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

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 the initial places and represent the dynamic state of the model.

M_0 is currently in an initial state where only 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 a unfitting trace. With accumulation of deviations, process model needs

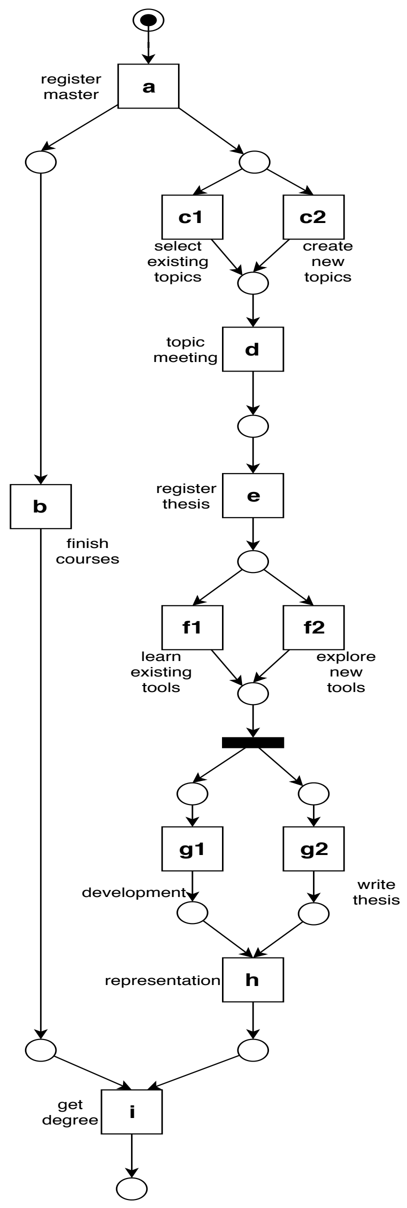


Figure 1.1: original master study process M_0

amending to reflect reality. In the following part, different
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. Event Log* L_1 –

$$\begin{aligned} Positive : \{ &< a, b, c1, d, \mathbf{x1}, f, g1, g2, h, i >^{50}, \\ &< a, b, c2, d, \mathbf{x2}, f, g2, g1, h, i >^{50} \} \end{aligned}$$

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.

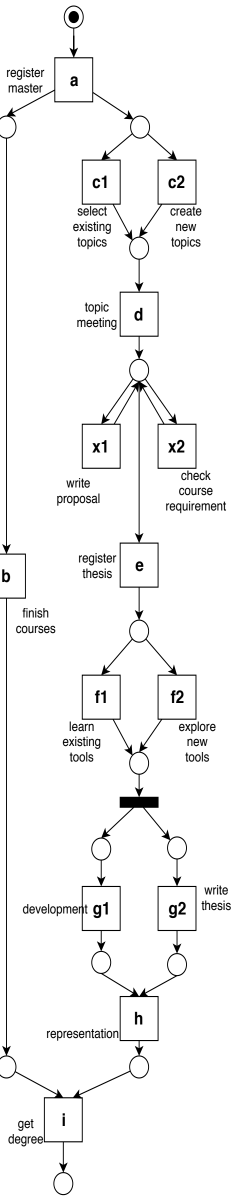
The repaired model is shown in Figure 1.2a, where the two additional activities are added in the form of loop. The repair algorithm in [4] builds upon [5] and considers the performance of the event log. However, the repaired model is the same as the one in Figure 1.2a. The reasons are: (1) there is no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

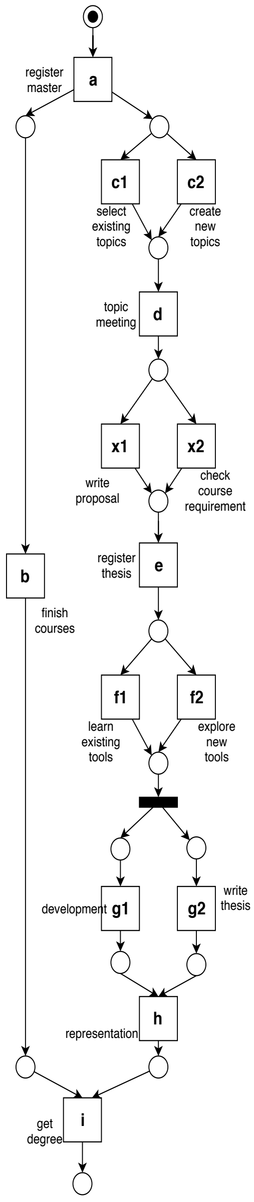
1.1.2 Situation 2: Repairing A Model with Fitting Traces

