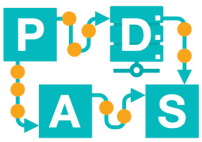

RWTH AACHEN UNIVERSITY

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 results in the wide application of process mining in organizations. Process mining bridges the gap of data mining in big data and business process management by combining data analysis with modeling, controlling and improving business processes.

Process enhancement, as one of the main focuses in process mining, identifies, evaluates and improves the existing processes according to actual execution event logs . It enables continuous improvement on business performance in organizations. Nowadays, more attention is gained on process enhancement and multiple techniques are proposed on it. However, few researches are contributed on the use of negative instances which are execution sequences but lead to bad business performance.

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

Algorithm

This chapter describes the repair algorithm to incorporate the negative instances on process enhancement. At start, the main architecture is listed to provide an overview of our strategy. Next sections will explain the main modules step by step. Firstly, the impact of the existing model, positive and negative instances are balanced in the media of the directly-follow relations. Inductive Miner is then applied to mine process models from those directly-follows relations. Again, we review the negative instances and express its impact by adding the long-term dependency. To add long-term dependency, extra places and silent transitions are created on the model, aiming to enforce the positive instances and block negative instances. Furthermore, the model in Petri net with long-term dependency can be post-processed by reducing the silent transitions for the sake of simplicity.

1.1 Architecture

Figure 1.1 shows the steps of our strategy to enhance a process model. The basic inputs are an event log, and a Petri net. The traces in event log have an attribute for the classification labels of positive or negative in respect to some KPIs of business processes. The Petri net is the referenced model for the business process. To repair model with negative instances, the main steps are conducted.

- *Generate directly-follows graph* Three directly-follows graph are generated respectively for the existing model, positive instance and negative instances from event log.

- *Repair directly-follows graph* The three directly-follows graphs are combined into one single directly-follows graph after balancing their impact.

- *Mine models from directly-follows graph* Process models are mined by Inductive Miner as intermediate results.

- *Add long-term dependency* Long-term dependency is detected on the intermediate models and finally added on the Petri net. To simplify the model, the reduction of silent transitions can be applied at end.

More details can be provided in the following sections.

