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

Evaluation

In this chapter, we evaluate the proposed 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.

1.1 Evaluation Measurements

We evaluate the 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.
- *Precision.* It quantifies 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 the environment where only positive instances are available. Therefore, when it comes to the model performance, where negative instances are also possible, the measurement metrics need to 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 and confusion matrix is applied in our experiments.

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

Table 1.1: Confusion Matrix

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

for actual classification on data. The repaired model can be seen a binary classifier and produces four outcomes- true positive, true negative, false positive and false negative shown in the Table 1.1.

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

1.2 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 due to the time limits. So partial experiments are still conducted in ProM.

1.2.1 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.

1.2.2 Experiments with ProM Plugins

Due to the frequent errors on the corresponding plugin, we exclude the tests on repair techniques in [4] and conduct experiments with 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 Dfg-Repair 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.

1.3 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. For convenience,we refer the repair methods in [5] by the main auther's name Fahland, and the repair techniques in [4] as Dees. The Inductive Miner is abbreviated as IM. Our method which built on directly-follows graph is denoted as Dfg-repair.

1.3.1 Test on Demo Example

In this part, experiments are performed on the motivating examples which are listed in Introduction. Thereby, we are able to answer the question: Will our repair method overcome shortcomings of current techniques which are shown in the introduction chapter?

1.3.1.1 Answer to Situation 1

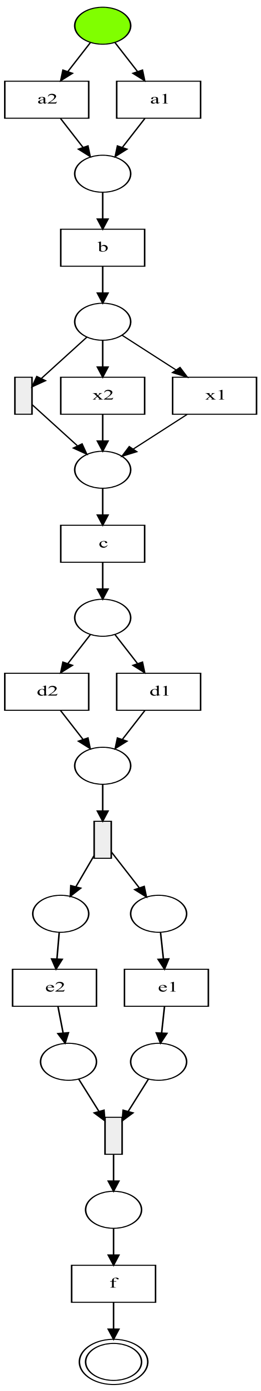
Situation 1 in Introduction shows the drawbacks of current repair methods [4, 5]. Additional activities **x1,x2** lead to good performance as shown in the actual event log L_1. Given the original Petri net M_0 and event log L_1, subprocesses for **x1,x2** are added as loops in the model, which brings more unexpected behaviors in the model. Rediscover strategy doesn't take the original model into account and generates a new model $M_{1.2}$ in Figure **??** that deviates from the original model M_0.

When we apply our repair techniques on all situations listed on Introduction, and get repaired models listed in Figure 1.1. The parameters for our method are set in the following : weight for the existing model is 0.45, weight for positive examples is 1.0, the Inductive Miner for Infrequent is chosen and has a noise with 20, the same setting as the rediscovery method by Inductive Miner in Introduction. As seen in Figure 1.1a, the subprocesses for **x1,x2** are added in a sequence with others and can be skipped by the silent transition. In this way, $M_{1.3}$ is able to reflect the deviations in positive instances while keeping similar to the reference model M_0. Compared to techniques in [5], it increases the precision without loops.

