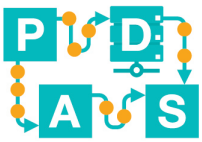


RWTH AACHEN UNIVERSITY

Model Repair by Incorporating Negative Instances In Process Enhancement

Master Thesis

Author : **Kefang Ding**

Supervisor : Dr. Sebastiaan J. van Zelst

Examiners : Prof. Wil M.P. van der Aalst
Prof. Thomas Rose

Registration date : 2018-11-15

Submission date : 2019-04-08

This work is submitted to the institute

PADS RWTH University

Acknowledgments

The acknowledgments and the people to thank go here, don't forget to include your project advice.

Abstract

Process mining is based on business execution history in the form of event log, aim to bring visual insights on the business process and to support process analysis and enhancements. It bridges the gap between traditional business process management and advanced data analysis techniques like data mining and gains more interests and application in recent years.

Process enhancement, as one of the main focuses in process mining, improves the existing processes according to actual business execution in the form of event logs. The records in an event log can be classified as positive and negative according to predefined Key Performance Indicators, e.g. the throughput time, and cost. Most of the current enhancement techniques only consider positive instances from an event log to improve the model, while the value hidden in negative instances is simply neglected.

This thesis provides a novel strategy that considers not only the positive instances and the existing model but also incorporate negative information to enhance a business process. Those factors are balanced on directly-follows relations of activities and generate a process model. Subsequently, long-term dependencies of activities are detected and added to the model, in order to block negative instances and obtain a higher precision.

We validate the ability of our methods to incorporate negative information with synthetic data at first. Then, we conduct experiments in a scientific workflow platform KNIME to show the statistical performance of our methods. The results showed that our method is able to overcome the shortcomings of the current repair techniques in some situations and repair models with a higher precision.

Chapter 1

Introduction

Process mining is a relatively new discipline that bridges the gap of data mining and business process management. The objective of process mining is to support the analysis of business processes, provide valuable insights in processes and further improve the business execution based on the business execution data which is recorded in event logs. According to [1], process mining techniques are divided into three categories: *process discovery*, *conformance checking*, *and process enhancement*. *Process discovery* techniques derive visual models from event logs of the information system, aiming at a better understanding of real business processes. *Conformance checking* analyzes the deviations between a referenced process model and observed behaviors driven from its execution. *Process enhancement* adapts and improves existing process models by extending the model with additional data perspectives or repairing the existing model to accurately reflect observed behaviors.

Most of the organizations have predefined process execution rules which are captured in a process model. However, in real life, business processes often encounter exceptional situations where it is necessary to execute process differing from the reference model. To reflect reality, the organizations need to adjust the existing process model. Basically, one can apply process discovery techniques again to obtain a new model only based on the event log. However, there is a need that the improved model should be as similar as possible to the original model while replaying the current process execution[2]. In this situation, the rediscovery method tends to fail due to the ignorance of the impact from the existing model. To meet this need, *model repair* techniques are proposed in [2].

Model repair belongs to process enhancement[2]. It analyzes the workflow deviations between an event log and a process model, and fix the deviations mainly by adding subprocesses on the model. As known, organizations are goal-oriented and aims to have high performance according to a set of Key Performance Indicator(KPI)s,e.g. average conversion time for the sales, payment error rate for the finance. However, little research in process mining is conducted on the basis of business performance[3]. The authors of [3] point out several contributions like [4] to consider business performance into process mining. The work in [4] divides deviations of model and the event log into positive and negative according to certain KPIs. Then it applies repair techniques in [5] only with positive deviations, to avoid introducing negative instances into the repaired model.

However, the current repair methods have some limits. Model repair techniques fix the model by adding subprocesses. They guarantee that the repaired model replays well the event log but overgeneralizes the model, such that it allows more behaviors than expected. Furthermore, it increases the model complexity. Even the performance is considered in

[4], but only deviations in positive is used to add subprocesses, the negative information is ignored, which disables the possibility to block negative behaviors from model.