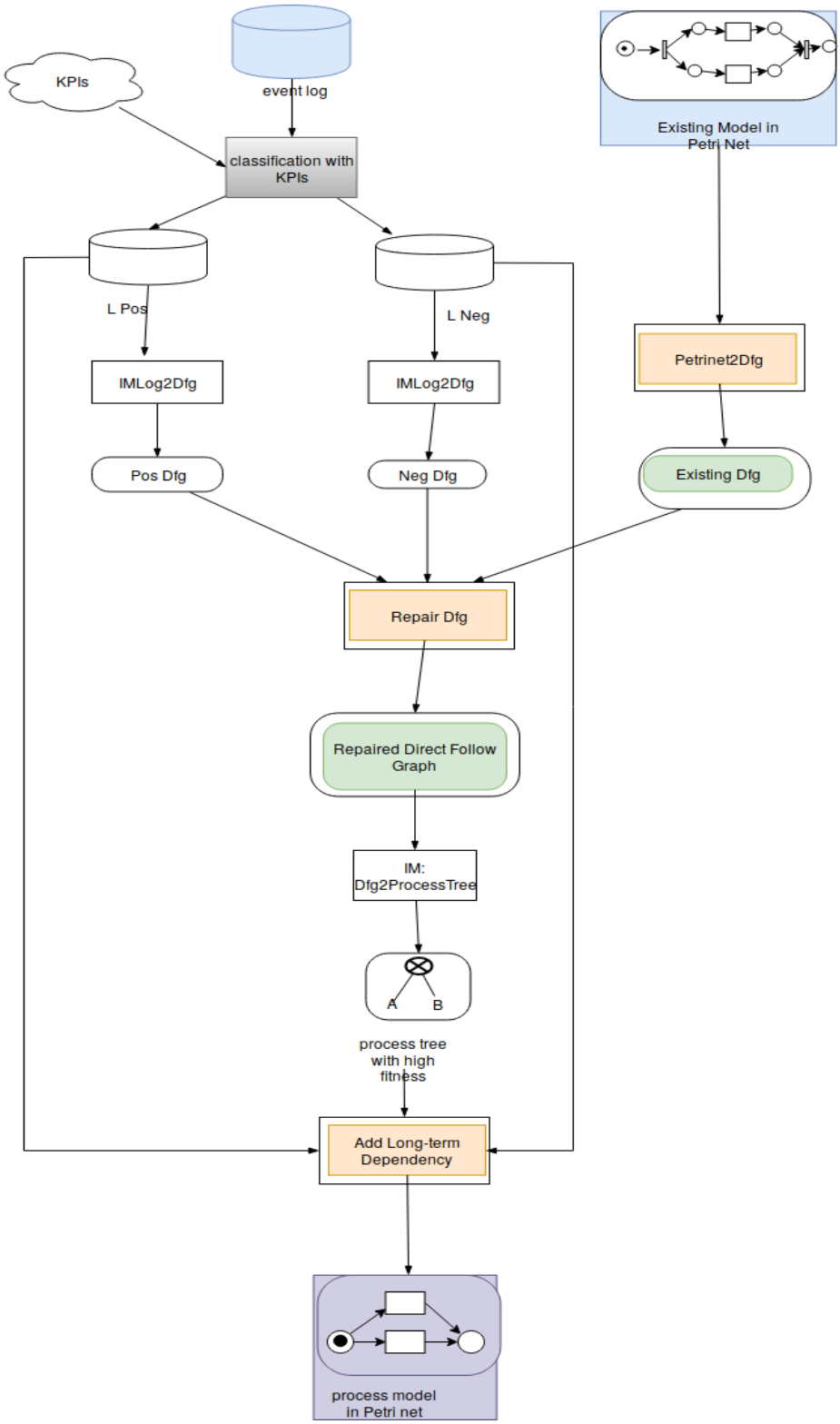

Figure 1.1: Model Repair Architecture – Rectangles represents processes and output data in eclipse shape, especially customized processes and data are in doubled lattice shape. Input event log and existing model are in blue, KPIs are in cloud. The output is a petri net in purple.

1.2 Generate directly-follows graph

Originally, the even log L is split into two sublogs, called L_{pos} and L_{neg}. L_{pos} contains the traces which is labeled as positive, while L_{neg} contains the negative instances in the event log. Then, the two sublogs are passed to procedure *IMLog2Dfg* to generate directly-follows graphs, respectively $G(L_{pos})$ and $G(L_{neg})$. More details about the procedure is available in [1].

To generate a directly-follows relation from a Petri net, we gather the model behaviors by building a transitions system of its states. Then the directly-follows relations are extracted from state transitions. Based on those relations, we create a directly-follows graph for the existing model.

From the positive and negative event log, we can get the cardinality for corresponding directly-follows graph, to represent the strength of this directly-follows relation. However, when the existing model is transformed into directly-follows graph $G(L_{ext})$, there is no point to assign cardinality on each edge. So we just set cardinality with 1 for each edge.

1.3 Repair directly-follows graph

To combine all information from $G(L_{pos})$, $G(L_{neg})$ and $G(L_{ext})$, the cardinality in directly-follows graphs has to be unified into the same range. In this thesis, the unification range is [0-1], the unified value is called weight and defined as the following.

Definition 1.1 (Weight of directly-follows relation)**.** Given a directly-follows graph G(L), the weight of each directly-follows relation is defined as

$$Weight(E(A,B)) = \frac{Cardinality(E(A,B))}{Cardinality(E(A,*))}$$

for start activities A, we have

$$Weight(Start(A)) = \frac{Cardinality(Start(A))}{Cardinality(Start(*))}$$

Similarly for end activities B, we have

$$Weight(End(B)) = \frac{Cardinality(End(B))}{Cardinality(End(*))}$$

E(A,*) means all edges with source A, E(*,B) means all edges with target B, Start(*) represents all start nodes, and End(*) represents all end nodes.

1.4 Add long-term dependency

1.5 Reduce Silent Transitions

Bibliography

[1] 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.