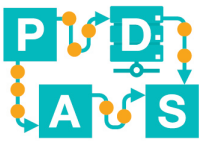


RWTH AACHEN UNIVERSITY

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

Big data projects have becomes a normal part of doing business, which raises the interest and application of process mining in organizations. Process mining combines data analysis with modeling, controlling and improving business processes, such that it bridges the gap of data mining on big data and business process management.

Process enhancement, as one of the main focuses in process mining, improves the existing processes according to actual execution event logs . It enables continuous improvement on business performance in organizations. However, most of the enhancement techniques only consider the positive instances which are execution sequences but lead to high business performance outcome. Therefore, the improved models tend to have a bias without the use of negative instances.

This thesis provides a novel strategy to incorporate negative information on process enhancement. Firstly, the directly-follows relations of business activities are extracted from the given existing reference process model, positive and negative instances of actual event log. Next, those relations are balanced and transformed into process model of Petri net by Inductive Miner. At end, long-term dependency on Petri net is further analyzed and added to block negative instances on the execution, in order to provide a preciser model.

Experiments for our implementation are conducted into scientific platform of KNIME. The results show the ability of our methods to provide better model with comparison to selected process enhancement techniques.

Chapter 1

Introduction

Process mining is a relatively new discipline that has emerged from the need to bridge the gap of data mining and business process management. The objective of process mining is to support the analysis of business process, provide valuable insights on processes and further improve the business execution. According to [15], techniques of process mining are divided into three categories: process discovery, conformance checking and process enhancement. Process discovery techniques focus on deriving process models from event logs of the information system, allowing the vision into the real business process. Conformance checking analyzes the deviations between an referenced process model and observed behaviors driven from its execution. 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.

Due to the increasing availability of detailed event logs of information systems, process mining techniques have recently enabled wider applications of process mining in organizations around the world[15]. After applying process discovery in organizations, a process model is fixed in information system to guide the execution of business. However,in real life, business processes often encounter exceptional situations where it is necessary to execute process differing from the predefined model. To reflect the reality, the organizations need to adapt the existing process model. Basically, one can apply process discovery techniques again to obtain a new model from event log. However, due to the facts, (1) the cost of rediscovery, and (2) the discovered model tend to have less similarity with the original model[8]. As shown in [8], there is a need to change an existing model similar to the original model while replaying the current process execution. Here comes the model repair.

Model repair belongs to process enhancement and stands between process discovery and conformance checking. It analyzes the workflow deviations between event log and process model, and fix the deviations mainly by adding sub processes on the model. As known, business in organizations is goal-oriented and aims to have high performance according to a set of Key Performance Indicators(KPIs), for example, average conversion time for the sales, payment error rate for the finance. However, there are few researches on applying the process mining with consideration of performance[10]. [10] points out the rare contributions like [6] to combine performance into process mining. Deviations are firstly analyzed to determine if they have a positive impact on the process performance. Model repair techniques in [9] are applied into traces with positive deviations.

However, the current repair methods have some limits. Model repair fixes the model

by adding subprocesses, silent transitions or loops, it guarantees the model fitness but overgeneralizes the model, such that it allows more behaviors than expected. On the other hand, it increases the model complexity. Even the performance is considered in [6], 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. A motivation example is listed to describe those limits.

1.1 Motivation Example

This section describes some situations where current repair techniques can't handle properly. For the sake of understanding, some examples are extracted from the registration procedure of thesis project at one German university to illustrate those situations.

The main activities for the registration process include topic selection, make proposal, meeting with supervisor to discuss the topic, and finish course requirements. After finishing all of those activities, the formal registration is enabled and the procedure comes to end. Figure 1.1 shows the original process in Petri net. 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. ***Tokens*** in the black dot are put in the initial places and represent the dynamic state of the model.

