2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
17-20 October, 2021. Melbourne, Australia

Empirical Studies of Three Commonly Used Process Mining Algorithms

Wenyu Peng[1], Zhenyu Zhang[1,2], Ryan Hildebrant[1], Shangping Ren[1]
[1]Department of Computer Science, San Diego State University, San Diego, CA 92182, USA
[2]Department of Computer Science, University of California, Irvine, CA 92697, USA
wpeng7231@sdsu.edu, zzhang4430@sdsu.edu, rhildebrant@sdsu.edu, sren@sdsu.edu

2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-4207-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMC52423.2021.9658861

***Abstract*— Process mining aims to extract useful process knowledge and provide valuable insights to better understand, monitor, and improve current business processes. The most critical learning task in process mining is process discovery. Process discovery takes an event log as an input and generates a process model as an output. In the last two decades, processing mining communities have proposed several process discovery algorithms. Many of these algorithms are based on or are extensions of three commonly used process mining algorithms. These algorithms are known as the α algorithm, the Heuristic algorithm and the Inductive algorithm. This study provides an evaluation of these three algorithms using both artificial event logs and real-life event logs. We study the impact of dependency patterns, noise, and complexity. Our work aims to provide clear guidelines for academics or business organizations that are interested in using process mining algorithms to discover their hidden process models and choose the most appropriate process discovery algorithm.**

I. INTRODUCTION

Over the past decade, *process mining* has emerged as a new research area that uses the available data in an organization, such as event logs, to better understand how processes are being executed in real life [31]. Given an event log, process mining aims to extract patterns and process knowledge to produce a process model. A process model can provide valuable insights that can help a business to understand, monitor, and improve a current process. The most crucial learning task in the broad field of process mining is *process discovery*. Process discovery is concerned with the derivation of process models from event logs.

Process discovery can be defined as extracting a control-flow model from an event log. To do this, the process discovery algorithm relies on a set of traces [21]. In an event log, each trace corresponds to a sequence of events. In the last two decades, several process discovery algorithms have been proposed [1], [3], [2], [4], [5], [6], [7], [8] that follow this technique. Additionally, these algorithms are based upon or are extensions of three commonly used process mining algorithms, which are: the α algorithm [1], the Heuristic algorithm [2], and the Inductive algorithm [8].

Throughout the literature, they often use two dimensions to measure the quality of process discovery algorithms; these dimensions are the *fitness* and the *precision*. The *fitness* measures the ability to replay the event logs from a discovered process model. The *precision* measures the level at which the generated traces from the discovered process model belong to the original event logs. We give the formal definition of these two metrics in section 2.4. In addition to these two metrics, we also consider each algorithm's running time as a quality metric to help show the trade-off between the running time of a specific algorithm and the results of the quality measurements.

Many comparisons have been made on different process mining algorithms in the traditional process mining literature. However, these comparisons are evaluated using small samples and a limited range of event logs. For example, Augusto et al. [26] conducted an experimental comparison based on twelve published real-life event logs. Another comparison by De Weerd et al. [27] focuses on eight real-life event logs and an additional twenty artificial event logs. Because of their small sample sizes, the conclusions in both works may require further investigation. Furthermore, these studies only focus on the given event logs and potential changes in the process discovery algorithms. Process model complexities and noise types that are inherent in the event logs are not studied in these works. As such, our work focuses on the experiments used in previous work but extends the depth of each experiment to account for various dependency patterns and noise types.

When using artificial event logs, we also consider another perspective called *rediscovery*. Rediscovery measures the similarity between the original process models that generate artificial event logs and the models discovered by the algorithms. The relationship between the previous two measurement metrics and rediscovery is shown in Fig. 1.

The purpose of this empirical study is to understand how each of these three commonly used process mining algorithms behave under different conditions. We achieve this by increasing the complexities of the process models by introducing different dependency patterns and applying different types of noise to artificial event logs. Additionally, this study will also look into the validity of the conclusions made by Augusto et al. [26] and De Weerd et al. [27] by introducing these new perspectives and extending the results. The ultimate goal of our work is to provide clear guidelines for academics or business organizations that are interested in using process mining algorithms to discover their hidden process models and choose the most appropriate process discovery algorithm.

The rest of the paper organize as follows: Section II introduces terms and definitions that are used throughout the paper. This section will also briefly summarize the related work in the field and provide quality measurements to help understand our results. In Section III, we present our

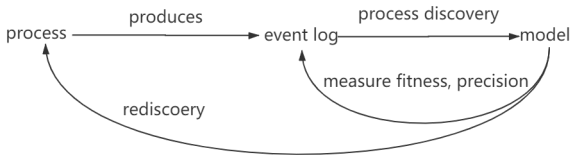


Fig. 1. Traditional model quality assessment (fitness, precision, generalization) and rediscovery.[13]

experimental evaluation in detail. Next, we list our findings and observations in Section IV. Lastly, we conclude and reveal our future work in Section V.

II. TERMS AND DEFINITIONS

In this section, we introduce terms and definitions that are used in the empirical study.

A. Event Logs

An event log is a set of traces. Each trace contains an identifier, a sequence of events, and other optional attributes. Events can be ordered merely as a sequence but are typically ordered based on a timestamp. We cite the formal definition given by Van Der Aalst [21].

