A Novel Fitness Improvement Method for Mined Business Process Models

Yaguang Sun and Bernhard Bauer

Software Methodologies for Distributed Systems, University of Augsburg, Germany
{yaguang.sun,bernhard.bauer}@informatik.uni-augsburg.de

Abstract. Business process model discovery (BPMD) is a significant research topic in the business process mining area. The present BPMD techniques encounter great challenges while dealing with real-life event logs that contain complex process behaviors. As a result, non-fitting process models might be obtained. In this paper, we propose a new mechanism for locating and handling the process behaviors recorded in event logs which cannot be expressed by the utilised model discovery algorithms.

Keywords: Business Process Mining, Business Process Model Discovery, Process Model Fitness Improvement

1 Introduction

As one of the most important research directions in business process mining area, the present business process model discovery (BPMD) techniques meet great challenges while trying to mine process models from real-life event logs that usually stem from the business processes implemented in highly flexible environments, e.g., healthcare, customer relationship management (CRM) and product development [1, 3]. Such real-life logs often contain complex process behaviors and utilising existing BPMD techniques to mine these logs might generate non-fitting process models.

The *mining algorithm enhancement-based strategy* has been put forward in the literature for solving the problem of low-fitness process models mined from real-life event logs. Such a strategy aims at improving the expressive ability of existing BPMD algorithms or developing new algorithms that are able to model more complex workflow patterns.

In this paper, we develop a new method which inherits the basic idea of *mining algorithm enhancement-based strategy* for assisting the present BPMD techniques in generating high-fitness process models from real-life event logs. In our method, the fitness improvement problem for the inaccurately mined process models is surveyed from a new perspective and redefined as an issue of locating the inexpressible process behaviors recorded in event logs and then transforming them into expressible behaviors for the utilised BPMD algorithms. The structure of the main contents of this paper is organised as:

In: S. España, M. Ivanović, M. Savić (eds.): Proceedings of the CAiSE'16 Forum at the 28th International Conference on Advanced Information Systems Engineering, Ljubljana, Slovenia, 13-17.6.2016, published at http://ceur-ws.org

- In Section 2, a novel method named IDC is put forward which is able to help the employed BPMD techniques mine more fitting process models from real-life logs.
- In Section 3, we carry out the proposed method IDC on an example event log to test the correctness and effectiveness of it.
- In Section 4, the related work from the academia is reviewed.
- In Section 5, a conclusion together with the future problems that we are going to solve are elaborated.

2 Approach Design

We firstly introduce a method for detecting and structuring the process behaviors from real-life event logs in Section 2.1. Afterwards, based on the method presented in Section 2.1 a heuristic technique named IDC for helping the utilised BPMD algorithms mine more fitting process models is put forward in Section 2.2.

2.1 Construct Process Behavior Space (PBS)

Business process behaviors from real-life event logs cannot be analysed before they are extracted and structured in an appropriate form. We solve the PBS construction problem by providing the following two concepts:

Definition 1. *(behavior-related activity)*
Let SA be the set of activities for event log E. Let $\succ_E$ represents a direct relation between any two activities from SA. For instance, for two activities $a, b \in SA$, $a \succ_E b = true$ if activity a is directly followed by b at least once in E. Symbol $\Rightarrow_E$ represents a behavior-based relation between any two activities from SA. For two activities $c, d \in SA$, $c \Rightarrow_E d = true$ if $c \succ_E d = true$ or $d \succ_E c = true$ and d is also called a behavior-related activity of c.

Definition 2. *(behavior-related sub-trace)*
Let SA be the set of activities and ST be the multiset of traces for event log E. Let $t \in ST$ be a trace from E, $st \sqsubseteq t$ be a sub-trace of t and SA_{st} be the set of activities for st. Given an activity $a \in SA$, st is a behavior-related sub-trace of a if $\forall b \in SA_{st} \wedge b \neq a$ such that $a \Rightarrow_E b$ and $\exists c \in SA_{st}$ such that $c = a$.