In the following part, motivating examples are given to describe those limits of the current repair techniques in several situations. Then we propose research questions to overcome those limits and define our research scope. At the end, we give the outline for the whole thesis.

1.1 Motivating Examples

This section describes some situations where current repair techniques tend to fail. For the sake of understanding, examples are extracted from the common master study procedure to illustrate those situations.

The main activities for the master study include *register master*, *finish courses* and *write a master thesis*. Here, we simplify the *finish courses* and only extend the activity *write a master thesis* into a set of sub activities. Those activities are shown in the Petri net model M_0 of Figure 1.1. The activities are modeled by the corresponding ***transitions*** which is represented by a square. Transitions are connected through a circle called ***place***. Transitions and places build the static structure of Petri net and describe the transition relations. ***Tokens*** in the black dot are put in places and represent the dynamic state of the model.

M_0 is currently in an initial state where one token is at the start place to enable the transition *register master*. After firing *register master*, the token at the initial place is consumed while two new token are generated in the output places of *register master*. In this way, activity *finish courses* can be executed concurrently the other branch except for the *get degree*. When multiple activities have the same input place, all of them are enabled but only one of them can be fired and executed, namely, they are exclusive to each other. As shown in the figure, *select existing topics* and *create new topics* are exclusive, and only one of them can be triggered. When a transition has multiple input places, it can be triggered with condition that all input places hold at least a token. *Get degree* is enabled only after *finish courses* and *representation* done.

Along the tokens flowing through the model, activities get fired and generate a sequence according to their execution order. One execution sequence is called a trace. A set of traces depicts the model behavior and is recorded into a data file called event log. In real life, activities might be executed with deviation to the process model. A trace which has no deviation to the model is fitting. Otherwise, it's an unfitting trace. With accumulation of deviations, process model needs

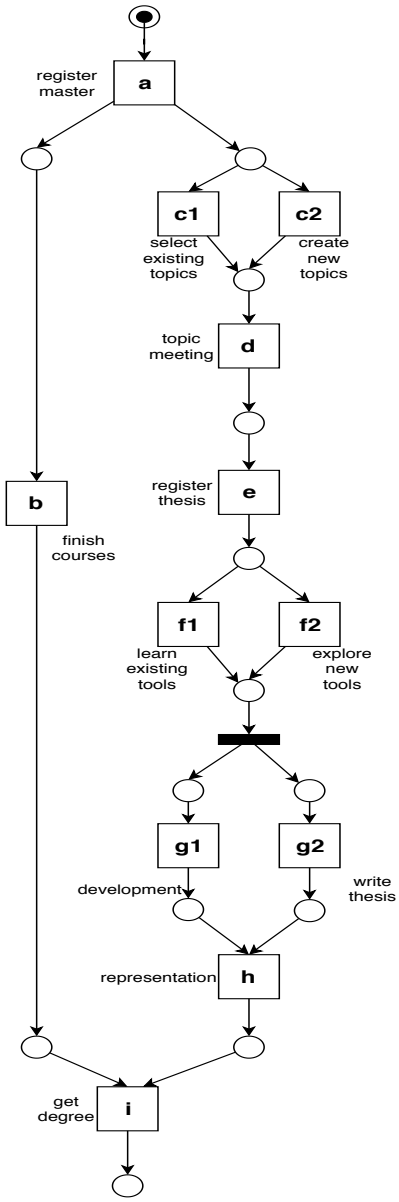


Figure 1.1: original master study process M_0

amending to reflect reality. In the following part, In the following part, several situations are introduced to demonstrate the shortcomings of current techniques to repair a model.