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. *Event Log* L_2 –

$$\begin{aligned} Positive : \{ &< a, \mathbf{b}, c1, d, f, g2, g1, h, i >^{50}, \\ &< a, \mathbf{b}, c2, d, f, g1, g2, h >^{50} \} \\ Negative : \{ &< a, c1, d, f, g2, g1, \mathbf{b}, h, i >^{50}, \\ &< a, c1, \mathbf{b}, d, f, g1, g2, h, i >^{50}, \} \end{aligned}$$



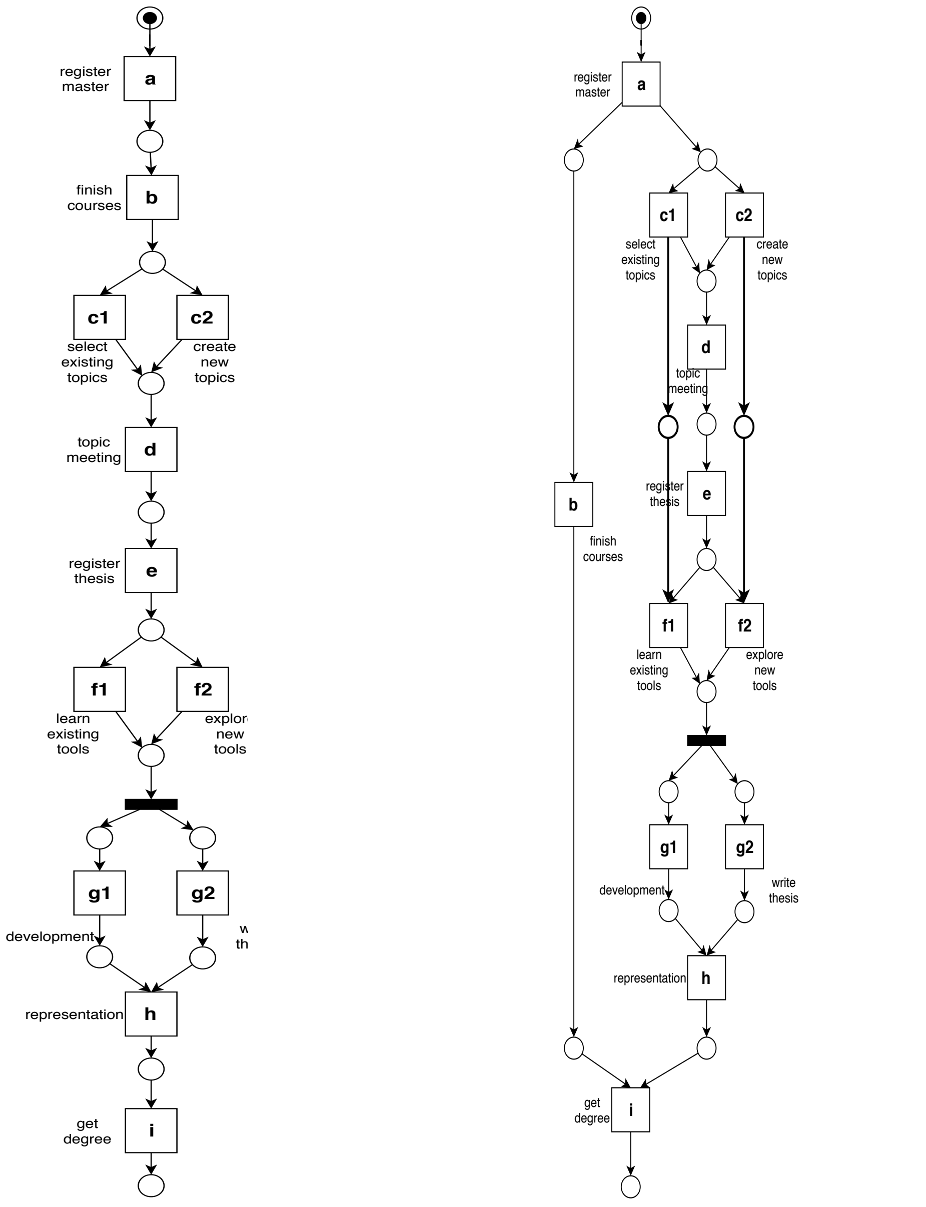
(a) repaired model $M_{1.1}$ with additional activitiesties



(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 $M_{1.2}$.

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. 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. *Event Log* L_3 –

$$\begin{aligned}
Positive : \{&< a, b, \mathbf{c1}, d, e, \mathbf{f1}, g1, g2, h, i >^{50}, \\
&< a, b, \mathbf{c2}, d, e, \mathbf{f2}, g2, g1, h, i >^{50}\} \\
Negative : \{&< a, b, \mathbf{c1}, d, e, \mathbf{f2}, g2, g1, h, i >^{50}, \\
&< a, b, \mathbf{c2}, d, e, \mathbf{f1}, g1, g2, h, i >^{50}\}
\end{aligned}$$

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?