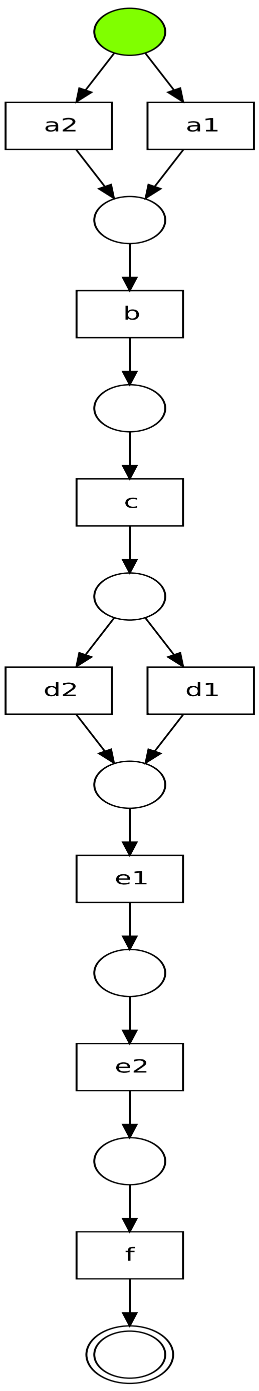
1.3.1.2 Answer to Situation 2

Situation 2 describes the inability of current repair methods that fitting traces with negative performance outcomes cannot be used to repair a model. The execution order of **e1, e2** affects the performance outcomes and **e1** is expected to position before **e2**. Without negative information, the repaired models have the same structure as the reference ones, because the execution of **e2** before **e1** brings also the positive outcomes.

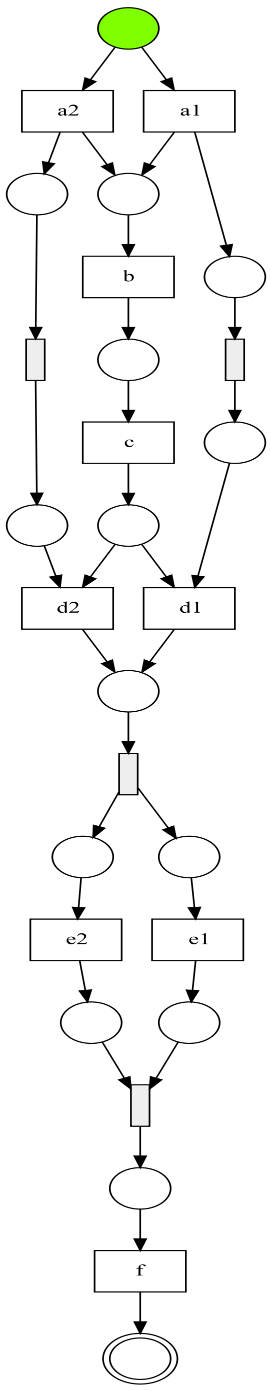
If we apply our repair methods on the model M_0 and event log L_2, with setting 1.0 for all control weights, and the same Inductive Miner-Infrequent with noise 20, the repaired model $M_{2.3}$ is obtained. In $M_{2.3}$, **e1** is executed before **e2**. The reason behind our method



(a) $M_{1.3}$ for situation 1



(b) $M_{2.3}$ for situation 2



(c) $M_{3.3}$ for situation 3

Figure 1.1: repaired models with our techniques for situation 1,2 and 3 in Introduction part. The green place is the initial marking of the Petri net and the doubled place is the final marking.

is that it incorporates the negative information and balances the forces from the existing model, positive and negative instances.

1.3.1.3 Answer to Situation 3

Situation 3 concerns the long-term dependency in Petri nets. As observed in event log, there exists the long-term dependency set, $LT = \{a1 \rightsquigarrow d1, a2 \rightsquigarrow d2\}$. With adding long-term dependency as expected in Figure **??**, precision and accuracy increase, since the model limits activity selection and blocks the negative behavior due to free execution of xor branches. Yet none of the current repair and rediscovery techniques are able to detect and add long-term dependencies in the Petri net.

In our repair techniques, the long-term dependency is taken into account with negative information. With inputs of the Petri net M_0 and event log L_3, our methods produces the repaired model $M_{3.3}$ with long-term dependency. Two silent transitions that are used to explicitly represent the long-term dependencies can be deleted with post procedure to reduce the redundant silent transitions and places. After reduction, our repaired model is simplified as the model M_3.

1.3.1.4 Comparison with Confusion Matrix

In this section, we list the evaluation results of the repaired models based on confusion matrix. In Table 1.2, for Situation 1 with only positive instances, the repair techniques give us the same confusion matrix result. However, $M_{1.2}$ with loops implicates a lower precision. In Situation 2, with current techniques or rediscovery methods in IM, the model stays the same as the reference model M_0. Since no negative instance is rejected, the recall has the value 1 but precision below 0.6. In comparison, our method uses the negative instances and adjusts the model correspondingly. Therefore, the repaired model $M_{2.3}$ has higher precision, accuracy and F1 score. In Situation 3 with long-term dependency, our method succeeds to detect and add long-term dependency in the model. Thereby, it has no false positive or false negative instances in the confusion matrix, and holds the highest values for all listed measurements.