Take a simple event log $E_1 = \{<l_1, m_1, n_1, x_1, y_1, z_1>^{15}, <l_1, n_1, m_1, y_1, z_1>^{15}, <l_1, m_1, x_1>^{3}, <l_1, n_1, m_1, z_1>^{5}\}$ (utilised for the rest of Section 2) as an example. According to Definition 1, the activity n_1 from E_1 has three behavior-related activities which are activity l_1, m_1 and x_1. Then, according to Definition 2, a set of behavior-related sub-traces $\{<l_1, m_1, n_1, x_1>^{15}, <l_1, n_1, m_1>^{20}\}$ for activity n_1 can be extracted from E_1. Let $\Xi : (ST^+, SA^+, SA) \rightarrow SBT$ be a function for finding all of the behavior-related sub-traces for a specific activity from a given trace, where ST^+ represents the set of traces, SA^+ represents the set of all possible sets of behavior-related activities, SA represents the set of activities and SBT stands for the set of all possible multisets of behavior-related sub-traces. The PBS construction procedure is described in Algorithm 1.

Algorithm 1 Construct the PBS for a specific event log E (CPBS)

Input: an event log E, the set of activities SA for E
Let BA be a set of behavior-related activities.
Let BT be a set of behavior-related sub-traces.
Let PBS be a set of sets of behavior-related sub-traces.
1: $BA \leftarrow null$, $BT \leftarrow null$, $PBS \leftarrow null$
2: **for** each activity $a \in SA$ **do**
3: **for** each trace $t_1 \in E$ **do**
4: **for** each activity $b \in t_1$ **do**
5: **if** $b \notin BA$ && $a \Rightarrow_E b = true$ **then**
6: $BA \leftarrow BA \cup b$
7: **end if**
8: **end for**
9: **end for**
10: **for** each trace $t_2 \in E$ **do**
11: $BT \leftarrow BT \cup \Xi(t_2, BA, a)$
12: **end for**
13: $PBS \leftarrow PBS \cup BT$
14: $BA \leftarrow null$
15: $BT \leftarrow null$
16: **end for**
Output: a set of sets of behavior-related sub-traces PBS

2.2 Detection and Conversion of Inexpressible Process Behaviors

In this subsection, we firstly define a new concept called *activity environment item*. Afterwards, a new algorithm named IDC that is able to assist in detecting and converting the inexpressible process behaviors related to one particular activity is presented.

Definition 3. *(activity environment item)*
Let SA be the set of activities from event log E, activity a, b and c are three activities from SA, the tuple (b, c) is an environment item of activity a if $\exists t \in E$ such that $<b, <a \ldots>, c> \sqsubseteq t$, where t stands for a trace from E and $<a \ldots>$ represents a sub-trace that only consists of activity a.

Take the event log E_1 mentioned in Section 2.1 as an example. According to Definition 3, the activity $n_1 \in E_1$ has two types of environment items which are (m_1, x_1) and (l_1, m_1). Activity $l_1 \in E_1$ also has two types of environment items that are $(null, m_1)$ (the value $null$ indicates that l_1 appears as a starting activity) and $(null, n_1)$. Let $\Theta : (SA, SBT) \rightarrow SEI^+$ be an environment item searching function for a specific activity, where SA represents the set of activities, SBT represents the set of all multisets of behavior-related sub-traces and SEI^+ stands for the set of all sets of environment items. Let $\Phi : (SA, SEI, SBT, SNA) \rightarrow SBT$ be an activity conversion function, where SEI stands for the set of environment items and SNA represents the set of activity names. Given an activity, function Φ is capable of converting this activity into a new activity (with a new name)

Algorithm 2 Inexpressible behaviors detection and conversion (IDC)

Input: an activity a from event log E, the *PBS* for log E, a model fitness threshold ε, a model fitness improvement threshold η

Let BST_a and BST_t be two sets of behavior-related sub-traces.
Let SEI_a and SEI_r be two sets of environment items.
Let f_1 and f_2 be two variables of type Double.
Let i be a variable of type Integer.
Let ei_t be an environment item.

