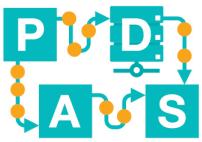


Chair of Process and Data Science

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

At first, I would like to express my deep gratitude to Prof. Wil M.P. van der Aalst for his valuable and constructive suggestions for planning and development of my thesis. Also, the support from Prof. Thomas Rose as my second supervisor on my thesis is greatly appreciated.

For the help given by Dr. Sebastiaan J. van Zelst, I am particularly grateful. His patience guidance and enthusiastic encouragement helped me keep my progress on schedule. Moreover, he kept pushing me into a higher level into scientific research through useful critiques. With those critiques, I realized the limits not only of my methods but also the working strategy, which benefits me a lot in the scientific field.

I also want to thank the whole PADS group and FIT Fraunhofer for their valuable technical support; Especially, the advice from Alessandro Berti has saved me a lot of troubles and improved my work. Finally, I wish to thank my friends and parents for their support and encouragement throughout my study.

Abstract

Based on business execution history recorded in event logs, Process Mining provides visual insight on the business process and supports process analysis and enhancements. It bridges the gap between traditional business process management and advanced data analysis techniques such as 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 logistic time, and production cost in a manufacture. 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.

Contents

List of Figures

List of Tables

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 models with additional data perspectives or repairing the reference models to accurately reflect observed behaviors.

Most of the organizations have predefined process execution rules of activities which are captured in a process model. One execution of related activities in this model is called a trace. A trace which has no deviation to the model is a fitting trace. However, in real life, business processes often encounter exceptional situations where it is necessary to execute process differing from the reference model. With accumulation of deviations, the reference process model needs amending to reflect reality.

Basically, one can apply process discovery techniques again on the event log to obtain a new model only based. This method is referred as process rediscovery. 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 fixes the deviations mainly by adding subprocesses on the model. As known, organizations are goal-oriented and aim to have high performance according to a set of Key Performance Indicator(KPI)s, e.g. the production time for a car industry, the logistic cost for importer companies. However, little research in process mining is conducted on the basis of business performance [3]. The authors of [3] point out several contributions e.g. [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 [2] only with positive deviations which are deviations leading to positive KPI outcomes. This reason behind this approach is 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 the whole event log but overgeneralizes the model, such that more behaviors are allowed than expected. Furthermore, the model complexity is increased by the repair techniques in [2]. Even the performance is considered in [4], but only positive deviations are used to add subprocesses, the negative information is ignored, which disables the possibility to block negative behaviors from model.

In the following sections, 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. To benefit better 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 M_0 of Figure 1.1. Activities are denoted as rectangle and connected by circles called places. Activities are executed following the directed arcs in the Petri net.

M_0 begins with activity **register master**. After executing **register master**, activity **finish courses** can be executed concurrently with the activities branch for writing master thesis on the right set of M0. According to the model, a student can **select existing topics** or **create new topics**. Those two activities are exclusive. The exclusive choice relation also exist in **learn existing tools** or **explore new tools**. Before **Get degree**, **finish courses** and hold a **presentation** are supposed to be completed. For convenience, alphabets are used to represent the corresponding activities

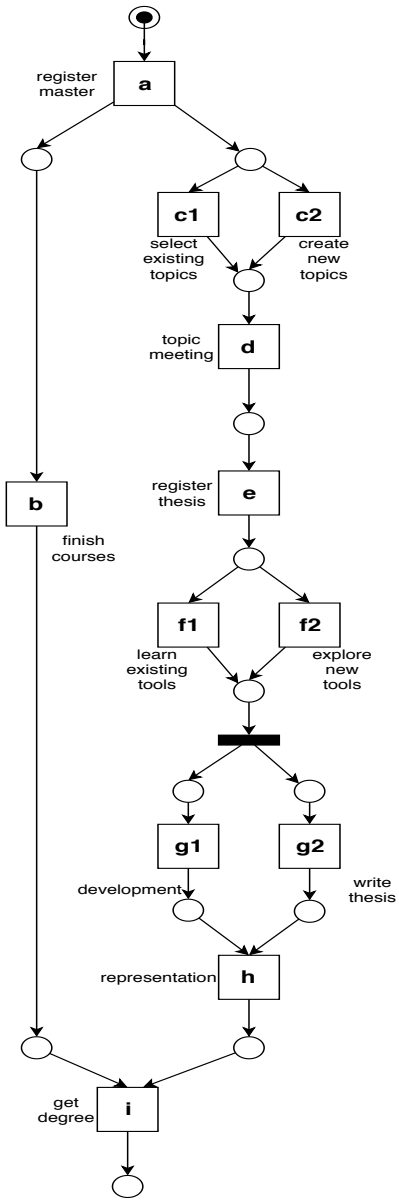


Figure 1.1: original master study process M_0

and annotated in the model.