Table 1.2: Test Result on BPI15-M1 data

Situation	**method**	**Generated model**	**confusion matrix metrics**							
			TP	**FP**	**TN**	**FN**	**recall**	**precision**	**accuracy**	**F1**
S1	IM	$M_{1.1}$	50	50	0	0	1	0.5	0.5	0.667
S1	Fahland	$M_{1.2}$	50	50	0	0	1	0.5	0.5	0.667
S1	Dfg-repair	$M_{1.3}$	50	50	0	0	1	0.5	0.5	0.667
S2	IM/Fahland	M_0	60	45	0	0	1	0.571	0.571	0.727
S2	Dfg-repair	$M_{2.3}$	50	5	40	10	0.833	0.909	0.857	0.870
S3	IM/Fahland	M_0	100	100	0	0	1.0	0.5	0.5	0.667
S3	Dfg-repair	$M_{3.3}$	100	0	100	0	1	1	1	1

As conclusion, our proposed method is able to overcome shortcomings of current techniques which are shown in the Introduction chapter. It avoids the loops in model by repairing the model with additional activities, incorporates the negative information in the data to adjust the model, also detect and add the long-term dependency into model. In this way, the repaired model has better recall, accuracy.

1.3.2 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.

1.3.2.1 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 **SUM-leges** 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 activities 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.

Table 1.3: 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

We filter the raw event log by ***Filter Log By Simple Heuristic*** in ProM with the following setting. 40 for the start, end activities and the events between them, at end. We get the event log $D1$. After this, we calculate the throughput time for each trace and add it as a trace attribute **throughput time**. Then we classify traces according to **SUMleges** and **throughput time** separately. When our performance goal is to reduce the cost of application, if **SUMleges** of one trace is over 0.7 of the whole traces, this trace is treated as negative, else as positive. The similar strategy is applied on the attribute **throughput time**. A trace with **throughput time** higher than 0.7 of all traces is considered as a negative instance. Following this preprocess, we have event logs in Table 1.3 available for our tests.

Table 1.4: 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 1.4 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.

1.3.2.2 Test Result

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 1.3 and models from Table 1.4. The experiment result is shown in the Table 1.5.

With the similar measurements, it is observed that the repaired models from **Type 2** are more complex that Dfg-repair 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.

Table 1.5: Test Result on BPI15-M1 data

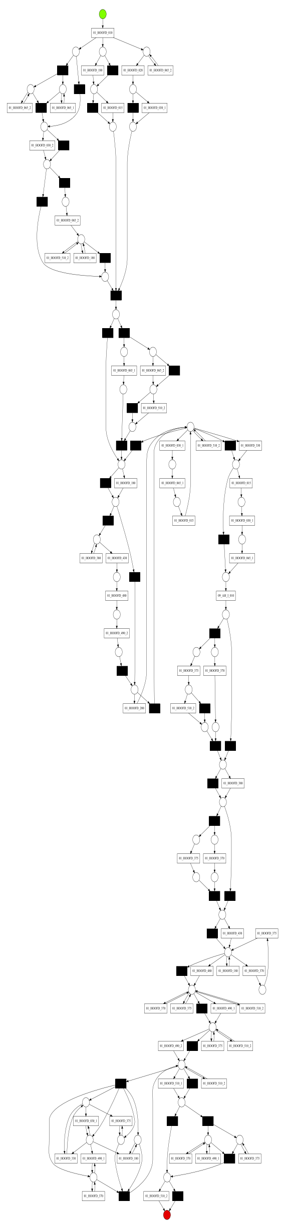
event log	**reference model**	**method**	**confusion matrix metrics**							
			TP	**FP**	**TN**	**FN**	**recall**	**precision**	**accuracy**	**F1**
D3.1	-	IM	137	48	118	289	0.32	0.74	0.43	0.45
D3.1	M1	Fahland	343	136	10	6	0.983	0.716	0.713	0.829
D3.3	M1	Dfg-repair	124	52	94	225	0.355	0.705	0.44	0.472
D3.1	M2	Fahland	317	133	13	32	0.908	0.704	0.667	0.793
D3.3	M2	Dfg-repair	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-repair	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-repair	0	0	349	146	0	NaN	0.295	0
D4.1	-	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-repair	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-repair	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-repair	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-repair	0	0	346	149	0	NaN	0.303	0