Definition 1: (Event log [21]). *An event log L is a set of event traces, i.e., $L = \{\sigma_1, ..., \sigma_n\}$, where σ denote event trace. And we use $|L|$ to denote the number of traces in the event log.*

B. Process Model Representations

A process model describes a set of ordered activities that a trace must follow to be considered valid. Many different languages are used to model processes, such as BPMN (Business Process Modeling Notation), EPC (Event-Driven Process Chains), and Petri nets [19]. In this paper, we use Petri nets to model our processes.

Definition 2: (Petri net [19]). *A Petri net is a tuple $N = (P, T, F, \pi)$ where P and T are a set of places and transitions, respectively, $F \subseteq (T \times P) \cup (P \times T)$ is a set of directed arcs connecting places and transitions, $\pi : T \to A \cup \tau$ is a function mapping transitions to either activities or τ (unobservable activities).*

C. Process Discovery Algorithms

Several process discovery algorithms have been proposed in the last two decades. But we will focus on three families of commonly used process discovery algorithms:

- The α algorithm [1] and its derivatives;
- The Heuristic algorithm [2] and its derivatives
- The Inductive algorithm [8] and its derivatives

The α algorithm [1] is the first process discovery algorithm proposed by the process mining research community. The main idea behind the α algorithm is to find the relationships between each event and form a process model based on these relationships. There are four different relationships defined in the α algorithm: the *Sequence*, *choice*, *parallel*, and *loop* relations. An example of these four relationship patterns is shown in Fig 2. Although the α algorithm is easy to understand, it has many shortcomings. For example, the algorithm is not good enough to handle loops that are less than a length of 3. To overcome this challenge, many derivatives have been proposed, such as the $\alpha+$ [4], $\alpha++$ [11], and α^* [7]. These variants also explore the ordering relations of events and generate a more robust Petri net as a process model.

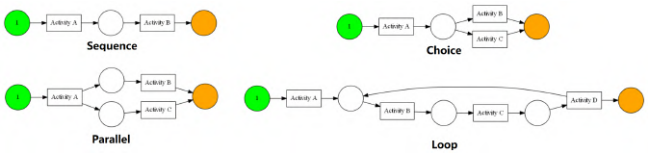


Fig. 2. Four basic dependency patterns in Petri net

The Heuristic algorithm [2] is a frequency-based approach. It takes the frequencies of events and sequences into account to construct a process model similar to a causal net [21]. Like the *alpha* algorithm, there are many variants, i.e., the Streaming Heuristic Miner [25] is proposed to handle streaming data.

The Inductive algorithm [8] uses a divide and conquer strategy to split the event log recursively into sub-logs. The Inductive miner uses the same set of relationships, i.e., *Sequence*, *choice*, *parallel*, and *loop* to construct process models. Similar to the α and heuristic algorithms, various inductive process discovery techniques have also been developed [12], [13]. The Inductive algorithm follows the rules to discover a model with perfect fitness that equal to 1 by adding invisible transitions.

D. Comparison Metrics

Our experimental study uses a set of comparison metrics that are commonly used in the literature. They are fitness, precision, F1 Score, Behavioral precision/recall, and structural precision/recall. These metrics will be displayed in section 3 and 4 to show the advantages and disadvantages of each algorithm. Their formal definitions are given below.

Definition 3: (Fitness [28]). *Let k be the number of different traces from the aggregated log. For each log trace i ($1 \leq i \leq k$), n_i is the number of process instances combined into the current trace, m_i is the number of missing tokens, r_i is the number of remaining tokens, c_i is the number of consumed tokens, and p_i is the number of produced tokens during log replay of the current trace. The token-based fitness metric f is defined as follows :*

$$f = \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}) \quad (1)$$

Note that, for all i, $m_i \leq c_i$ and $r_i \leq p_i$, therefore $0 \leq f \leq 1$.

Definition 4: (Precision [28]). *Let k be the number of different traces from the aggregated log. For each log trace i ($1 \leq i \leq k$), n_i is the number of process instances combined into the current trace, and x_i is the mean number of enabled transitions during log replay of the current trace (note that invisible tasks may enable succeeding labeled tasks but they are not counted themselves). Furthermore, T_V is the set of visible tasks in the Petri net model. The precision metric p*

is defined as follows:

$$p = \frac{\sum_{i=1}^{k} n_i(|T_V| - x_i)}{(|T_V| - 1) \cdot \sum_{i=1}^{k} n_i} \quad (2)$$

Definition 5: (F1 score [21]). *The fitness and precision are two aspects of a process model that may not always be consistent. The F1 score is defined as the harmonic mean of fitness f and precision p.*

$$F1 = \frac{2 \times f \times p}{f + p} \quad (3)$$

To measure rediscovery, we use behavioral and structural precision/recall to calculate the similarity between the referenced model and mined model [5].