1: $f_1 \leftarrow 0$, $f_2 \leftarrow 0$, $i \leftarrow 0$
2: $ei_t \leftarrow null$, $BST_a \leftarrow null$, $BST_t \leftarrow null$, $SEI_a \leftarrow null$, $SEI_r \leftarrow null$
3: $BST_a \leftarrow PBS[a]$ # collect the set of behavior-related sub-traces for a from *PBS*
4: $SEI_a \leftarrow \Theta(a, BST_a)$ # find the set of environment items for activity a
5: $f_1 \leftarrow \Sigma(\Omega(BST_a), BST_a)$ # calculate the fitness of the model mined from BST_a
6: **while** ($|SEI_a| > 0$ && $f_1 < \varepsilon$) **do**
7: **for** each environment item $ei_a \in SEI_a$ **do**
8: $BST_t \leftarrow \Phi(a, ei_a, BST_a, a_i)$ # transform a into a_i under environment ei_a
9: **if** ($\Sigma(\Omega(BST_t), BST_t) > f_2$) **then**
10: $f_2 \leftarrow \Sigma(\Omega(BST_t), BST_t)$
11: $ei_t \leftarrow ei_a$
12: **end if**
13: **end for**
14: **if** ($f_2 - f_1 > \eta$) **then**
15: $f_1 \leftarrow f_2$
16: $BST_a \leftarrow \Phi(a, ei_t, BST_a, a_i)$
17: $SEI_r \leftarrow SEI_r \cup ei_t$
18: remove ei_t from SEI_a
19: $i \leftarrow i + 1$
20: **else**
21: *break*
22: **end if**
23: **end while**

Output: a set of environment items SEI_r for activity a.

under a certain environment in all of the behavior-related sub-traces of the given activity. For the activity $n_1 \in E_1$, given an environment item $ei = (m_1, x_1)$ of n_1, the set of behavior-related sub-traces $BST_{n_1} = \{< l_1, m_1, n_1, x_1 >^{15}, < l_1, n_1, m_1 >^{20}\}$ of n_1 and a new activity name n_{1_0}, $\Phi(n_1, ei, BST_{n_1}, n_{1\_0}) = \{< l_1, m_1, n_{1\_0}, x_1 >^{15}, < l_1, n_1, m_1 >^{20}\}$. Let *SST* be the set of all possible multisets of traces, $\Omega : SST \rightarrow SM$ be a workflow discovery algorithm, where *SM* is the set of process models. $\Sigma : (SM, SST) \rightarrow SV$ represents a process model fitness evaluation schema with an input of a process model and its relevant set of traces and an output of an assessed value from *SV* (the set of all possible values output by Σ). The details of IDC is described in Algorithm 2.

For a specific activity a from log E, the algorithm IDC firstly obtains the set of behavior-related sub-traces BST_a for a through step 3, the set of environment items SEI_a for a by step 4 and the fitness of the behavior-related model (by mining the generated set of behavior-related sub-traces BST_a) of a by step

5. Afterwards, steps $6-13$ try to find the environment item of a under which converting a into a_i in the set of behavior-related sub-traces of a will help generate a behavior-related model (by mining BST_t) with maximal fitness and the discovered environment item is put in ei_t and the value of the maximal fitness is stored by f_2. Then, it is checked if the fitness improvement acquired is larger than or equal to a threshold η by step 14. If this is the case, the environment item is put in set SEI_r (step 17) which will be finally output by IDC. The original set of behavior-related sub-traces is updated (step 16) by transforming the relevant activity under the found environment which is removed from SEI_a later by step 18 and IDC continues to search for the next environment item if SEI_a is not empty and the fitness of the new behavior-related model is less than a target value ε. Otherwise, the technique IDC stops.

3 Evaluation

In our experiment, we utilise the Heuristics Miner (HM) [7] from ProM 6[1] for mining process models from event logs. At present, several application instances [9–14] of the *mining algorithm enhancement-based strategy* (introduced in Section 1) have been put forward in the literature. These proposed methods are able to help mine high-fitness process models expressed by Petri net [15]. However, few efforts have been made to help the HM mine high-fitness models. According to [16], the HM which expresses process model by Heuristics net is one of the most popular BPMD tools in the ProM framework and this is why we adapt our technique to HM for the evaluation. Adapting our technique to other BPMD algorithms will be a future job. Furthermore, we use the ICS fitness [8] for assessing the accuracy of the mined models because it has a computationally efficient calculative process.

We tested the correctness and effectiveness of our technique by utilising an example event log E_e. As shown in Figure 1, log E_e has nine activities and 523 traces and the process model generated by implementing the HM on E_e has a low ICS fitness value which is 0.7748.