1.1.1 Situation 1: Repairing Model with Unfitting Traces

In some universities, before registering a master thesis, the activities *write proposal* and *check course requirement* with exclusive choice relation might be necessary in the master study procedure. The real process are recorded in the event log L_1. Traces with either of those activities are considered as positive. For convenience, alphabet characters are used to represent the corresponding activities and annotated in the model. **x1, x2** represent the activities *write proposal* and *check course requirement*.

$$\begin{aligned} L_1 := \{ &< a, b, c1, d, \mathbf{x1}, e, f1, g1, g2, h, i >^{50,pos}, \\ &< a, b, c2, d, \mathbf{x2}, e, f2, g2, g1, h, i >^{50,pos} \} \end{aligned}$$

Because the existing repair techniques [5] don't consider the performance of traces in event log, all instances with positive labels are used to repair the model. Firstly, the deviations of the existing model M0 and the event log L_1 are computed. After computation of deviations, each deviation has the same start and end place and two deviations appear at the same position in the model. When repairing this model, each subprocess has one place as its start and end place, which forms a loop in the model. If there is only one such subprocess, the subprocess is added in a sequence in the model, which leads to a higher precision. Yet the algorithm does not discover orderings between different subprocesses at overlapping locations. So the subprocesses are kept in a loop form.

The repaired model is shown in Figure 1.2a, where the two additional activities are added in the form of loop. The repair algorithm in [4] builds upon [5] and considers the performance of the event log. However, the repaired model is the same as the one in Figure 1.2a. The reasons are: (1) there is no deviation from negative factors. (2) positive deviations are used in the same way as [5].

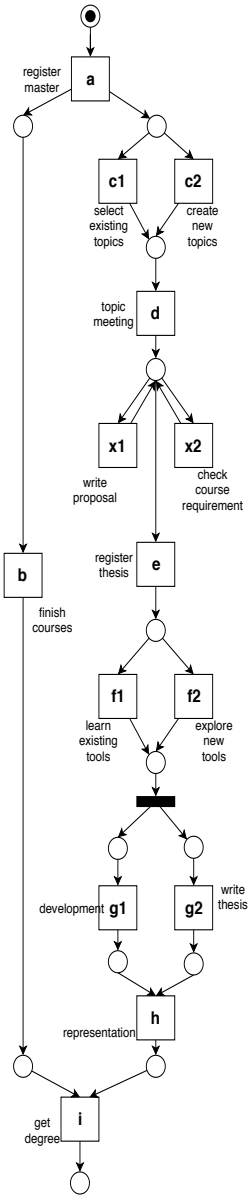
Compared to the model in Figure 1.2a where the two extra activities are shown in loop, the model in Figure 1.2b are more expected, since it includes the two activities in sequence and has a higher precision.

1.1.2 Situation 2: Repairing A Model with Fitting Traces

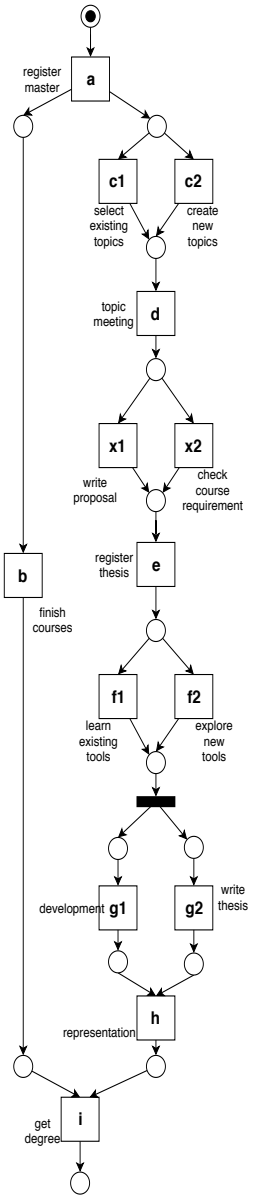
This situation describes the existing problem in the current methods that fitting traces with negative performance outcomes cannot be used to repair a model. Given an actual event log L_2, when activity *finish courses* is fired after *begin thesis* and before writing master thesis, it reduces the pressure for the master thesis phase and traces in such an order are treated as positive. Else, the negative outcomes are given.

$$\begin{aligned} L_2 := \{ &< a, \mathbf{b}, c1, d, e, f1, g2, g1, h, i >^{50,pos}, \\ &< a, \mathbf{b}, c2, d, e, f2, g1, g2, h >^{50,pos}; \\ &< a, c1, d, e, f2, g2, g1, \mathbf{b}, h, i >^{50,neg}, \\ &< a, c1, \mathbf{b}, d, e, f1, g1, g2, h, i >^{50,neg} \} \end{aligned}$$

Compared to the model, the event log L_2 contains no deviations. When we apply the techniques in [5] and [4] to repair the model, the model keeps untouched due to no deviation.