The model in Figure 1.1 is in an initial state where only one token is at the start place to enable the first transition begin thesis. After firing *begin thesis*, the token at the initial place is consumed while two new token are generated in the output places. In this way, activity *finish course requirements* can be executed in concurrency with the other activities except for the *register thesis*. 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 have multiple input places, it can be triggered with condition that all input places hold at least a token. *Register thesis* is enabled only after *finish course requirements* and getting the *approve* done.

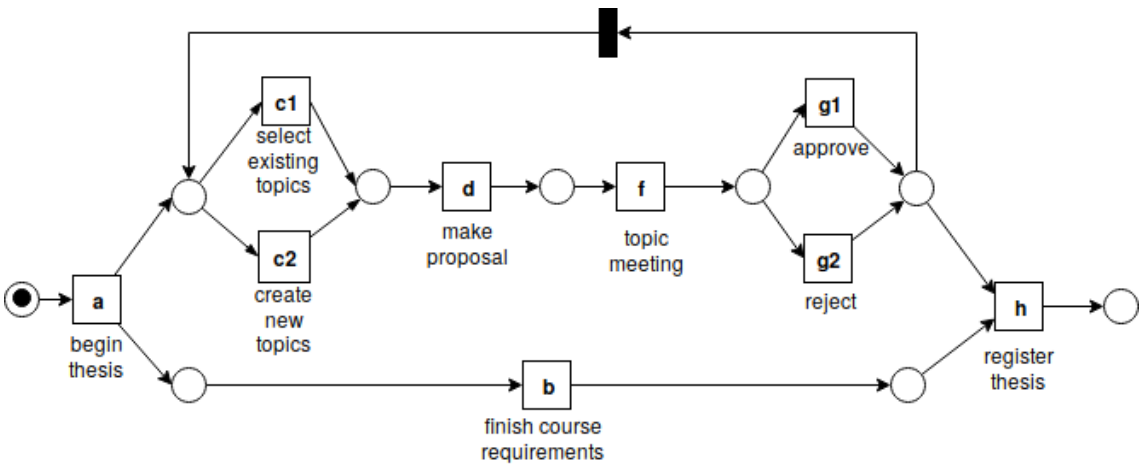


Figure 1.1: original registration process

In the following part, those situations are introduced to demonstrate the shortcomings of current techniques.

1.1.1 Situation 1: repair model with unfit traces

Following the given model, in order to address the work between *make proposal* and before the activity *topic meeting*, two exclusive activities *prepara carefully* and *prepara casually* are executed in the real life and constitutes the actual event log L_1 listed below. With those two activities, it makes the whole procedure more efficient and clear and the traces with them are considered positive. For convenience, alphabet characters are used to represent the corresponding activities and annotated in the model. **e1, e2** represent the activities *prepare carefully* and *prepare casually.*
Event Log L_1 –

$$\begin{aligned} Positive : \{ &< a, b, c1, d, \mathbf{e1}, f, g1, h >^{50}, \\ &< a, b, c2, d, \mathbf{e2}, f, g1, h >^{50} \} \end{aligned}$$