Definition 6: (Behavioral precision and recall [29]). *Let σ be a trace in an event log. $L(\sigma)$ be the number of occurrences of σ in an event log. N_r and N_m be the respective Petri net for the reference and the mined models. C_r and C_m be the respective causality relations for N_r and N_m. The behavioral precision $B_p$and recall B_r is defined as:*

$$B_p = \frac{\sum_{\sigma \in L}\left(\frac{L(\sigma)}{|\sigma|} \times \sum_{i=0}^{|\sigma|-1} \frac{Enabled(C_r,\sigma,i) \cap Enabled(C_m,\sigma,i))}{Enabled(C_m,\sigma,i)}\right)}{\sum_{\sigma \in L} L(\sigma)} \quad (4)$$

$$B_r = \frac{\sum_{\sigma \in L}\left(\frac{L(\sigma)}{|\sigma|} \times \sum_{i=0}^{|\sigma|-1} \frac{Enabled(C_r,\sigma,i) \cap Enabled(C_m,\sigma,i))}{Enabled(C_r,\sigma,i)}\right)}{\sum_{\sigma \in L} L(\sigma)} \quad (5)$$

Definition 7: (Structural precision and recall [29]). *Let C_r and C_m be the respective causality relations for N_r and N_m. The structural precision and structural recall are defined as:*

$$S_p = \frac{|C_r \cap C_m|}{|C_m|} \quad (6)$$

$$S_r = \frac{|C_r \cap C_m|}{|C_r|} \quad (7)$$

III. EMPIRICAL STUDIES

This experimental study has two major goals: first, to compare the three commonly-used process mining algorithms under different conditions. In particular, we will study the impact of (1) dependency patterns, (2) the complexities of process models, (3) and the noise types that are inherent in a given event log. To do this, we will measure the quality of each process discovery algorithm in terms of the measurement criteria we defined in II. Although there are many comparisons done in the literature on different process discovery algorithms, to the best of our knowledge, no study has been done in evaluating different process mining algorithms' resilience to dependency patterns, complexity levels and noise changes. The second goal is to further validate the conclusions made by Augusto and De Weerd by applying the quality measures to real-life event logs and a small set of artificial event logs.

A. Datasets and Setup

In order to achieve these two major goals, we use both artificial event logs and real-life event logs.

Dataset: We use two collections of artificial event logs. The first is a collection of public artificial event logs downloaded from 4TU.ResearchData. These event logs have been made available from the Eindhoven University of Technology [24]. In this collection, we use twelve artificial event logs all containing one thousand event traces. The number of distinct activities ranges from eight to eighty. There is only one dependency pattern inherently existing in each event log. The second set of artificial logs are generated from the Processes and Logs Generator (Plg) [20]. Plg is an open-source software tool that creates process models and generates artificial event logs. We generate artificial event logs based on the complexities of process models and the types of noise. We also use seven real-life event logs downloaded from the same website [23]. The characteristics of these real-life event logs are summarized in Table I.

TABLE I
REAL-LIFE EVENT LOG

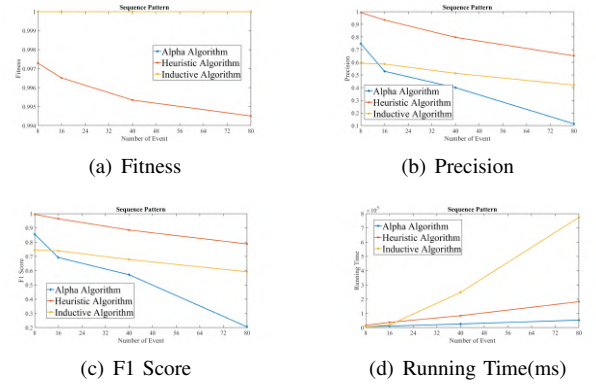
Real-life Event Log	#Trace	#Event	$\neq$ Event	Average Event
BPIC 2013	819	2351	5	2.87
BPIC 2020	10500	56437	17	5.37
Hospital Billing	100000	451359	18	4.51
Prepaid Travel Cost	2099	18246	29	8.69
Receipt Phase	1434	8577	27	2.07
Review Process	10000	236360	20	23.63
Road Traffic Fine	150370	561470	11	3.73

Setup: We performed our experiments using a single machine with the following hardware and software: an Intel(R) Core(TM) CPU i7-9750H @2.60GHz with 32GB RAM running Java 8 and python 3.7.

B. Comparison of The Process Mining Algorithms under Different Conditions

The Impact of Dependency Patterns Our first set of experiments explores whether the presence of a dependency pattern in a given event log can impact the quality of each process discovery algorithm. In this experiment, we assume that one dependency pattern is present in a log and we apply three process mining algorithms to each log. More specifically, given an artificial event log, we apply the three commonly-used process mining algorithms using ProM and record the running time. ProM is an open-source framework for process mining algorithms[14] that can be used to generate different types of process models. For the process models derived from the three algorithms, we apply quality measurements to calculate the fitness, precision, and F1 score. Fig. 4, Fig. 3, Fig. 5, and Fig. 6 show the results of the fitness, precision, F1 score, and running time applied to each dependency pattern: the choice pattern, the sequence pattern, the parallel pattern, and the loop pattern.

