: Fodina: A robust and flexible heuristic process discovery technique. Decision Support Systems **100**, 109–118 (Aug 2017). https://doi.org/10.1016/j.dss.2017.04.005, https://www.sciencedirect.com/science/article/pii/S0167923617300647
18. vanden Broucke, S.K.L.M., De Weerdt, J., Vanthienen, J., Baesens, B.: Determining Process Model Precision and Generalization with Weighted Artificial Negative Events. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1877–1889 (Aug 2014). https://doi.org/10.1109/TKDE.2013.130, https://ieeexplore.ieee.org/document/6573923, conference Name: IEEE Transactions on Knowledge and Data Engineering
19. Buijs, J., Dongen, B., van der Aalst, W.M.P.: Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity. International Journal of Cooperative Information Systems **23**, 1440001 (Mar 2014). https://doi.org/10.1142/S0218843014400012
20. Buijs, J., Dongen, B., van der Aalst, W.M.P.: On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. vol. 7565, pp. 305–322 (Sep 2012). https://doi.org/10.1007/978-3-642-33606-5_19
21. Burattin, A.: PLG2: Multiperspective Process Randomization with Online and Offline Simulations. In: Online Proceedings of the BPM Demo Track. Rio de Janeiro, Brasil (2016)
22. Burattin, A., Sperduti, A.: Heuristics Miner for Time Intervals. Computational Intelligence (2010)
23. Cardoso, J.: Control-flow Complexity Measurement of Processes and Weyuker's Properties. vol. 8, pp. 213–218 (Jan 2005)
24. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Information Systems **37**(7), 654–676 (Nov 2012). https://doi.org/10.1016/j.is.2012.02.004, https://www.sciencedirect.com/science/article/pii/S0306437912000464
25. Esmita, A., Gupta, E.: Process Mining A Comparative Study (Dec 2014). https://doi.org/10.17148/ijarcce
26. Ferrante, M., Saltalamacchia, M.: The Coupon Collector's Problem (2014),
27. Günther, C., van der Aalst, W.M.P.: Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. vol. 4714, pp. 328–343 (Sep 2007). https://doi.org/10.1007/978-3-540-75183-0_24
28. Janssenswillen, G., Depaire, B.: Towards Confirmatory Process Discovery: Making Assertions About the Underlying System. Business & Information Systems En-

gineering **61**(6), 713–728 (Dec 2019). https://doi.org/10.1007/s12599-018-0567-8, https://doi.org/10.1007/s12599-018-0567-8
29. Latva-Koivisto, A.: Finding a Complexity Measure for Business Process Models (Mar 2001)
30. Laue, R., Gruhn, V.: Complexity Metrics for business Process Models. pp. 1–12 (Jan 2006)
31. Leemans, S., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. pp. 311–329 (Jan 2013). https://doi.org/10.1007/978-3-642-38697-8_17
32. Leemans, S., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. vol. 171, pp. 66–78 (May 2014). https://doi.org/10.1007/978-3-319-06257-0_6
33. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Incomplete Event Logs. In: Ciardo, G., Kindler, E. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 91–110. Lecture Notes in Computer Science, Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_6
34. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Using Life Cycle Information in Process Discovery. In: Reichert, M., Reijers, H.A. (eds.) Business Process Management Workshops, vol. 256, pp. 204–217. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_17, http://link.springer.com/10.1007/978-3-319-42887-1_17, series Title: Lecture Notes in Business Information Processing
35. Lekic, J., Milicev, D.: Discovering Block-Structured Parallel Process Models from Causally Complete Event Logs. Journal of Electrical Engineering **67**(2), 111–123 (2016). https://doi.org/10.1515/jee-2016-0016, num Pages: 111-123 Place: Bratislava, Poland Publisher: De Gruyter Poland