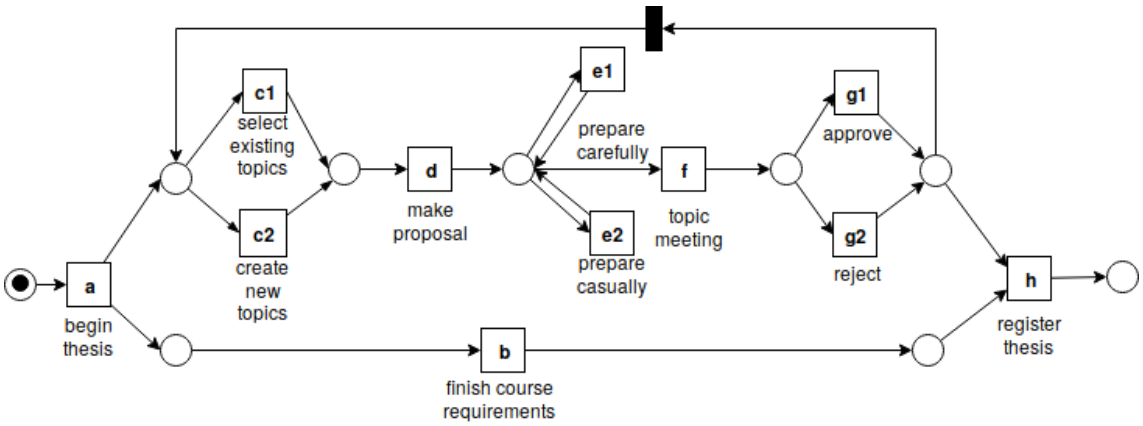
Because the repair techniques in [9] don't distinguish the performance of event log, to make the model enforcing the positive performance, only positive instances are employed to repair the model. Firstly, the deviations of the existing model in Figure 1.1 and the event log L_1 is computed. After computation of deviations, each deviation has the same start and end place and two deviations appear at the same position of 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 will be added in an sequence in the model. This leads to a higher precision. Yet the algorithm does not discover orderings between different subprocesses at overlapping locations. So subprocesses are kept in loop form.

The repaired model is shown in Figure 1.2a, where the two additional activities are added in the form of loop. Compared to the model in Figure 1.2b where the two extra activities are shown in sequence with others, the repaired model in Figure 1.2a has less precision. The repair algorithm in [6] builds upon [9] and considers the performance of 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 like [9]. As a conclusion, the repair techniques in [6] can't deal with situation 1 properly, either.

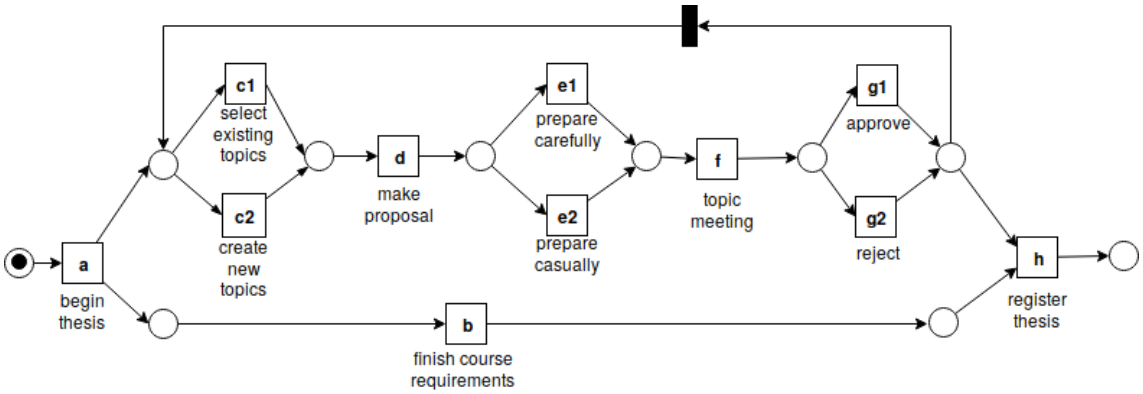
1.1.2 Situation 2: repair model with fit traces

This situation describes the existing problem in the current methods that fit traces with negative performance outcomes can not be used to repair model. Given an actual event log L_2, when activity *finish course requirements* is fired after *begin thesis* and before the topic selection part, it reduces the pressure of master thesis and has a positive outcome. Else, the negative outcomes are given. *Event Log* L_2 –

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



(a) process model with preparation after repair



(b) expected process model with preparation

Figure 1.2: example for model change under model repair

Compared to the model, the event log L_2 contains no deviation. When we apply the techniques in [9] and [6] to repair the model, the model keeps untouched due to no deviation. Apparently, those two methods can't incorporate the negative information in fit traces. When we expect a model which enforce the positive instances and avoid the negative instance as the model in Figure **??**, the current methods fail our need.