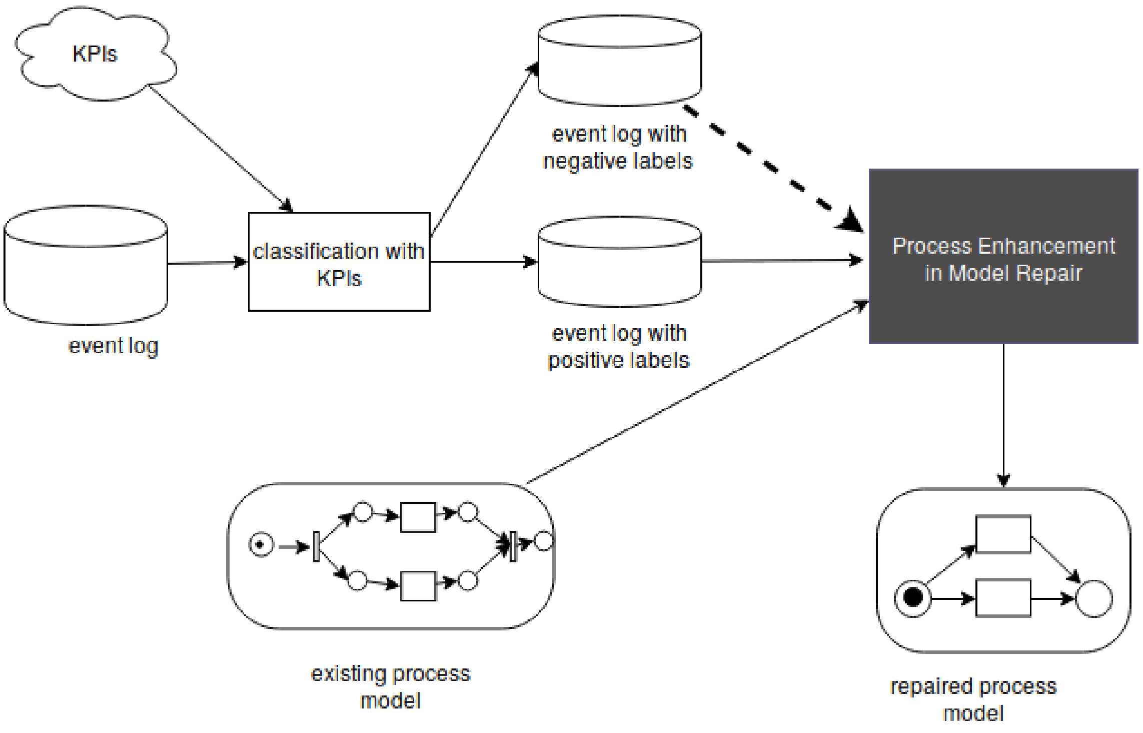

KPIs
event log
classification with KPIs
event log with negative labels
event log with positive labels
Process Enhancement in Model Repair
existing process model
repaired process model

Figure 1.4: The problem description

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.

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

Related Work

To update an existing process model in organizations, there are two strategies, rediscovery and process enhancement. Process rediscovery applies the discovery techniques on the actual event log to mine a new model. Process enhancement improves the model based on not only the actual event log but also the existing model.

Process discovery has been intensively researched in the past two decades and many algorithms have been proposed[6]. Directly-follows[7, 8] methods investigate the activities order in the traces and extracts higher relations which are used to build process models. State-based methods like [9, 10] builds a transition system to describe the event log, and then group the state regions into corresponding Petri net node. Language-based algorithms uses integer linear system to represent the place constraint where the token at one place can never go negative. By solving the system, a petri net is created. Its representative techniques are Integer Linear Programming(ILP) Miner[11]. Other methods due to [12] include search-based algorithm like Genetic Algorithm Miner[13], heuristic-based algorithm Heuristics Miner[14].

Among those discovery methods, Inductive Miner is widely applied[8]. It investigates the activity order in the traces and represents the order in a directly-follows graph. Based on the graph, it finds the most prominent split from the set of exclusive choice, sequence, parallelism, and loop splits on the event log. Afterward, the corresponding operator to the split is used to build a block-structured process model called a process tree. Iteratively, the split sublogs are passed as inputs for the same procedure until one single activity is reached and no split is available. A process tree is output as the mined process model and can be converted into another process model called Petri net.

When the actual event log differs a lot from the referred process model, it is suitable to use the rediscovery method to improve the business execution. However, in some cases, the process enhancement focuses to extend or improve an existing process model by using an actual event log[1]. Besides extending the model with more data perspectives, model repair is another type of enhancement. It modifies the model to reflect observed behavior while keeping the model as similar as possible to the original model.