36. Mannel, L., van der Aalst, W.M.P.: Finding Complex Process-Structures by Exploiting the Token-Game. pp. 258–278 (May 2019). https://doi.org/10.1007/978-3-030-21571-2_15
37. Mannel, L.L., Van Der Aalst, W.M.P.: Discovering Process Models with Long-Term Dependencies while Providing Guarantees and Filtering Infrequent Behavior Patterns. Fundamenta Informaticae **190**(2-4), 109–158 (2024). https://doi.org/10.3233/FI-242168
38. de Medeiros, A.K.A.: Genetic process mining (2006). https://doi.org/10.6100/IR614016, https://research.tue.nl/en/publications/genetic-process-mining(9f585688-ed7b-42b1-8446-6e80fc06c7db).html, publisher: [object Object]
39. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Mining and Knowledge Discovery **14**(2), 245–304 (Apr 2007). https://doi.org/10.1007/s10618-006-0061-7, https://doi.org/10.1007/s10618-006-0061-7
40. Mendling, J.: Detection and Prediction of Errors in EPC Business Process Models. Ph.D. thesis (Jan 2007), journal Abbreviation: Emisa Forum - EMISA Publication Title: Emisa Forum - EMISA Volume: 27
41. Muñoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: Hull, R., Mendling, J., Tai, S. (eds.) Business Process Management. pp. 211–226. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15618-2_16

42. Peng, W., Zhang, Z., Hildebrant, R., Ren, S.: Empirical Studies of Three Commonly Used Process Mining Algorithms. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 2492–2499 (Oct 2021). https://doi.org/10.1109/SMC52423.2021.9658861, https://ieeexplore.ieee.org/document/9658861, iSSN: 2577-1655
43. Rennert, C., Mannel, L., van der Aalst, W.M.P.: Improving the eST-Miner Models by Replacing Imprecise Structures Using Place Projection (Jun 2023)
44. Rozinat, A., Alves De Medeiros, A.K., Günther, C.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: Towards an Evaluation Framework for Process Mining Algorithms. Reactivity of Solids (Jan 2007)
45. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**, 64–95 (Mar 2008). https://doi.org/10.1016/j.is.2007.07.001
46. Sánchez-González, L., Garcia, F., Mendling, J., Ruiz, F., Piattini, M.: Prediction of Business Process Model Quality Based on Structural Metrics. vol. 6412, pp. 458–463 (Nov 2010). https://doi.org/10.1007/978-3-642-16373-9_35
47. Vanhoenshoven, F., Janssenswillen, G.: bupaverse/bupaRminer: process discovery algorithm for bupaR (2023), https://github.com/bupaverse/bupaRminer
48. Weber, P., Bordbar, B., Tino, P., Majeed, B.: A framework for comparing process mining algorithms (Feb 2011). https://doi.org/10.1109/IEEEGCC.2011.5752616
49. Weijters, A., van der Aalst, W.M.P., Medeiros, A.: Process Mining with the Heuristics Miner-algorithm, vol. 166 (Jan 2006), journal Abbreviation: Cirp Annals-manufacturing Technology - CIRP ANN-MANUF TECHNOL Publication Title: Cirp Annals-manufacturing Technology - CIRP ANN-MANUF TECHNOL
50. Weijters, A., Ribeiro, J.: Flexible Heuristics Miner (FHM). In: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp. 310–317 (Apr 2011). https://doi.org/10.1109/CIDM.2011.5949453, https://ieeexplore.ieee.org/document/5949453
51. Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice Constructs. Data Min. Knowl. Discov. **15**, 145–180 (Oct 2007). https://doi.org/10.1007/s10618-007-0065-y
52. Wen, L., Wang, J., van der Aalst, W.M.P., Huang, B., Sun, J.: A novel approach for process mining based on Event Types. Journal of Intelligent Information Systems **32**, 163–190 (Apr 2009). https://doi.org/10.1007/s10844-007-0052-1
53. Van der Werf, J.M., Polyvyanyy, A., Wensveen, B., Brinkhuis, M., Reijers, H.: All That Glitters Is Not Gold: Towards Process Discovery Techniques with Guarantees (Dec 2020)