1.1.1 Situation 1: Add Subprocesses as Loops

One obvious shortcoming of repair techniques in [5] is that subprocesses for deviations are created and added as loops, when the deviations start and end at the same place. If there is only one such subprocess, it is fixed as a sequence in the model with post procedure. Yet, the algorithm does not discover orders between different subprocesses at overlapping locations. Therefore, the subprocesses are kept in a loop form, which causes a lower precision of the repaired model. An example is given with respect to the master study.

In some universities, before registering a master thesis, the activities **write proposal** and **check course requirement** 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. **x1, x2** represent the activities **write proposal** and **check course requirement**.

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

Because the existing repair techniques [5] don't consider the performance of traces in event log, all instances with negative labels are ignored to compute the deviations. L_1 has deviations of **x1, x2** at the same place without order. So the subprocesses for them are added in the loop as shown in Figure 1.2a.

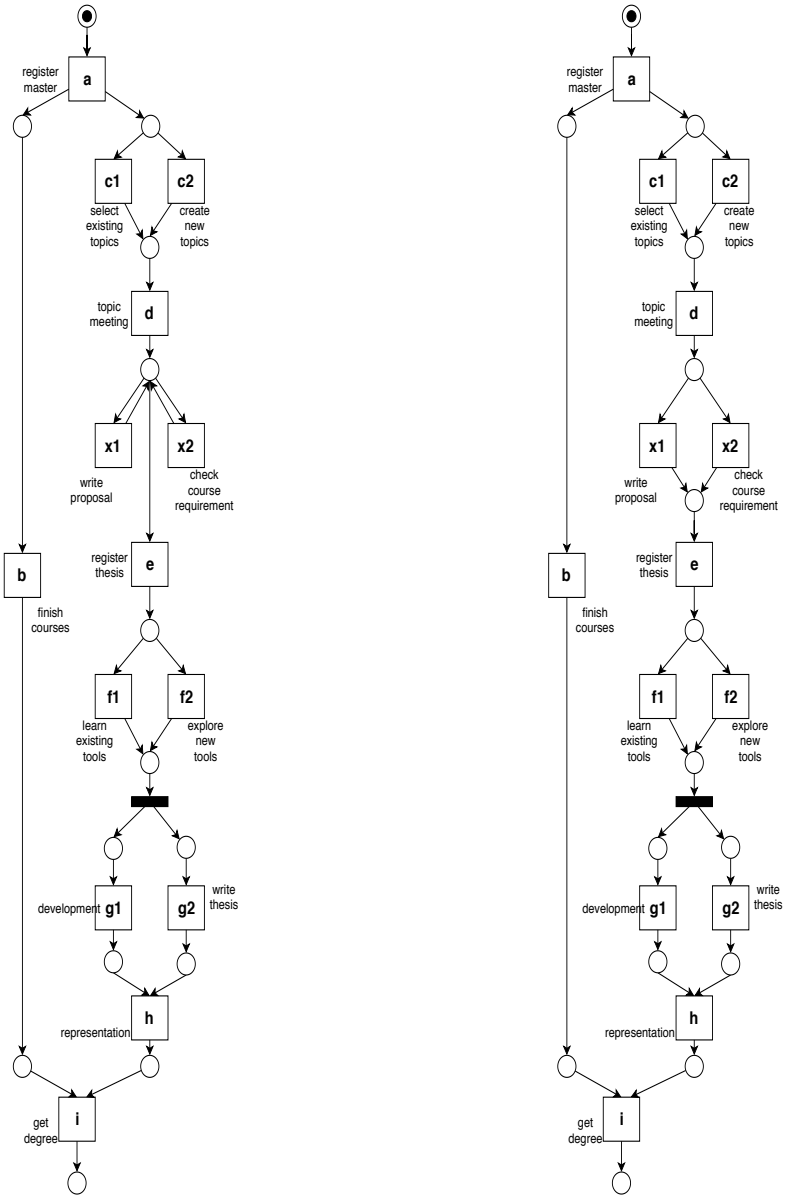
The repair algorithm in [4] builds upon [5] and considers the performance of the event log. However, the repaired model is the same as the one in Figure 1.2a. The reasons are: (1) there is no deviation from negative factors. (2) positive deviations are used to add subprocesses in the same way as [5].

When using rediscovery strategy and apply Inductive Miner on the generated model, it generated model $M_{1.2}$ listed in Figure 1.2b. The main parallel structure has been changed to adapt the deviations.

Compared to the model $M_{1.1}$ and $M_{1.2}$ in Figure 2.1 where the two extra activities are shown in loop, or the main original structure has been changed, we expect the subprocesses with **x1, x2** can be added in sequence without affecting the main structure. In this way, the model has a higher precision of the repaired model, while keeping similar to original one.

1.1.2 Situation 2: Unable to Adapt Model with Fit 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. Traces in such case are



(a) repaired model $M_{1.1}$ with techniques in [5]

(b) expected model $M_{1.2}$

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 $M_{1.2}$.

treated as positive.