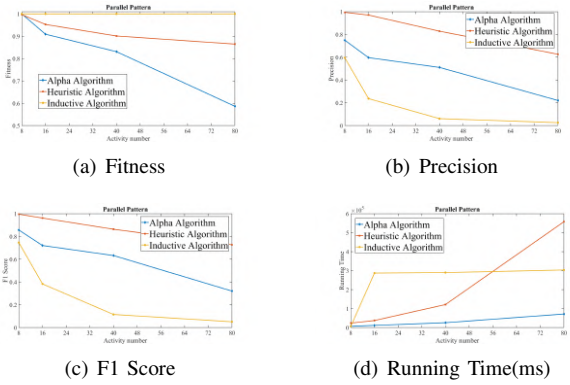
The Impact of Reference Model Complexity This set of experiments is designed to evaluate the different complexities of process models that can be inherent in event logs. For evaluating the quality of algorithms, six levels of complexity are considered by using four different dependency patterns [29]. Level 0 only contains the sequence pattern. Levels 1 and 2 incorporate the choice pattern and the parallel pattern into a sequence pattern. Level 3 combines the parallel pattern and the choice pattern into a sequence pattern. Level 4



(a) Fitness (b) Precision

(c) F1 Score (d) Running Time(ms)

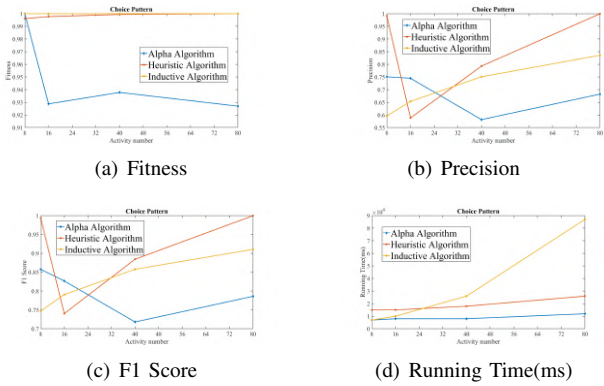
Fig. 3. Sequence Pattern



(a) Fitness (b) Precision

(c) F1 Score (d) Running Time(ms)

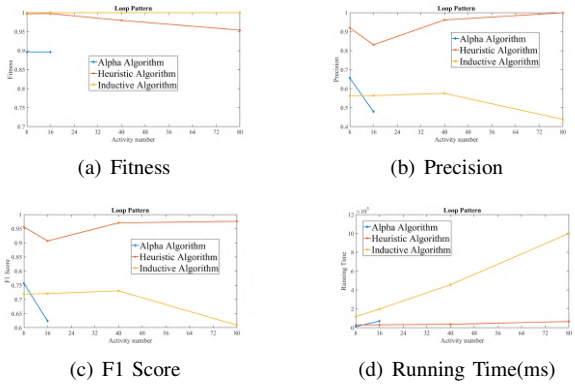
Fig. 5. Parallel Pattern



(a) Fitness (b) Precision

(c) F1 Score (d) Running Time(ms)

Fig. 4. Choice Pattern



(a) Fitness (b) Precision

(c) F1 Score (d) Running Time(ms)

Fig. 6. Loop Pattern

then combines all the dependency patterns. Lastly, Level 5 shows a more complex model where we also have all four dependency patterns. However, at this level, the number of activities for each pattern is increased. For instance, we can have a parallel pattern with three activities, such as B, F, and G. In the previous levels, we only consider parallel patterns with two activities such as D and E.

In this experiment, we randomly generated ten process models for each level of complexity as reference models. For each reference process model, we randomly generate five artificial event logs using the Plg. For this set of artificial logs, we assume that there are one thousand event traces. We apply the α algorithm, the Heuristic algorithm, and the Inductive algorithm to each artificial event log and discover the mined models. Finally, we calculate the average behavior precision B_p, behavior recall B_r, structural precision S_p and structural recall S_r for reference models and the mined models. The results are depicted in Table II.

The Impact of Noise Types Our final experiments explore whether different noise types can impact the quality of the three process mining algorithms. There are seven types of noise as follows: missing head of a trace, missing tail of a trace, missing episode of a trace, a trace with perturbed order, a trace with an additional event, a trace with an alien event, and traces with a changed name event [22]. For each type of noise, we generate 300 artificial event logs and inject them with different amounts of noise ranging from 0% to 20%. These noise ranges are incremented by 5%. These event logs are generated from sixty process models using the six complexity levels that we have mentioned above. For each artificial event log, we apply the three commonly used process mining algorithms to obtain the process models and calculate the average fitness and precision using pm4py, which is an open-source framework for python users [15]. The results are shown in Fig. 8, Fig. 9, Fig. 10.

C. Comparison of The Process Mining Algorithms on Real-life Event Logs

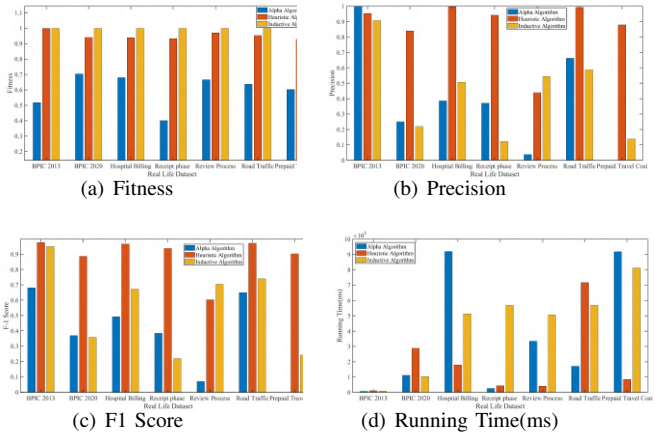
In these experiments, we first apply the process mining algorithms on seven real-life event logs that are presented in Table I to discover the process models. We use pm4py to calculate the fitness, precision, F1 score, and the running time. The results are depicted in Fig.7.

IV. FINDINGS AND DISCUSSIONS

A. The Impact of Dependency Patterns

In general, the Heuristic algorithm has a higher F1 score, the Inductive algorithm has a higher fitness, abd the α algorithm has the fastest running time. Below we discuss the effects of each of the four dependency patterns.