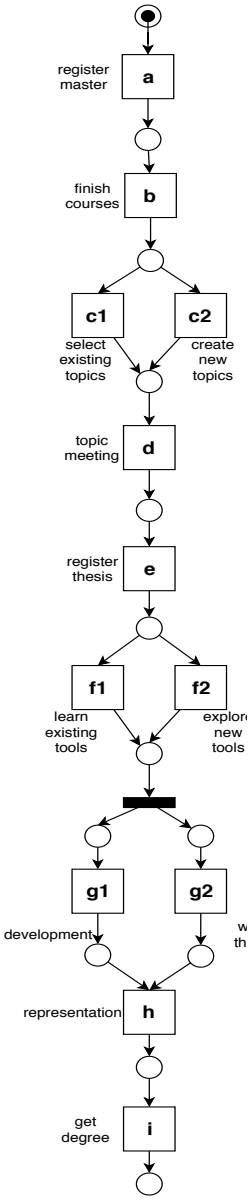
(a) repaired model $M_{1.1}$ with additional activitiesties



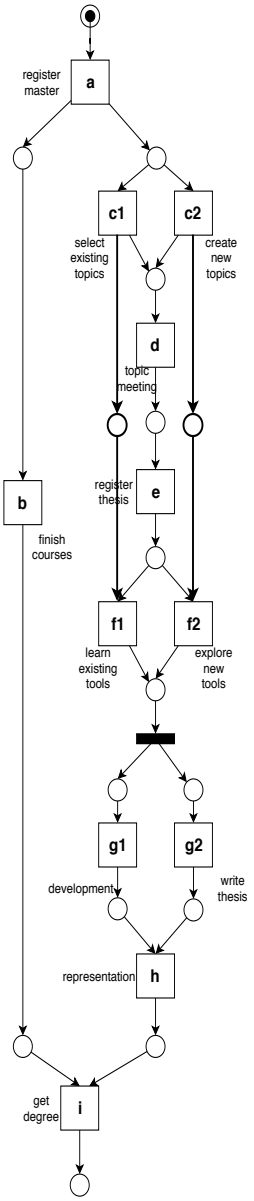
(b) expected model $M_{1.2}$ with additional activi-

Figure 1.2: example for situation 1 where $M_{1.1}$ is repaired by adding subprocess in the form of loops, which results in lower precision compared with the expected model $M_{1.2}$.

Apparently, the reason that those two methods can't incorporate the negative information in fit traces causes this shortcoming. When we expect a model which enforce the positive instances and avoid the negative instance as the model M_2, the current methods don't allow us to obtain such results.



(a) expected model M_2 with order change



(b) model M_3 with long-term dependency

Figure 1.3: example for situation 2 and 3

1.1.3 Situation 3: detect long-term dependency

This part introduces a problem which causes a lower precision in process mining. It is the inability in current methods to detect the long-term dependency in the Petri net. The long-term dependency describes the phenomenon that one execution choice decides

the execution of activities that do not follow directly. Due to the long distance of this dependency, current methods cannot detect it and improve the precision by adding long-term dependency on the model. An event log L_3 is given in the following. By using time consumption as one KPI, if the total sum goes over one threshold, we mark this trace as negative, else as positive. Since the activity *create new topics* usually demands new knowledge rather than checking the existing tools. So if students choose to learn existing tools, it's possibly not useful and time is wasted. In the other case, if we select existing topics with existing background, it saves time when we directly learn the existing tools. According to this performance standard, we classified those event traces.