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

Compared to M_0, the event log L_2 contains no deviation. When we apply the techniques in [5] and [4] to repair the model, the model remains unchanged. Apparently, the fact that those two methods can't incorporate the negative information in fitting traces causes this shortcoming. A model as M_2 is expected because it enforces the positive instances and avoids the negative instance. Unfortunately, the current methods don't allow us to obtain such results.

1.1.3 Situation 3: Disable to Detect Long-term Dependency

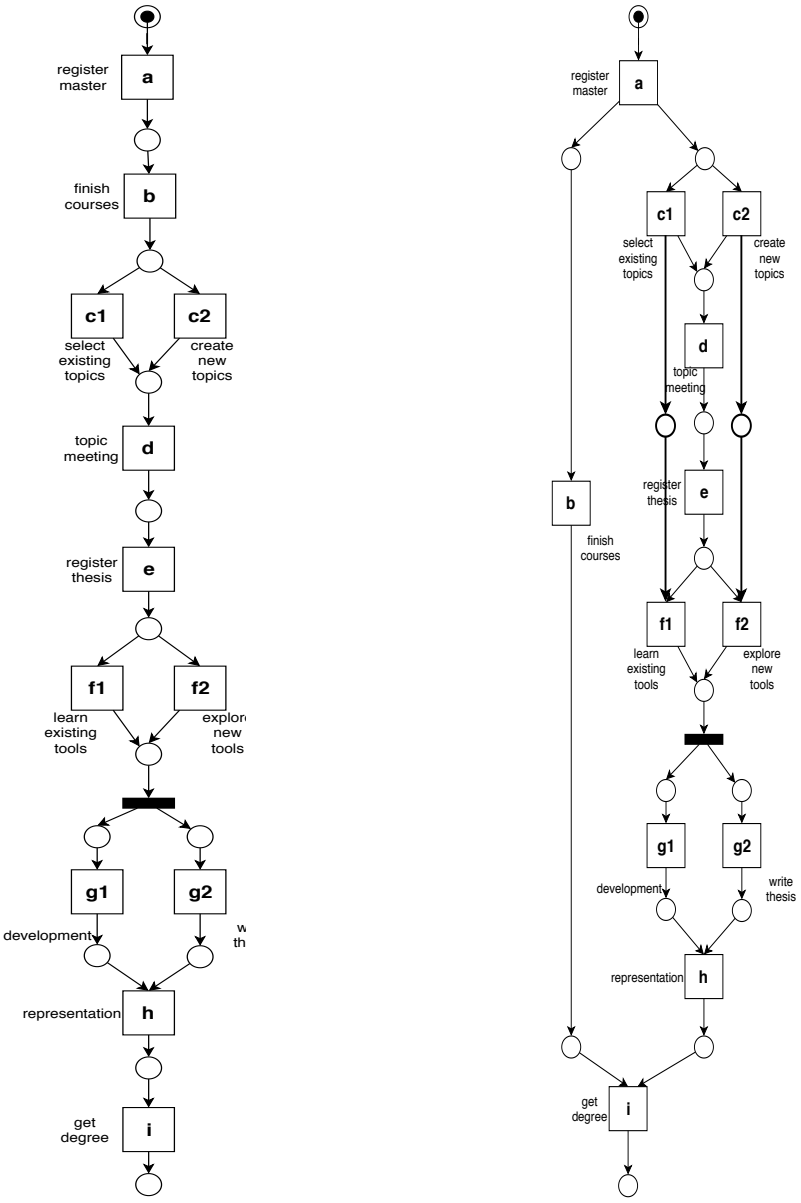
Another problem is that current methods are unable to detect the long-term dependency in the Petri net, which causes a lower precision. The long-term dependency describes the phenomenon that the execution of an activity decides the execution of activities that do not follow directly. Due to the long distance of this dependency, current methods cannot detect it and improve the precision by adding long-term dependency on the model. One example on M_0 is used to demonstrate this problem.

Given the time consumption on the whole study as the KPI, if the total sum goes over one threshold, the trace is negative, else as positive. Since the activity **create new topics** usually demands new knowledge rather than **checking the existing tools**. If students choose to learn existing tools, it's possibly not useful and time-wasting. 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 into positive and negative shown above. An event log L_3 is given in the following.

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

As observed, ***c1*** decides ***f1*** while ***c2*** decides ***f2***. They have long-term dependencies. However, there are no deviations of the model and event log L_3 according to the algorithms in [5] and [4]. Also, the Inductive Miner can't detect long-term dependency. Therefore, the original model stays the same and negative instances can't be blocked.

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. This leads to the demand of improving current repair model techniques with incorporating negative instances. In the next section, the demand is analyzed and defined in a formal way.