For sequence patterns, the order of activities is critical when decide the relations. The α algorithm determines two activities as non relations when the order is not consistent.



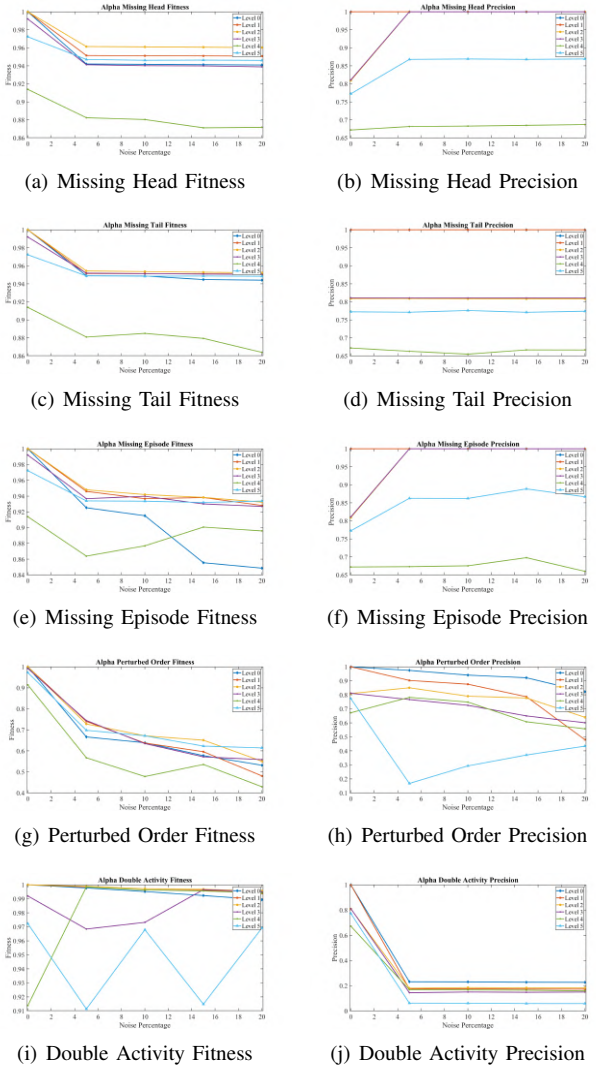
(a) Fitness (b) Precision

(c) F1 Score (d) Running Time(ms)

Fig. 7. Real-life Event Logs

TABLE II
REDISCOVERY

Level	Algorithm	B_p	B_r	S_p	S_r
0	Alpha	1	1	1	1
0	Heuristic	1	1	1	1
0	Inductive	1	1	1	1
1	Alpha	1	1	1	1
1	Heuristic	1	1	1	1
1	Inductive	1	1	1	1
2	Alpha	1	1	1	1
2	Heuristic	1	1	1	1
2	Inductive	0.696	1	0.759	1
3	Alpha	1	1	1	1
3	Heuristic	1	1	1	1
3	Inductive	0.841	1	0.808	1
4	Alpha	1	0.8758	1	0.97
4	Heuristic	1	1	1	1
4	Inductive	0.883	1	0.837	1
5	Alpha	1	0.9204	1	0.9181
5	Heuristic	1	1	1	1
5	Inductive	0.7574	1	0.802	1



(a) Missing Head Fitness (b) Missing Head Precision

(c) Missing Tail Fitness (d) Missing Tail Precision

(e) Missing Episode Fitness (f) Missing Episode Precision

(g) Perturbed Order Fitness (h) Perturbed Order Precision

(i) Double Activity Fitness (j) Double Activity Precision

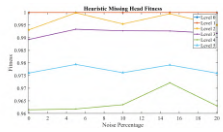
Fig. 8. Alpha Algorithm

So these activities are not connected in the process model which holds a higher fitness because it can be replayed anywhere. However, precision tends to decrease because it allows more behaviors. The reason cause fitness decrease using the Heuristic algorithm is that some arcs are not shown due to the frequency, so we missed tokens when we calculate fitness. The Inductive algorithm keeps a higher fitness by using invisible transitions, so fitness is high and approaching 1. When activity numbers increase, the model is huge, which allows more behaviors in sequence patterns. This causes precision decrease in three algorithm. So the Heuristic algorithm is better when considering F1 score.

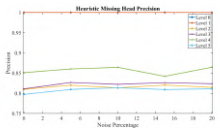
For choice patterns, the α algorithm is better when activity numbers are small. When we consider a more complex model with more activity numbers, the Heuristic algorithm is better. However, fitness and precision decrease and then increase for both the α algorithm and the Heuristic algorithm as the complexity increases. Overall, the Inductive algorithm holds the highest fitness, and its precision increases as the activity numbers increase.

For parallel patterns, when the activity numbers increase, fitness and precision of all three algorithms decrease. Fitness decrease dramatically using the α algorithm. Because for some parallel exist once, the α algorithm consider it as choice, which cause missing tokens. Precision decreases for the Inductive algorithm because it adds invisible transitions for each parallel pattern, which leads to a "flower model". The flower model allows for all behaviors to be observed. Therefore, it has perfect fitness and zero precision, which abstracts away the transactions of each event [27].