$$\begin{aligned} L_3 := \{ &< a, b, \mathbf{c1}, d, e, \mathbf{f1}, g1, g2, h, i >^{50,pos}, \\ &< a, b, \mathbf{c2}, d, e, \mathbf{f2}, g2, g1, h, i >^{50,pos}; \\ &< a, b, \mathbf{c1}, d, e, \mathbf{f2}, g2, g1, h, i >^{50,neg}, \\ &< a, b, \mathbf{c2}, d, e, \mathbf{f1}, g1, g2, h, i >^{50,neg} \} \end{aligned}$$

There are no deviations of the model and event log L_3 according to the algorithms in [5] and [4]. Therefore, the original model stays the same and allows for the execution of negative instances. After checking the model and log, those long-term dependencies have significant evidence. Transition ***c1*** decides ***f1*** while ***c2*** decides ***f2***. After addressing long-term dependency like the model M_3 in Figure 1.3b by connecting transitions to extra places, negative instances are blocked and the model has higher precision.

Clearly, the use of negative information can bring significant benefits, e.g, enable a controlled generalization of a process model: the patterns to generalize should never include negative instances. The demand to improve current repair model techniques with incorporating negative instances appears. In the next section, the demand is analyzed and defined in a formal way.

1.2 Research Scope And Questions

After analyzing the current model repair methods, we limit our research scope as shown in Figure 1.4. The inputs for our research are one existing process model M, an event log L . According to predefined KPIs, each trace in event log is classified into positive or negative. After applying repair techniques in the black box, the model should be improved to enforce the positive instances while disallowing negative instance, with condition that the generated model should be as similar to the original model as possible.

In this scope, we come up with several research questions listed in the following.

RQ1: How to overcome the shortcomings of current repair techniques in situations 1-3 above?

RQ2: How to balance the impact of the existing model, negative and positive instances together to repair model?

RQ3: How to block negative instances from the model while enforcing the positive ones?

In the remainder, we propose a solution for the black box. It analyzes process performance on trace level and balances the existing model, positive traces and negative traces on directly-follows relation, in order to incorporate all the factors on model generation.