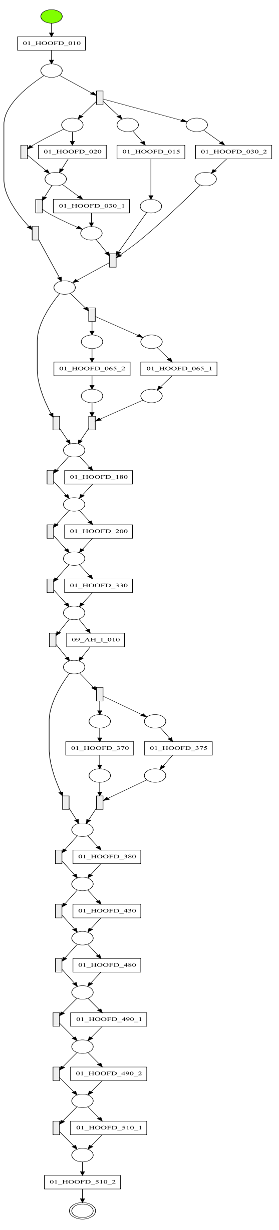
Each of three control parameters for the existing model, positive and negative instances changes value from 0.0 to 1.0 with step 0.1. With this setting, directly-follows relation is generated. Afterward, the default setting of Inductive Miner Infrequent with noise threshold 20 is used to mine Petri nets from the generated directly-follows graph. In total, thousand experiments are conducted to investigate Dfg-repair method. 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 1.3, 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 1.4 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 1.5 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 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 $M_{1.2}$.