(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

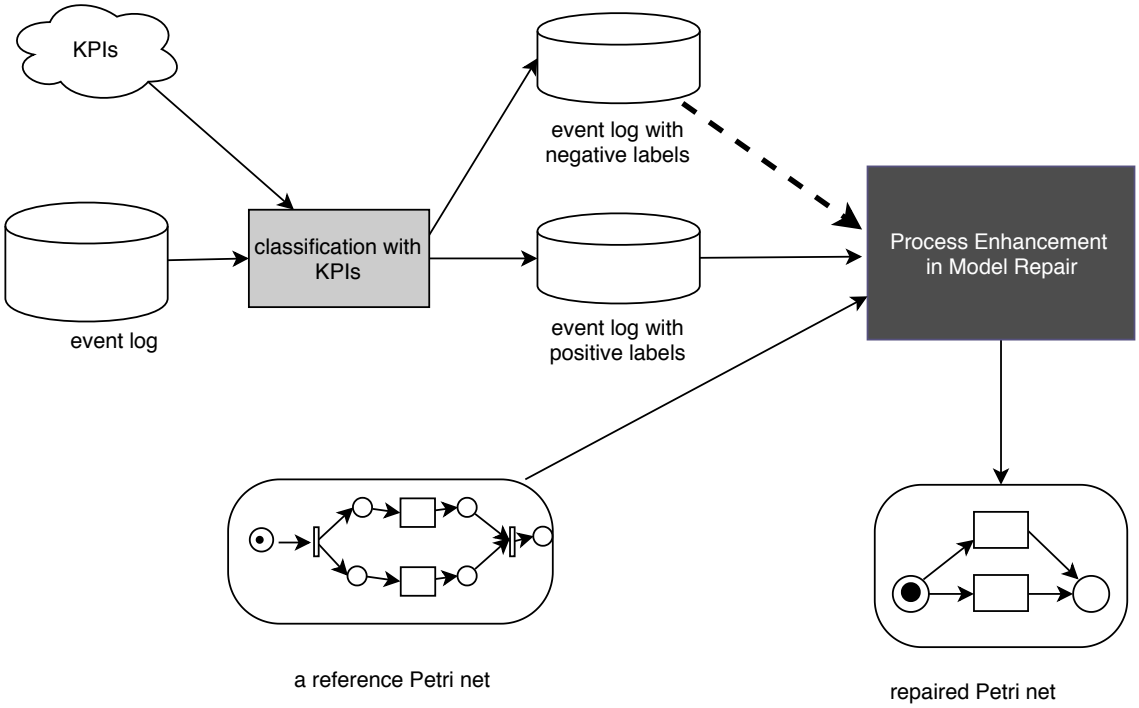


Figure 1.4: The resaerch problem scope

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 this thesis, 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, to incorporate all the factors on model generation.

1.3 Outline

This thesis aims to answer the questions presented in section 1.2 in the remaining chapters and provides a solution for the black box. Chapter 2 introduces the related works. Chapter

3 recalls the basic notions on process mining and list the preliminaries to solve the problem. Chapter 4 firstly presents an original framework to incorporate negative information into model repair. Based on this framework, a concrete algorithm are proposed. In Chapter 5, screenshots of the implementation tools are shown to demonstrate the usage. Chapter 6 answers the last question RQ3, by conducting a bundle of experiments. Later, results are analyzed and discussed. At last, we summarize our work in Chapter 7.

Chapter 2

Evaluation

In this chapter, we evaluate our repair techniques based on the quality of repaired model. 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

Table 2.1: Confusion Matrix

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

produces four outcomes- true positive, true negative, false positive and false negative shown in the Table 2.1.

- 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.

Furthermore, 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 complete the experiments, the following plugins in [1] are in need.

- Model Repair. This plugin is developed on the work in [2, 5]. It repairs a Petri net according to an event log.
- PNetReplayer. It checks conformance of the process model in Petri net with an event log.

2.3 Experiment Results

We conduct our experiments into two main parts, one is to verify if our method overcomes the limits of current repair algorithms and provides a repaired model like expected. This experiment is based on the synthetic data and models from Introduction chapter. Next, real life data is used to test our method.

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?

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.

[1] ProM`http://www.promtools.org`

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.

2.3.2 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.

2.3.2.1 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

Table 2.2: Test event log from real life data 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

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 2.2 available for our tests.

Table 2.3: Generated reference models for test

Model ID	**Used Data**	**Setting**	**Event Class**	**CM Evaluation**								
				Data	TP	FP	TN	FN	recall	precision	accuracy	F1
M1	D1	IM-infrequent: Noise Setting: 20	20	D3.3	112	40	106	237	0.321	0.737	0.440	0.447
				D4.3	131	21	128	215	0.379	0.862	0.523	0.526
M2	D1	IM-infrequent: Noise Setting: 50	20	D3.3	106	39	107	243	0.304	0.731	0.430	0.429
				D4.3	125	20	129	221	0.361	0.862	0.513	0.509
M3	D2	IM-infrequent: Noise Setting: 20	12	D3.3	0	0	146	349	0	NaN	0.295	0
				D4.3	0	0	149	346	0	NaN	0.301	0
M4	D2	IM-infrequent: Noise Setting: 50	12	D3.3	0	0	146	349	0	NaN	0.295	0
				D4.3	0	0	149	346	0	NaN	0.301	0

Based on the filtered data, we derive corresponding Petri nets as reference process models. The Table 2.3 lists the models with different setting. **IM-infrequent** is one variant of Inductive Miner working on event logs with infrequent traces. **Noise** is set as the threshold to filter out infrequent traces. After mining a reference model, we compare them with corresponding event logs to get the basis lines for later evaluation.