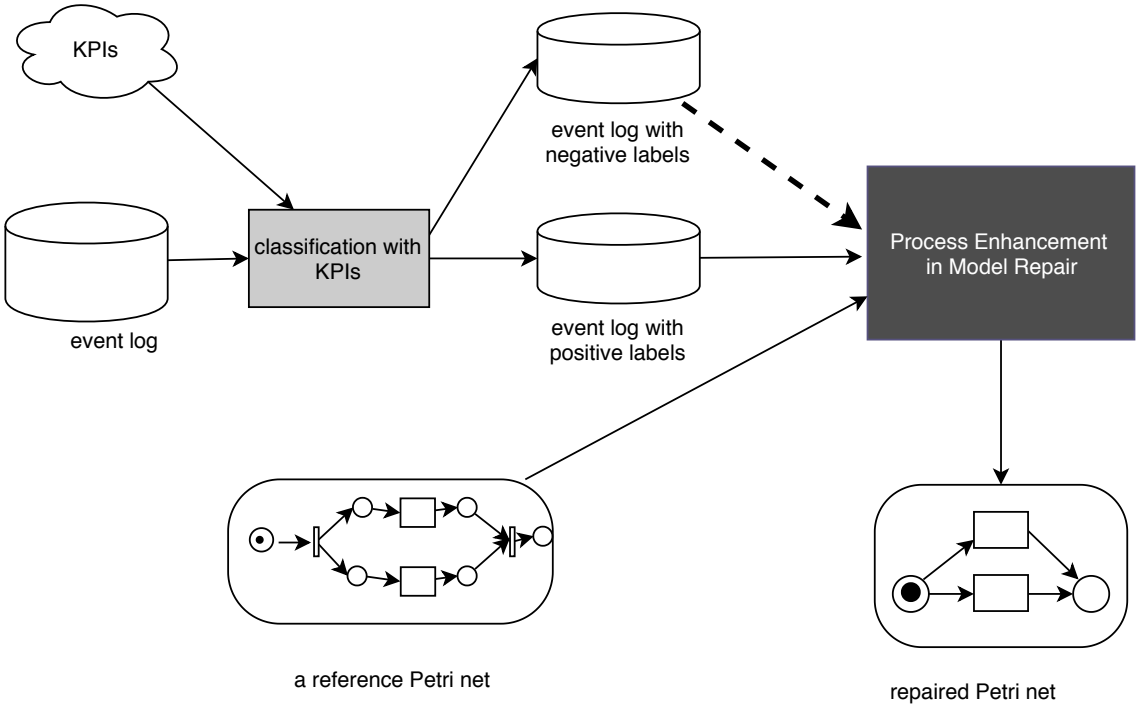


Figure 1.4: The problem description

1.3 Outline

This thesis tries to answer the questions presented in section 1.2 in the remainder chapters and provide a solution for the black box. Chapter 2 and 3 introduces the related work and recalls the basic notions on process mining and list the preliminary to solve the problem. Chapter 4 describes the algorithm to solve the problem. In Implementation part, screenshots are given from the finished tools of our algorithm to demonstrate the use. Experiment chapter answers the last question RQ3, by conducting a bundle of experiments. Later, results are analyzed and discussed. The last chapter is the summary of our work.

Chapter 2

Evaluation

This chapter presents an experimental evaluation of our repair techniques. At first, we define the evaluation criteria. Next, we briefly introduce the test platforms KNIME and relevant ProM plugins tools. Then, we conduct two kinds of tests. One is based one the demo example proposed in the introduction part, one is on the real life data.

2.1 Evaluation Measurements

We evaluate repair techniques based on the quality of repaired models with respect to the given event logs. In process mining, there are four quality dimensions generally used to compare the process models with event logs.

- *fitness.* It quantifies the extent of a model to reproduce the traces recorded in an event log which is used to build the model. Alignment-based fitness computation aligns as many events from trace with the model execution as possible.
- *precision.* It assesses the extent how the discovered model limits the completely unrelated behavior that doesn't show in the event log.
- *generalization.* It addresses the over-fitting problem when a model strictly matches to only seen behavior but is unable to generalize the example behavior seen in the event log.
- *simplicity.* This dimension captures the model complexity. According to Occam's razor principle, the model should be as simple as possible.

The four traditional quality criteria are proposed in semi-positive environment where only positive instances are available. Therefore, when it comes to the model performance, where negative instances are also possible, the measurement metrics should be adjusted. With labeled traces in the event log, the repaired model can be seen as a binary prediction model where the positive instances are supported while the negative ones are rejected. Consequently, the model evaluation becomes a classifier evaluation.

Confusion matrix has a long history to evaluate the performance of a classification model. A confusion matrix is a table with columns to describe the prediction model and rows for actual classification on data. The repaired model can be seen a binary classifier and produces four outcomes- true positive, true negative, false positive and false negative shown in the Table 2.1.

Table 2.1: Confusion Matrix

		repaired model	
		allowed behavior	not allowed behavior
actual data	positive instance	TP	FN
	negative instance	FP	TN

- True Positive(TP): The execution allowed by the process model has an positive performance outcome.
- True Negative(TN): The negative instance is also blocked by the process model.
- False Positive(FP): The execution allowed by the process model has an negative performance outcome.
- False Negative(FN):The negative instance is enabled by the process model.

Various measurements can be derived from confusion matrix. According to our model, we choose the following ones as the potential measurements.

- recall. It represents the true positive rate and is calculated as the number of correct positive predictions divided by the total number of positives.

$$Recall = \frac{TP}{TP + FN}$$

- precision. It describes the ability of the repaired model to produce positive instances.

$$Precision = \frac{TP}{TP + FP}$$

- accuracy. It is the proportion of true result among the total number. It measures in our case how well a model correctly allows the positive instances or disallows the negative instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- F-score is is the harmonic mean of precision and recall.

$$F_1 = \frac{2 * Recall * Precision}{Precision + Recall}$$

Generally, there is a trade-off between the quality criteria. So the measurements are only used to compare specific aspects of our techniques.