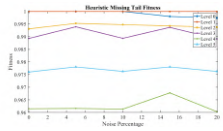
For loop patterns, the α algorithm is not useful. When the activity numbers increases, the α has some errors using ProM and cannot create a Petri net within two hours. Therefore, the result just shows the activities number to 16. The running time for the Inductive algorithm is very high. Therefore, the Heuristic algorithm should be considered the best option for handling loops in terms of the running time and the F1 value.



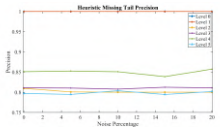
(a) Missing Head Fitness



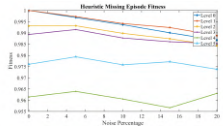
(b) Missing Head Precision



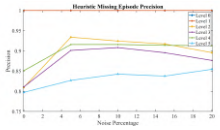
(c) Missing Tail Fitness



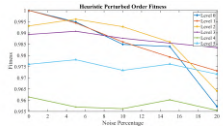
(d) Missing Tail Precision



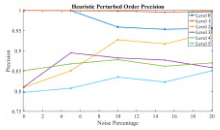
(e) Missing Episode Fitness



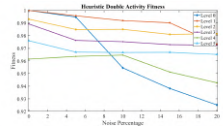
(f) Missing Episode Precision



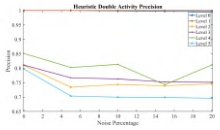
(g) Perturbed Order Fitness



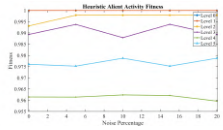
(h) Perturbed Order Precision



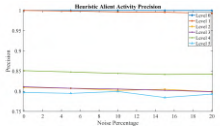
(i) Double Activity Fitness



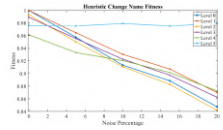
(j) Double Activity Precision



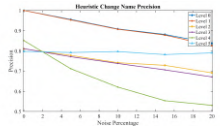
(k) Alient Activity Fitness



(l) Alient Activity Precision

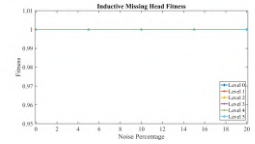


(m) Change Name Fitness

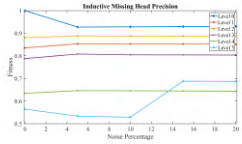


(n) Change Name Precision

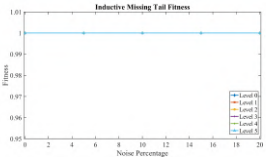
Fig. 9. The Heuristic Algorithm



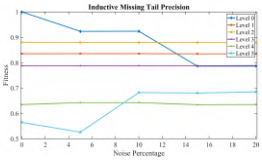
(a) Missing Head Fitness



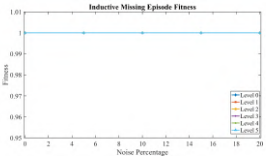
(b) Missing Head Precision



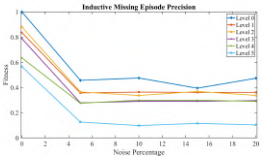
(c) Missing Tail Fitness



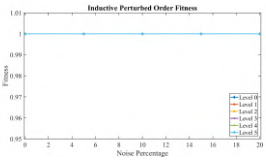
(d) Missing Tail Precision



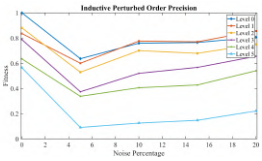
(e) Missing Episode Fitness



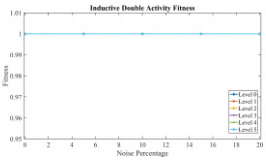
(f) Missing Episode Precision



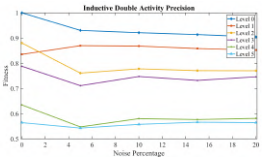
(g) Perturbed Order Fitness



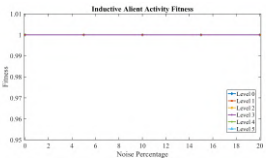
(h) Perturbed Order Precision



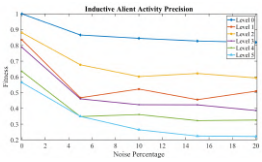
(i) Double Activity Fitness



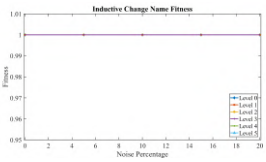
(j) Double Activity Precision



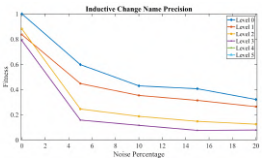
(k) Alient Activity Fitness



(l) Alient Activity Precision



(m) Change Name Fitness



(n) Change Name Precision

Fig. 10. The Inductive Algorithm

B. The Impact of Reference Model Complexities

In general, in lower levels, we can use the α algorithm. In high levels, we should use the Heuristic algorithm.

In the first two levels, the models are simple, and each algorithm can discover the referenced model. However, the α algorithm cannot handle short loops that are less than a length of 3. When a short loop appears in level 4, the discovered model from the α algorithm is smaller than the reference model, so the B_r and S_r decrease. As we approach level 5, the total activity numbers increase, which makes the percentage of loops decreases. Thus, the B_r and S_r are higher than what is observed in level 4. Overall, the Heuristic

algorithm is better at rediscovery because it can detect all the relations that exist in the referenced model. As we approach high levels, the Inductive algorithm uses invisible transitions to connect parallel relations. These invisible transitions cause the B_p and S_p to decrease. In level 3 and level 4, the parallel relations percentage starts to decrease while the B_p and S_p increases. In level 5 the parallel percentage increases, whereas B_p and S_p start to decrease.