In the process of our experiment, we firstly utilise the algorithm CPBS (introduced in Section 2.1) to build the PBS for log E_e. Afterwards, the algorithm IDC (introduced in Section 2.2) is used to analyse the built PBS for discovering the activities together with their relevant environment items that are the main factors to turn out the inexpressible process behaviors for HM in log E_e. The model fitness threshold ε is set to 1 and the model fitness improvement threshold η is set to 0.3 for IDC.

According to Figure 2, the method IDC finds that the activity F under environment (D, E) and (H, I) in log E_e is the key factor for generating the process behaviors that cannot be expressed by HM. With this discovery from IDC, we replace the activity F under environment (H, I) by a new activity named F_0 and the activity F under environment (D, E) by using F_1 in log E_e so

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

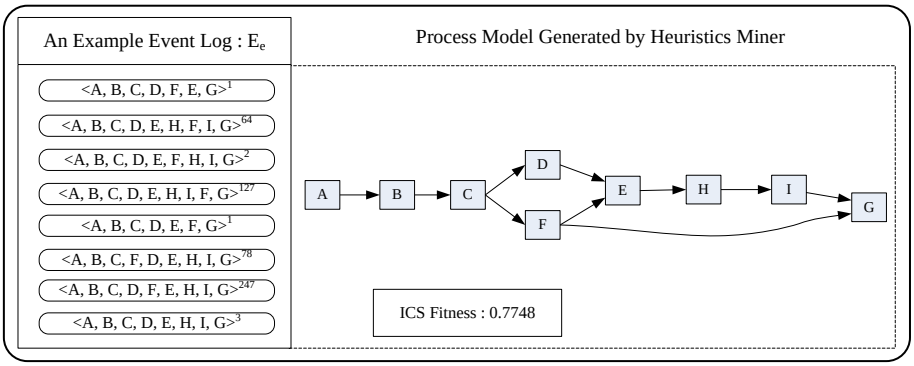

Fig. 1. The example event log E_e and the business process model mined from E_e by HM.

that a new log E_{e^*} (in Figure 2) can be generated. As shown in Figure 2, a more fitting process model (with a fitness value 0.9987) is output by HM executed on the newly generated log E_{e^*}. This positive evaluation results indicate that our technique IDC successfully locate the inexpressible process behaviors from E_e for HM.

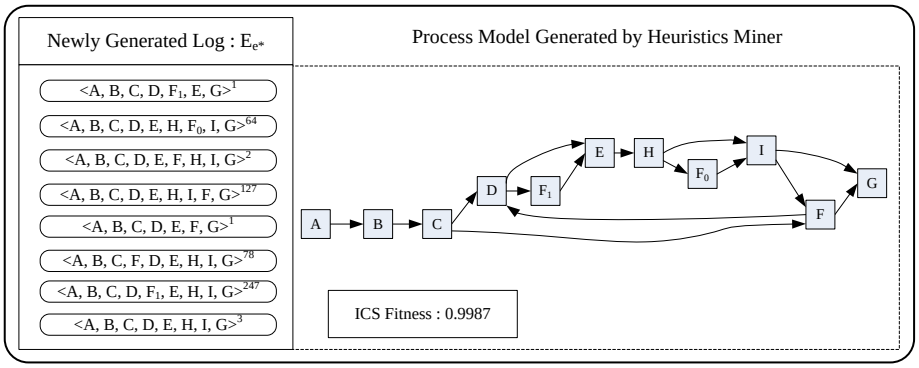

Fig. 2. The newly generated event log E_{e^*} and the business process model mined from E_{e^*} by HM.

4 Related Work

In the literature, several effective techniques stemming from the *mining algorithm enhancement-based strategy* have been devised. Most of these presented techniques are developed to mine fitting process models expressed by Petri net.