1.1.3 Situation 3: detect long-term dependency

This part introduces a problem which causes a lower precision in process model. It is the inability to detect the long-term dependency in the Petri net. An event log L_3 with labels is given in the following. We take the frequency into account, *Event Log* L_3 –

$$\begin{aligned} Positive : \{ &< a, b, \mathbf{c1}, d, f, \mathbf{g1}, h >^{50}, \\ &< a, b, \mathbf{c2}, d, f, \mathbf{g2}, b, \mathbf{c1}, d, f, \mathbf{g1}, h >^{50} \} \\ Negative : \{ &< a, \mathbf{c1}, b, d, f, \mathbf{g1}, h >^{10}, \} \end{aligned}$$

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 Problem And Questions

We analyze the current model repair methods, and give the formal definitions.

Definition 1.1. Given an input of one existing process model M, an event log L and a set of KPIs, how to improve current process enhancement techniques by incorporating negative information, and generate a process model to enforce the positive instances while blocking the negative instance, with condition that the generated model should be as similar to the original model as possible Therefore, the repaired model provides a better way to understand and execute the real business process compared to the original model.

This paper tries to provide a solution for it. Our idea is to analyze the positive and negative impact on process performance of each trace. It balances the existing model, positive traces and negative traces on directly-follows relation, in order to incorporate all the factors on model generation. Later, the directly-follows relation is used to create process model by Inductive Miner. What's more, the impact of the existing model, positive and negative instances are parameterized by weights, to allow more flexibility of the generated model.

1.3 Outline

The reminder is organized in the following order. Section 2 recalls the basic notions on process mining and list the preliminary to solve the problem. The next section lists our methods are introduced and formal definitions are given. In Implementation Section, the details of algorithms are given. Later, we evaluate our methods with simulated data and real data respectively and list the results. Subsequently, the discussion on this paper is presented. At last section, a conclusion is drawn on the paper.

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[16]. Directly-follows[18, 12] methods investigates the activities order in the traces and extracts higher relations which are used to build process models. State-based methods like [1, 4] 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[19]. Other methods due to [20] include search-based algorithm like Genetic Algorithm Miner[5], heuristic-based algorithm Heuristics Miner[22].

Among those discovery methods, Inductive Miner is widely applied[12]. 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. Afterwards, the corresponding operator to the split is used to build a block-structured process model called process tree. Iteratively, the split sublogs are passed as inputs for the same procedure until single activity is reached and no split is available. A process tree is output as the mined process model.

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[15]. Besides extending the model with more data perspectives, 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 [8], 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 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 [8], the sublogs are obtained in a greedy way, while in [9], sublogs are gathered by using ILP Miner to guarantee the fitness. Additional subprocesses and loops are introduced into the existing model to ensure the

fitness, which also brings variants of execution paths into the model.

Later, compared to [8, 9], where all deviations are incorporated in model repair, [6] considers the impact of negative information. In [6], 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 [9]. According to the study case in [6], it provides better result than [9] on the aspect of performance.

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

In [11], 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 [21], the author improves the method in [11] by assigning weights on artificial events with respect to unmatching window, in order to offer generalization on model.

The work in [13] 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 [3] 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 the 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] 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.

[2] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In *OTM Conferences*, 2012.

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

[4] 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.

[5] 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.

[6] 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.

[7] Kefang Ding. Incorporatenegativeinformation.

[8] 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.

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

[10] Mahdi Ghasemi and Daniel Amyot. Process mining in healthcare: a systematised literature review. 2016.

[11] 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.

[12] 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.

[13] 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.

[14] Eindhoven Technical University. © 2010. Process Mining Group. Prom introduction.

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

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

[17] Wil van der Aalst. *Process Mining: Data Science in Action*. Springer Publishing Company, Incorporated, 2nd edition, 2016.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.