2.2 Experiment Platforms

KNIME, as a scientific workflow analytic platform, supports automation of test workflow, which helps us repeat experiments efficiently. Yet, traditional evaluation plugins in ProM are not integrated into KNIME, so partial experiments are conducted in ProM.

2.2.1 KNIME

KNIME supports automation of test workflow mainly through the following mechanisms.

- Loop Control Structure. KNIME provides a bunch of control nodes which support re-executing workflow parts. Nodes representing *Loop Start* appear in pairs with nodes for *Loop Nodes*, the workflow between pairs is executed recursively in a fixed number, or until certain conditions are met. In our test, we repeat our repair techniques for different parameter settings by applying loop structure into KNIME workflow.

- Flow Variables. Flow Variables are used inside a KNIME workflow to parameter node settings dynamically. When it combines with loop control structure, tests with different settings is able to conduct automatically.

What's more, there are nodes provided by KNIME to optimize the value of some parameters with respect to a cost function. As long as a cost function is provides, KNIME is able to automatically optimize any kind of parameters.

2.2.2 ProM Evaluation Plugins

Although KNIME offers a powerful approach to conduct experiments, the integration of traditional process mining evaluation plugins into KNIME is out of our capability due to the time limits. To evaluate repaired models with traditional metrics, we use an existing ProM plugin called ***Multi-perspective Process Explorer*** [6]. This plugin accepts Petri net and an event log as inputs, and gives out fitness, and precision measurements which are based on the traditional alignment conformance checking.

2.3 Experiment Results

2.3.1 Test on Demo Example

In this experiment, we aim to answer the question: Will our repair method overcome shortcomings of current techniques which are shown in the introduction chapter?

2.3.2 Comparison To Other Techniques

This section represents some situations where current repair techniques can't handle properly, while our algorithm gives out an improved repaired model.

Situation 1, unfit part!! added subprocess are too much!! Where the addition of subprocesses and loops are allowed, while the structure changes are impossible, Fahrland's method applies the extension strategy to repair model by adding subprocesses and loops in the procedure. It introduces unseen behavior into the model. However, if the behaviors which are already in the model is unlikely to be removed from the model. One simple example is shown in the following part.

Dee's method is based on Fahrland's method. Deviations are calculated at first and used to build subprocesses for model repair. However, before building subprocesses, it classifies the deviations into positive and negative ones with consideration of trace performance. Only positive deviations are applied to repair model. Different to Fahrland's method, it improves the repaired model performance by limiting the introduced subprocesses. Still, it can't get rid of the defect mentioned before.

Situation 2, For fitted data in the model, can not recognize them!! where overlapped data noise can not be recognized, trace variant with more negative effect is treated as positive and kept in the model, which we should delete them.

Situation 3, with long-term dependency!! fitted part or new added part!! none of the current techniques can handle this problem yet. Simple examples listed, but will this repeat the last section??

For one exclusive choices, but with long-term dependency detected and added in the model, precision and accuracy increase, since model with long-term dependency blocks the negative information by adding transitions and places to limit activity selection.

2.3.3 Test On Real life Data

We choose a publicly available event log from BPI challenge 2015 as our user cases and compare current repair techniques on it.

2.3.3.1 Data Description

The data set for BPI Challenge 2015 contain 5 event logs which are provided by five Dutch municipalities respectively. Those event logs describe the building permit application around four years. We choose it as our user cases due to the following reasons.

- The event logs hold attributes as potential KPIs to classify traces. Attribute **SUMleges** which records the cost of the application is a candidate to label traces as positive or negative if its value is over one threshold. What's more, we can take the throughput time of the application as another potential KPI.
  In a word, this data set provides us information to reasonably label traces.