As seen in table above, the reference models don't apply well to the corresponding event logs. So changes on the models are in demand, to reflect better the reality and also to enforce the positive instances and avoid negative instances.

2.3.2.2 Test Result

We conduct experiments in the following types.

- **Type 1** Inductive Miner only on the positive event log to discover a model. The default setting with infrequent variant and noise threshold as 20 is chosen. Later, the mined model is checked on the labeled event with positive and negative instances. This method is abbreviated as IM.
- **Type 2** Repair Model from [5] is applied on the positive event log to discover a model. The default setting is chosen. Later, the mined model is checked on the labeled event with positive and negative instances. This method is abbreviated as Fahland, named after the name of main author.
- **Type 3** The method proposed from our thesis, is applied on the labeled event log with positive and negative instances. Default setting for the control parameters is 1.0 while the parameters to generate Petri nets from directly-follows graph are set as the same as experiment Type 1. Later, the repaired model is evaluated on the labeled data. This method is abbreviated as Dfg.

Those types are applied on pairs of event log groups of D3.1,D3.2,D3.3 and D4.1,D4.2,D4.3 in Table 2.2 and models from Table 2.3. The experiment result is shown in the Table 2.4.

Table 2.4: Test Result on BPI15-M1 data

event log	**reference model**	**method**	**confusion matrix metrics**							
			TP	**FP**	**TN**	**FN**	**recall**	**precision**	**accuracy**	**F1**
D3.1	M1,M2,M3,M4	IM	137	48	118	289	0.32	0.74	0.43	0.45
D3.1	M1	Fahland	343	136	10	6	0.983	0.716	0.713	0.829
D3.3	M1	dfg	124	52	94	225	0.355	0.705	0.44	0.472
D3.1	M2	Fahland	317	133	13	32	0.908	0.704	0.667	0.793
D3.3	M2	dfg	124	52	94	225	0.355	0.705	0.44	0.472
D3.1	M3	Fahland	349	145	1	0	1.0	0.706	0.707	0.828
D3.3	M3	dfg	0	0	349	146	0	NaN	0.295	0
D3.1	M4	Fahland	271	107	77	39	0.779	0.718	0.628	0.747
D3.3	M4	dfg	0	0	349	146	0	NaN	0.295	0
D4.1	M1	IM	131	21	128	215	0.379	0.862	0.523	0.526
D4.1	M1	Fahland	325	133	16	21	0.939	0.710	0.689	0.808
D4.3	M1	dfg	139	36	113	207	0.402	0.794	0.509	0.534
D4.1	M2	Fahland	325	130	19	21	0.939	0.714	0.695	0.811
D4.3	M2	dfg	139	36	113	207	0.402	0.794	0.509	0.534
D4.1	M3	Fahland	87	29	120	259	0.251	0.75	0.418	0.377
D4.3	M3	dfg	0	0	346	149	0	NaN	0.303	0
D4.1	M4	Fahland	63	20	129	283	0.182	0.759	0.388	0.294
D4.3	M4	dfg	0	0	346	149	0	NaN	0.303	0

With the similar measurements, it is observed that the repaired models from **Type 2** are more complex that Dfg method. One example is given for the tests of M1 on event log D4.1 for **Type 2** and M1 on event log D4.3.

Due to the different settings in our method, forces from the reference model, positive, and negative event logs are balanced differently during repair, which results in different process

models. To investigate the effect of those setting on the repaired model, we repeat our experiments on the following setting.