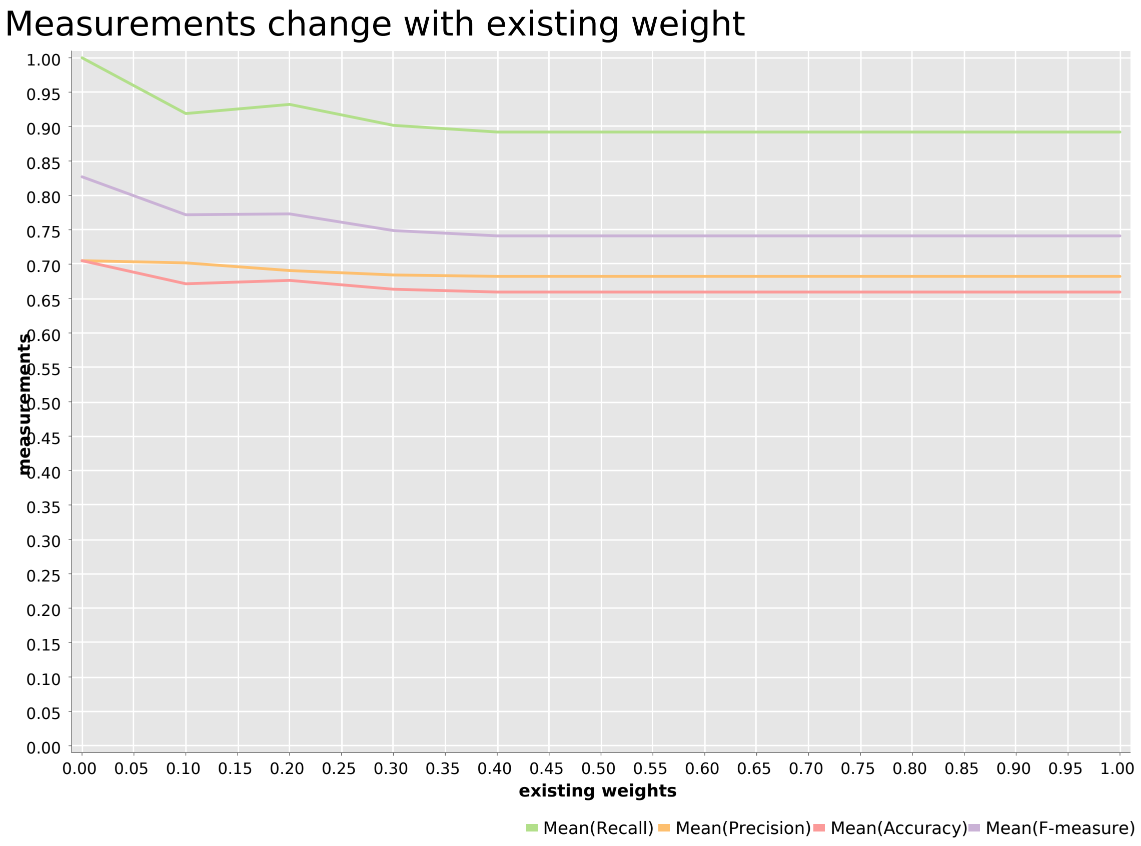


Figure 1.3: result with control parameter for existing model on event log D3.3 and model M3

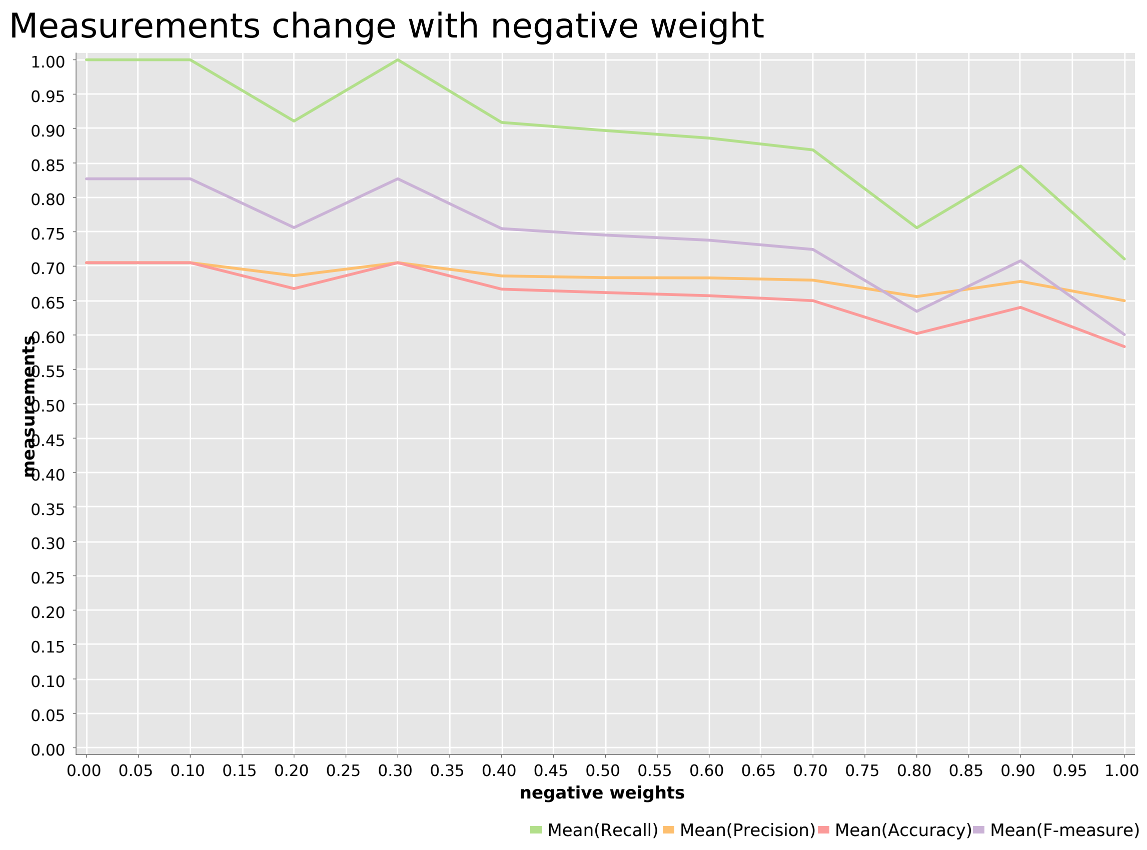


Figure 1.4: result with control parameter for negative instance on event log D3.3 and model M3