C. The Impact of Noise Types

In general, for missing head of noise and missing tail of noise, we can use the α algorithm. For missing episode noise, perturbed order noise, doubled activity noise, alien event noise, and change name noise, we can use the Heuristic algorithm. If we just consider fitness, we can use the Inductive algorithm. Then we talk about each noise types with the processing mining algorithms.

The quality of the α algorithm is stable when handling missing head and tail noise. To handle the missing head and tail noise, the α algorithm adds an additional arc to jump over the missing head or tail from the start to the second head or from the second tail to the end. So fitness decreases due to the missing tokens. However, it is more accurate and has higher precision. Missing episode noise can also cause a relational change, which causes fitness and precision decreases. Perturbed order noise causes a large decrease in fitness because it changes the order of events and greatly affects the relations between events [22]. An example of this is when we observe a change from a directly follows relation to a parallel relation. If doubled event noise is observed, it creates a self-loop relation [22]. The α algorithm cannot solve a self-loop relation so fitness and precision decrease dramatically. The α algorithm cannot handle alient event noise and changing name event noise. When we observe these noise types, we cannot generate a Petri net using ProM within two hours. Therefore, we do not include this data in Fig. 8. When the number of alient events increases, it takes a huge amount of time for the α algorithm to determine the relations which make it difficult to discover the Petri net.

The Heuristic algorithm considers the probability of relations and ignores rare activities. As a result, when rare activities exist in event logs, the Heuristic algorithm have the highest fitness and precision comparing to other algorithms. For missing head and tail noise, the quality of the Heuristic algorithm is stable. For missing episode noise, when the noise percentage increases, fitness decreases. However, precision increases because it becomes more accurate. For perturbed order noise, fitness decreases when the complexity level is small, especially in level 0 and level 2. For double event noise, fitness decreases quickly in level 0 because we are changing the sequence pattern to a self-looping pattern. Fitness also decreases because some rare self-loops are ignored. Precision decreases and eventually maintain stability. The Heuristic algorithm can handle the alient event noise, because it considers alient events as rare events and ignore it. This keeps fitness and precision of the Heuristic algorithm stable. However, change name noise is a challenge for the Heuristic algorithm. In level 1, an addition of 5% noise causes a 4.2% decrease in fitness and 4.5% decrease in precision. In level 5, when the model is very complex, the quality of the Heuristic algorithm is still high because small changes don't affect the whole structure of the model.

The Inductive algorithm always keeps fitness approaching 1. For every measurement of fitness shown in Fig 10, the value is equal to 1.0. However, to achieve better fitness, there must be a trade-off in precision [26]. Precision can also be affected when an event log contains missing head noise or missing tail noise. In simple models, there tends to be a large decrease in precision, while more complicated models tend to stabilize. Missing episode noise causes a decrease in precision. In order to achieve a higher fitness, the Inductive algorithm adds more invisible transitions on models. In lower levels, the difference is not huge. However, in level 5, an addition of 5% noise causes a 77.6% decrease in precision. For perturbed order noise, an addition of 5% noise causes a huge decrease in precision. However, when the noise increases, precision starts to also increase. Because the invisible transitions is considered as a ground truth. When considering the doubled event, the quality of the Inductive algorithm is stable. For alient event noise, the Inductive algorithm also have issues. To deal with alient events, it adds more invisible transitions for the alient place in the model. This decision creates a model with a higher fitness but precision decreases. Changing name noise influences all 3 algorithms. When a model is complex, the Inductive algorithm cannot discover a proper model within two hours. As a result, there are no results in level 4 or level 5.

D. The Performance against Real-life Event Logs

In this section, we talk about the conclusion we made compared to Augusto and De De Weerd. In the first conclusion De Weerd made, there exists an important difference between evaluation of process discovery algorithms based on either artificial or real-life event logs. However our results shows when the model become complex with noise, the difference between artificial and real-life event logs is not huge. We can get a "spaghetti-like" process model using both artificial and real-life event logs. Also for Augusto, the Heuristic algorithm seems better when handling real-life event logs, and same for artificial event logs when considering F1 score. Inductive algorithm is not good in fitness, but not good at F1 score.

V. CONCLUSION AND FUTURE WORK

In this paper, we first illustrate different quality metrics of process discovery algorithms when handling artificial event logs with different dependency patterns. Next, we use rediscovery to determine if the algorithms can rediscover the referenced process model with six levels of complexity. Then, we introduce different types of noise to the event logs and measure the changes. Lastly, we apply 3 algorithms on real-life event logs to see the quality. To conclude of our work, we use Table III, IV, V to show under which circumstances, we should choose which algorithm.