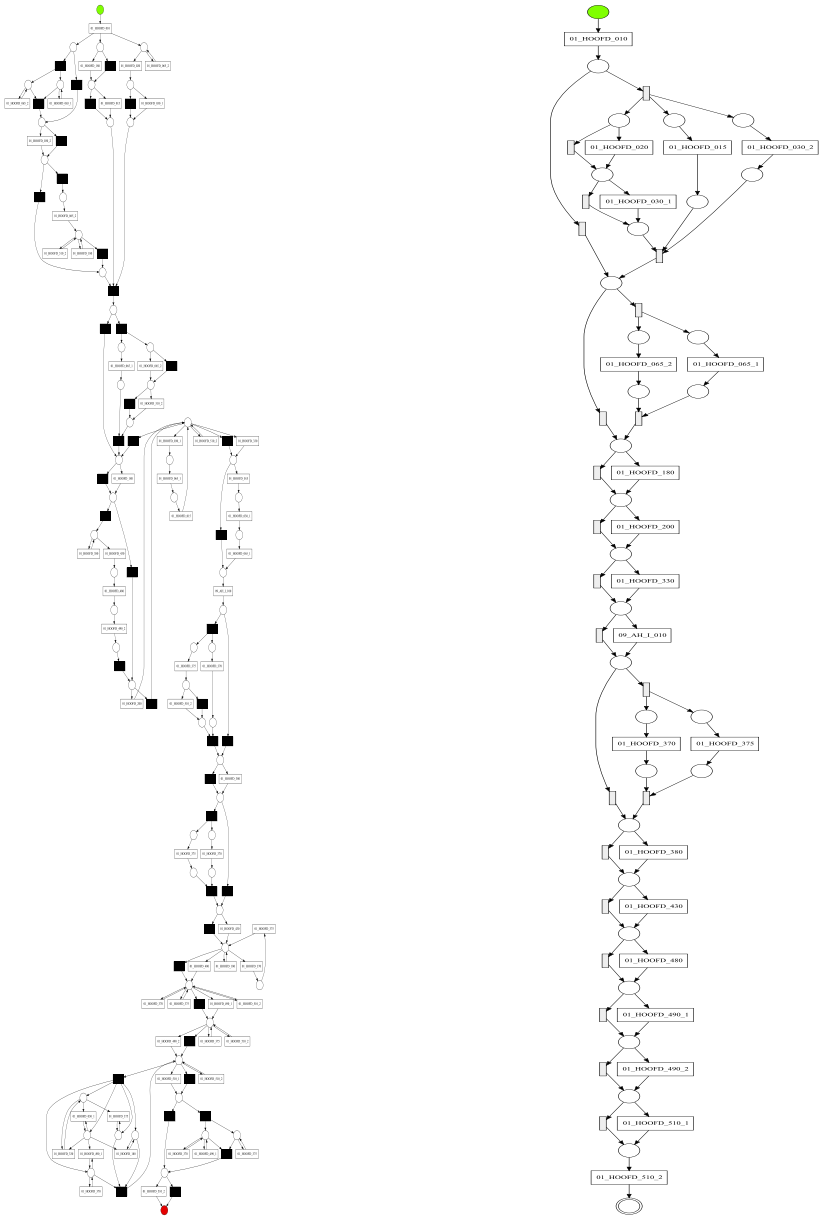
Each of three control parameters for the existing model, positive and negative instances changes value from 0.0 to 1.0 with step 0.1. With this setting, directly-follows relation is generated. Afterward, the default setting of Inductive Miner Infrequent with noise threshold 20 is used to mine Petri nets from the generated directly-follows graph.

So in total, we conduct over thousand experiments to investigate our methods. Based on those results, we draw plots to show the tendency of evaluation results on the parameters for the existing model, positive and negative event logs.

From the Figure 2.2, with the parameter for the existing model going up, recall goes up while accuracy and precision goes down. The reason behind it is possibly ????.

Figure 2.3 shows that if the parameter for negative event log increases, precision and accuracy go up. By addressing negative force, our techniques tend to block behavior which leads to low performance output. However, if the negative force is over the force from positive event log and the existing model, certain behavior which contributes to positive performance will also be deleted from the models. In contrast, this creates a model with less recall.

Figure 2.4 displays the tendency with the parameter for positive event log. When the positive parameter rises, the recall increases. Precision and accuracy also increases but with ???.



(a) repaired model with techniques in [5]

(b) repaired model with

Figure 2.1: 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 $M_{1.2}$.