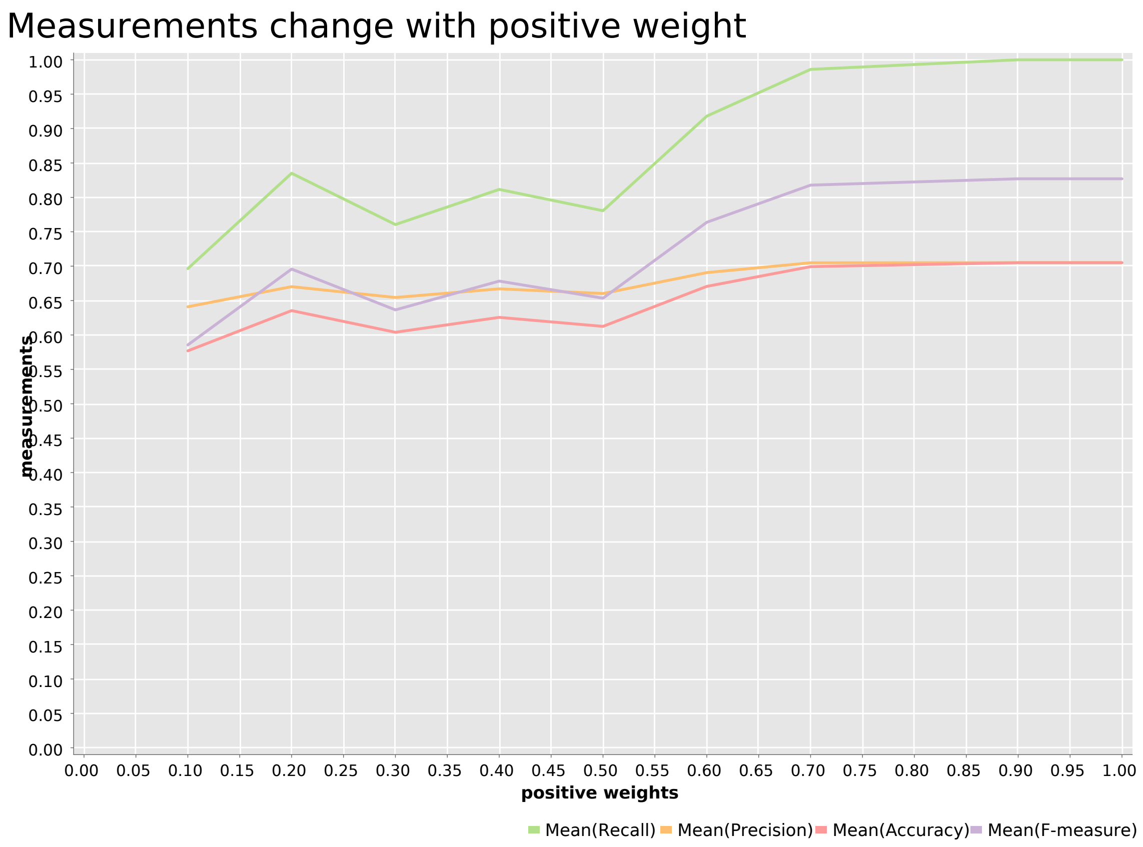


Figure 1.5: result with control parameter for positive instance on event log D3.3 and model M3

Chapter 2

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 synthetic data and real life data. In the situations shown with synthetic data, our method is able to overcome the shortcomings of the current repair techniques and provide repaired models with higher accuracy and precision. In experiments with the real life data, Additionally, with respect to other methods, it runs faster and generates simpler models.

As future work, we consider improving 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 loop. 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.

[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.

[19] BF Van Dongen, Jan Mendling, and WMP Van Der Aalst. Structural patterns for soundness of business process models. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 116–128. IEEE, 2006.

[20] Wil van der Aalst. *Process Mining: Data Science in Action*. Springer Publishing Company, Incorporated, 2nd edition, 2016. ISBN 3662498502, 9783662498507.

[21] 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.

[22] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[23] HMW Verbeek. Decomposed replay using hiding and reduction as abstraction. In *Transactions on Petri Nets and Other Models of Concurrency XII*, pages 166–186. Springer, 2017.

[24] Eindhoven Technical University. © 2010. Process Mining Group. Prom introduction. URL `http://www.promtools.org/doku.php`.

[25] Kefang Ding. Incorporatenegativeinformation. URL `http://ais-hudson.win.tue.nl:8080/job/IncorporateNegativeInformation/`.