- The five event logs describe an identical process, but includes deviations caused by the different procedures, regulations in those municipalities. Also, the underlying processes have changes over four years.
  So, this data set gives us a basic process but also allows deviations of the actual event logs and predefined process, which builds the environment for repair techniques.

Firstly, we conduct our experiments on event log called **BPIC15_1.xes.xml**. This event log includes 1199 cases and 52217 events. But the event classes for those events are with the sum of 398. So we preprocess the event log and get a proper subset of data as our user case.

We filter the raw event log by ***Filter Log By Simple Heuristic*** in ProM with the following setting. 40 for the start, end activities and the events between them, at end. We get the event log $D1$. After this, we calculate the throughput time for each trace and add it as a trace attribute **throughput time**. Then we classify traces according to **SUMleges** and **throughput time** separately. When our performance goal is to reduce the cost of application, if **SUMleges** of one trace is over 0.7 of the whole traces, this trace is treated as negative, else as positive. The similar strategy is applied on the attribute **throughput time**. A trace with **throughput time** higher than 0.7 of all traces is considered as a negative instance. Following this preprocess, we have event logs in Table **??** available for our tests.

2.3.3.2 Test Result

Table 2.2: Test Data from Event Log BPI15-1

Data ID	Data Description	Traces Num	Events Num	Event Classes
D1	Heuristic filter with 40	495	9565	20
D2	Apply heuristic filter on D1 with 60	378	4566	12
D3.1	classify on SumLedges; values below 0.7 as positive	349	6744	20
D3.2	classify on SumLedges; values above 0.7 as negative	146	2811	20
D3.3	union of D3.1 and D3.2	495	9596	20
D4.1	classify on throughput time; values below 0.7 as positive	349	6744	20
D4.2	classify on throughput time; values above 0.7 as negative	146	2811	20
D4.3	union of D4.1 and D4.2	495	9596	20

Table 2.3: Test Result on BPI15-M1 data

event log	**reference model**	**method**	**confusion matrix metrics**								**traditional metrics**	
			TP	**FP**	**TN**	**FN**	**recall**	**precision**	**accuracy**	**F1**	**fitness**	**precision**
D1.1	M1	IM	137	48	118	289	0.32	0.74	0.43	0.45	?	?
D1.1	M1	Fahland	0	0								
D1.3	M1	Dees										
D1.3	M1	dfg										
D1.1	M2	IM										
D1.1	M2	Fahland										
D1.3	M2	Dees										
D1.3	M2	dfg										
D1.1	M3	IM										
D1.1	M3	Fahland										
D1.3	M3	Dees										
D1.3	M3	dfg										
D1.1	M4	IM										
D1.1	M4	Fahland										
D1.3	M4	Dees										
D1.3	M4	dfg										
D2.1	M1	IM										
D2.1	M1	Fahland										
D2.3	M1	Dees										
D2.3	M1	dfg										
D2.1	M2	IM										
D2.1	M2	Fahland										
D2.3	M2	Dees										
D2.3	M2	dfg										

Chapter 3

Conclusion

Bibliography

[1] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011.

[2] Dirk Fahland and Wil MP van der Aalst. Repairing process models to reflect reality. In *International Conference on Business Process Management*, pages 229–245. Springer, 2012.

[3] Mahdi Ghasemi and Daniel Amyot. From event logs to goals: a systematic literature review of goal-oriented process mining. *Requirements Engineering*, pages 1–27, 2019.

[4] Marcus Dees, Massimiliano de Leoni, and Felix Mannhardt. Enhancing process models to improve business performance: a methodology and case studies. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 232–251. Springer, 2017.

[5] Dirk Fahland and Wil MP van der Aalst. Model repair—aligning process models to reality. *Information Systems*, 47:220–243, 2015.

[6] Felix Mannhardt, Massimiliano De Leoni, and Hajo A Reijers. The multi-perspective process explorer. *BPM (Demos)*, 1418:130–134, 2015.