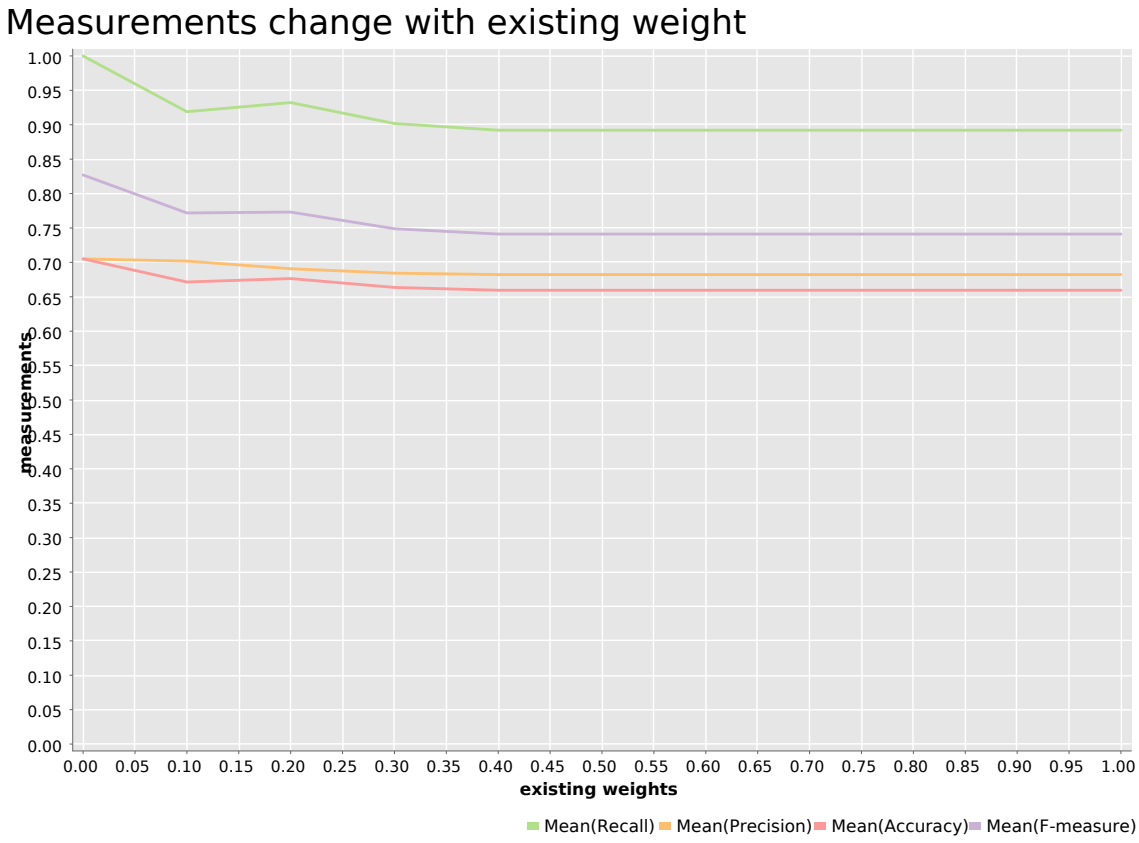


Figure 2.2: result with control parameter for existing model on event log D3.3 and model M3

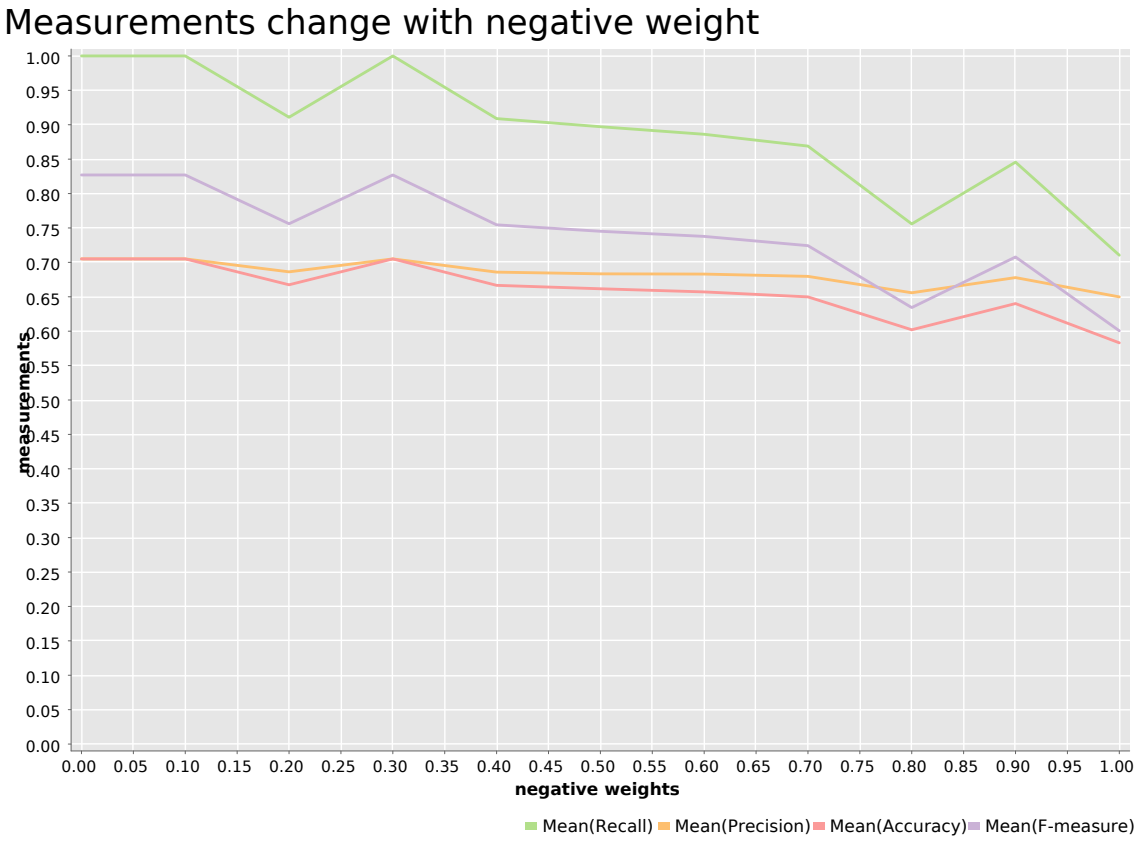


Figure 2.3: result with control parameter for negative instance on event log D3.3 and model M3