In [2], model repair is firstly introduced into process enhancement. By using conformance checking, the deviations of the event log and process model are detected. The consecutive deviations in log only are collected in the form of subtraces at a specific location Q in the model. Later, the subtraces are grouped into sublog that share the same location Q for subprocess discovery. In the earlier version in [2], the sublogs are obtained in a greedy way, while in [5], sublogs are gathered by using ILP Miner to guarantee the fitness. Additional subprocesses are introduced into the existing model to ensure that all traces fit the model, but it introduces more behavior into the model and lowers the precision.

Later, compared to [2, 5], where all deviations are incorporated in model repair, [4] considers the impact of negative information. In [4], the deviations of the model and event log are firstly analyzed, in order to find out which deviations enforces the positive performance. Given a trace and a selected KPI, an observation instance is built to correlate the number of each log move with KPI output. Based on the observation instance, a set of rules are derived in the form of a decision tree. According to the rules, the original event log is divided into sublogs with traces matching the rules. The sublogs are then repaired to contain only trace deviations which have a positive KPI output. Following repair, the sublogs are merged as the input for model repair in [5]. According to the study case in [4], it provides a better result than [5] on the aspect of performance.

As described above, the state-of-the-art repair techniques are based on positive instances, meanwhile, the negative information is neglected. Without negative information, it is difficult to balance the fitness and precision of those model. Likewise, little research gives a try to incorporate negative information in multiple forms on process discovery.

In [15], the negative information is artificially generated by analyzing the available events set before and after one position and represented in the form of the complement of positive event sets. Based on the positive and negative event sets, Inductive Logic Programming is applied to detect the preconditions for each activity. Those preconditions are then converted to Petri net after applying a pruning and post-process step. Similar work on model discovery based on artificial negative events are published later. In [16], the author improves the method in [15] by assigning weights on artificial events with respect to an unmatching window, in order to offer generalization on the model.

The work in [17] uses traces in the event log with negative outcomes as negative information. It extends the techniques of numerical abstract domains and Satisfiability Modulo Theories(SMT) proposed in [18] to incorporate negative information for model discovery. Each trace as positive or negative is transformed as one point in n-dimensional space, n is the number of distinct activities. The execution of a trace reflects the token transmission and marking limits on places in the model. Those limits are represented into a set of marking inequalities and in a form of convex polyhedron in n-dimensional space. Given half-space hypotheses, SMT solves the inequalities and gives the limits on the process model. Before SMT, negative information is incorporated to shift and rotate the polyhedron, which limits the generalization of the solution space. Because half-space is used, this method can not deal with negative instances overlapped into positive instances.

However, the field of model repair which considers the negative information is new. Furthermore, the idea to incorporate negative instances on trace level into model repair is innovative.

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] Wil Van der Aalst. Data science in action. In *Process Mining*, pages 3–23. Springer, 2016.

[7] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[8] Sander JJ Leemans, Dirk Fahland, and Wil MP 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*, pages 311–329. Springer, 2013.

[9] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer, 2007.

[10] Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alex Yakovlev. Synthesizing petri nets from state-based models. In *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, pages 164–171. IEEE, 1995.

[11] Jan Martijn EM Van der Werf, Boudewijn F van Dongen, Cor AJ Hurkens, and Alexander Serebrenik. Process discovery using integer linear programming. In *International conference on applications and theory of petri nets*, pages 368–387. Springer, 2008.

[12] Boudewijn F Van Dongen, AK Alves De Medeiros, and Lijie Wen. Process mining: Overview and outlook of petri net discovery algorithms. In *Transactions on Petri Nets and Other Models of Concurrency II*, pages 225–242. Springer, 2009.

[13] Ana Karla A de Medeiros, Anton JMM Weijters, and Wil MP van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[14] Anton JMM Weijters and Wil MP Van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

[15] Stijn Goedertier, David Martens, Jan Vanthienen, and Bart Baesens. Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10 (Jun):1305–1340, 2009.

[16] Seppe KLM vanden Broucke, Jochen De Weerdt, Jan Vanthienen, and Bart Baesens. Determining process model precision and generalization with weighted artificial negative events. *IEEE Transactions on Knowledge and Data Engineering*, 26(8): 1877–1889, 2014.

[17] Hernan Ponce-de León, Josep Carmona, and Seppe KLM vanden Broucke. Incorporating negative information in process discovery. In *International Conference on Business Process Management*, pages 126–143. Springer, 2016.

[18] Josep Carmona and Jordi Cortadella. Process discovery algorithms using numerical abstract domains. *IEEE Transactions on Knowledge and Data Engineering*, 26(12): 3064–3076, 2014.