The authors in [10] put forward the State-based Region Miner which is able to generate different categories of place-irredundant Petri nets from the transition systems and assure the fitness of the generated Petri nets. In [9], the authors point out that the BPMD approaches partly based on heuristic methods cannot guarantee an accurate mining results. Then, they develop a series of methods for adapting the technique of Petri net synthesis to the BPMD area so that the advantage from the Petri net synthesis can be utilised to mine high fitness process models from real-life event logs. The authors in [11] propose the ILP Miner based on the idea that places can restrict the possible firing sequences of a Petri net. The ILP Miner assures to mine highly fitting Petri nets whose sizes are independent of the number of events in the event logs. The authors in [12, 13] put forward the Inductive Miner which guarantees to return a fitting model that is also sound. The process model repair technique is presented by the authors in [14] for generating a high-fitness process model. Such a technique tries to divide the original log into several sublogs with non-fitting subtraces. For every sublog, a sub-model is obtained and added to the original model at the appropriate place.

5 Conclusion

In this paper, we mainly proposed a technique IDC for helping detect and convert the inexpressible process behaviors for the utilised BPMD techniques from particular real-life event logs. Through the evaluation results from Section 3, we demonstrated the effectiveness of our technique.

A fitting model might still be complex. Our future job will be focused on designing a new trace clustering technique [2–6] that contains two steps where the first step focuses on building sub-models with lower complexity and the second step focuses on improving the accuracy of the mined sub-models by using the technique proposed in this paper. In the meantime, we will also validate our method on some other real-life cases.

References

1. W.M.P. van der Aalst. Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin, 2011.
2. M. Song, C.W. Gnther and W.M.P. van der Aalst. Trace Clustering in Process Mining. In Business Process Management Workshops 2008, volume 17 of Lecture Notes in Business Information Processing, pages 109-120. Springer, Berlin.
3. J.D. Weerdt, S. vanden Broucke, J. Vanthienen and B. Baesens. Active Trace Clustering for Improved Process Discovery. IEEE Transactions on Knowledge and Data Engineering, 25(12):2708-2720, 2013.
4. Y.G. Sun and B. Bauer. A Novel Top-Down Approach for Clustering Traces. In Advanced Information Systems Engineering (CAiSE 2015), volume 9097 of Lecture Notes in Computer Science, pages 331-345.

5. C.C. Ekanayake, M. Dumas, L. Garcia-Banuelos and M. La Rosa. Slice, Mine and Dice: Complexity-Aware Automated Discovery of Business Process Models. In Business Process Management (BPM 2013), volume 8094 of Lecture Notes in Computer Science, pages 49-64.
6. L. Garcia, M. Dumas, M.L. Rosa, J.D. Weerdt and C.C. Ekanayake. Controlled Automated Discovery of Collections of Business Process Models. Inf. Syst., 46:85-101, 2014.
7. A.J.M.M. Weijters, W.M.P. van der Aalst and A.K. Alves de Medeiros. Process Mining with the Heuristics Algorithm. TU Eindhoven, BETA Working Paper Series 166, 2006.
8. A.A. de Medeiros. Genetic Process Mining. PhD. thesis, Eindhoven University of Technology (2006).
9. R. Bergenthum, J. Desel, R. Lorenz and S. Mauser. Process Mining Based On Regions of Languages. In Business Process Management (BPM 2007), volume 4714 of Lecture Notes in Computer Science, pages 375-383.
10. J. Cortadella, M. Kishinevsky, L. Lavagno and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. IEEE Transactions on Computers, 47(8):859-882, 1998.
11. J. van der Werf, B. van Dongen, C. Hurkens and A. Serebrenik. Process Discovery Using Integer Linear Programming. Applications and Theory of Petri Nets, pages 368-387, 2008.
12. S.J.J. Leemans, D. Fahland and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs-A Constructive Approach. In: Petri Nets, volume 7927 of Lecture Notes in Computer Science, pages 311-329, 2013.
13. S.J.J. Leemans, D. Fahland and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In Business Process Management Workshops 2013, volume 171 of Lecture Notes in Business Information Processing, pages 66-78.
14. D. Fahland and W.M.P. van der Aalst. Repairing Process Models to Reflect Reality. In Business Process Management (BPM 2012), volume 7481 of Lecture Notes in Computer Science, pages 229-245.
15. T. Murata. Petri Nets: Properties, Analysis and Applications. Proc. of the IEEE, 77(4):541-580, 1989.
16. J. Claes and G. Poels. Process Mining and the ProM Framework: An Exploratory Survey. In Business Process Management Workshops 2012, volume 132 of Lecture Notes in Business Information Processing, pages 187-198.