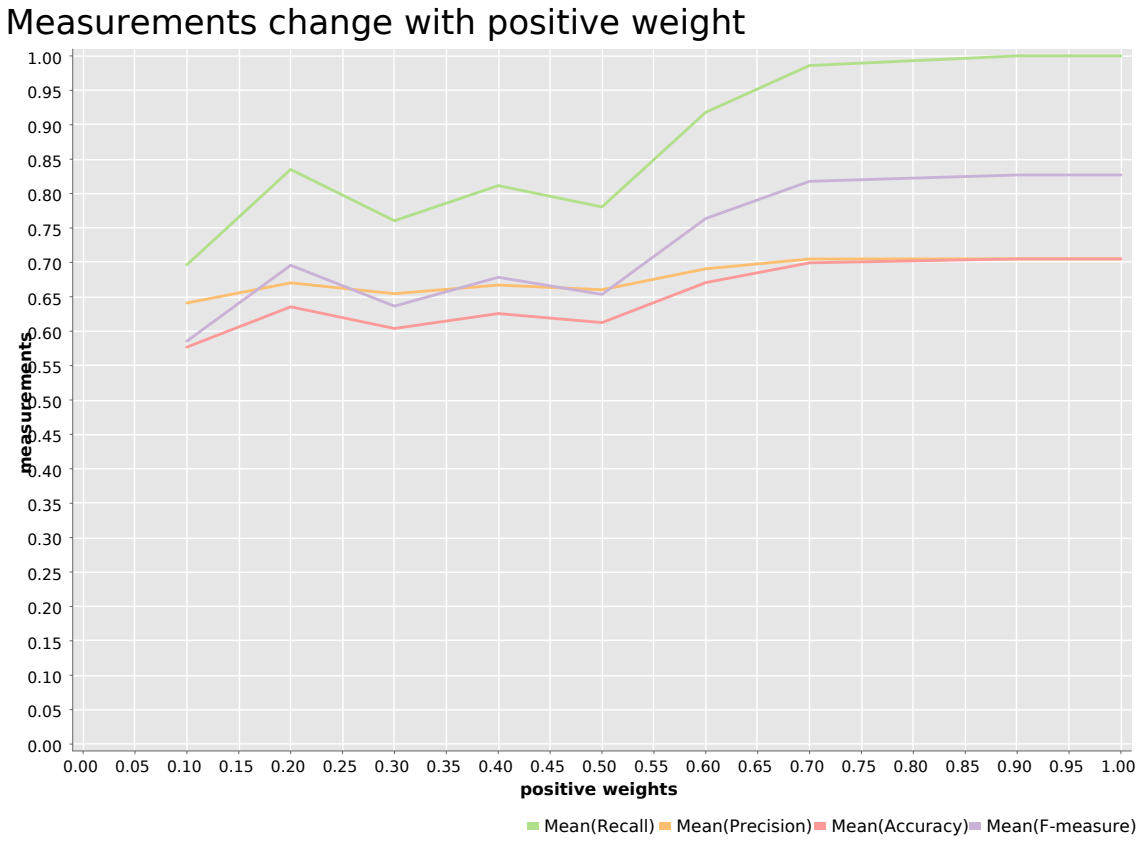


Figure 2.4: result with control parameter for positive instance on event log D3.3 and model M3

Chapter 3

Conclusion

In this thesis, we explore ways to use negative information in model repair and propose our innovative method. Firstly, we analyzed the current techniques on model repairs based on performance and detect their shortcomings. Then we proposed a general framework to incorporate the forces from the existing model, positive and negative event logs. Three abstraction data models in the same type are built to represent those forces. Later, forces are balanced based on data models and expressed in a new data model. From this new data model, process models are discovered and converted into repaired models with the required type. Optional post processes include long-term dependency detection and silent transition reduction, which further improves the repaired model.

Moreover, we demonstrate the usage of our method by conducting experiments with real life data. Our method is able to provide better repaired models than other repair methods in some situations. Also, with respect to other methods, it runs faster and generates simpler models.

Future work might be to improve the rules of balancing different forces, which choose the directly-follows relation on the simple subtraction and sum of those forces. Advanced data mining techniques such as association rules discovery, and Inductive Logistic Programming can be used on those forces to derive rules for building a process model. The same improvement can be applied on the long-term dependency discovery. Moreover, in this implementation, the long-term dependency discovery is restricted on the activities with exclusive choices relation. Later, we should extend the long-term discovery on other possible relations, like parallel relation. Also, we can drop the process tree as our intermediate result and adopt it directly on the Petri net.

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.