There are many issues behind real-life event logs that still need to be found i.e. scenarios and infrequent behavior will

TABLE III

CONCLUSION 1

Pattern	Algorithm	Fitness	Precision	F1 Score	Running Time
Sequence	Alpha	×			×
	Heuristic		×	×	
	Inductive	×			
Choice	Alpha				×
	Heuristic		×	×	
	Inductive	×			
Parallel	Alpha				×
	Heuristic		×	×	
	Inductive	×			
Loop	Alpha				
	Heuristic		×	×	×
	Inductive	×			

TABLE IV

CONCLUSION 2

Level	Alpha	Heuristic	Inductive
0	×	×	×
1	×	×	×
2	×	×	
3	×	×	
4		×	
5		×	

TABLE V

CONCLUSION 3

Condition	Metric	Alpha	Heuristic	Inductive
Missing Head	Fitness			×
Missing Head	Precision	×		
Missing Tail	Fitness			×
Missing Tail	Precision	×		
Missing Episode	Fitness			×
Missing Episode	Precision		×	
Perturbed Order	Fitness			×
Perturbed Order	Precision		×	
Double Activity	Fitness			×
Double Activity	Precision		×	
Alien Activity	Fitness			×
Alien Activity	Precision			
Change Name	Fitness		×	
Change Name	Precision		×	

also impact the quality of process discovery algorithms [31]. For future work, we will use machine learning techniques to analyze a given event log. After analyzing the patterns, noise and complexity, we hope to predict which algorithm is best suited for a particular event log.

VI. ACKNOWLEDGEMENT

This work is in part supported by NSF funding under NSF 1952247, NSF 1952225, and NSF 1929469.

REFERENCES

[1] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE transactions on knowledge and data engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[2] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[3] A. K. A. de Medeiros, B. F. van Dongen, W. M. van der Aalst, and A. Weijters, "Process mining for ubiquitous mobile systems: an overview and a concrete algorithm," in *International Workshop on Ubiquitous Mobile Information and Collaboration Systems*. Springer, 2004, pp. 151–165.

[4] L. Wen, W. M. Van Der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 145–180, 2007.

[5] A. K. A. de Medeiros, A. J. Weijters, and W. M. van der Aalst, "Genetic process mining: an experimental evaluation," *Data Mining and Knowledge Discovery*, vol. 14, no. 2, pp. 245–304, 2007.

[6] C. W. Günther and W. M. Van Der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," in *International conference on business process management*. Springer, 2007, pp. 328–343.

[7] L. Wen, J. Wang, W. M. van der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks," *Data & Knowledge Engineering*, vol. 69, no. 10, pp. 999–1021, 2010.

[8] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *International conference on applications and theory of Petri nets and concurrency*. Springer, 2013, pp. 311–329.

[9] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. Weijters, and W. M. van der Aalst, "The need for a process mining evaluation framework in research and practice," in *International Conference on Business Process Management*. Springer, 2007, pp. 84–89.

[10] V. Huser, "Process mining: Discovery, conformance and enhancement of business processes," 2012.

[11] L. Wen, J. Wang, and J. Sun, "Mining invisible tasks from event logs," in *Advances in Data and Web Management*. Springer, 2007, pp. 358–365.

[12] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *International conference on business process management*. Springer, 2013, pp. 66–78.

[13] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 2014, pp. 91–110.

[14] H. Verbeek, J. C. Buijs, B. F. Van Dongen, and W. M. Van Der Aalst, "Xes, xesame, and prom 6," in *International Conference on Advanced Information Systems Engineering*. Springer, 2010, pp. 60–75.

[15] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process mining for python (pm4py): bridging the gap between process-and data science," *arXiv preprint arXiv:1905.06169*, 2019.

[16] A. Adriansyah, N. Sidorova, and B. F. van Dongen, "Cost-based fitness in conformance checking," in *2011 Eleventh International Conference on Application of Concurrency to System Design*. IEEE, 2011, pp. 57–66.

[17] T. Chatain and J. Carmona, "Anti-alignments in conformance checking–the dark side of process models," in *International Conference on Application and Theory of Petri Nets and Concurrency*. Springer, 2016, pp. 240–258.

[18] J. C. Buijs, B. F. van Dongen, and W. M. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *International Journal of Cooperative Information Systems*, vol. 23, no. 01, p. 1440001, 2014.

[19] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[20] A. Burattin, "Plg2: Multiperspective process randomization with online and offline simulations." in *BPM (Demos)*, 2016, pp. 1–6.

[21] W. Van Der Aalst, "Data science in action," in *Process mining*. Springer, 2016, pp. 3–23.

[22] C. W. Günther, "Process mining in flexible environments," 2009.

[23] 4TU.ResearchData, 2021. [Online]. Available: `https://data.4tu.nl/portal`

[24] W. van der Aalst, 2020. [Online]. Available: `https://data.4tu.nl/articles/dataset/Benchmarking_logs_to_test_scalability_of_process_discovery_algorithms/12681647`

[25] A. Burattin, A. Sperduti, and W. M. van der Aalst, "Heuristics miners for streaming event data," *arXiv preprint arXiv:1212.6383*, 2012.

[26] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 4, pp. 686–705, 2018.

[27] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Information Systems*, vol. 37, no. 7, pp. 654–676, 2012.

[28] A. Rozinat and W. M. Van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[29] H.-J. Cheng and A. Kumar, "Process mining on noisy logs: can log sanitization help to improve performance?" *Decision Support Systems*, vol. 79, pp. 138–149, 2015.

[30] J. C. Buijs, B. F. Van Dongen, and W. M. van Der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2012, pp. 305–322.

[31] Z. Zhang, C. Guo, and S. Ren, "Mining timing constraints from event logs for process model," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 1011–1016.