A Novel Heuristic Method for Improving the Fitness of Mined Business Process Models

Yaguang Sun(✉) and Bernhard Bauer

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

Abstract. Business process model discovery (BPMD) is one of the most important research topics in the business process mining area. Many outstanding BPMD algorithms which perform well in most cases have been developed in the last years. As one of the most widely used BPMD algorithms, the Heuristics Miner meets great challenges while dealing with event logs that contain complex behaviours. As a result, process models with low fitness values might be obtained. In this paper, we propose a new technique that is able to locate the process behaviours recorded in an event log which cannot be expressed by the Heuristics Miner and then transform them into expressible behaviours so that a high-fitness model can be built.

Keywords: Business process mining · Business process model discovery · Model fitness improvement · Heuristics Miner

1 Introduction

The present BPMD techniques meet great challenges while mining models from real-life event logs. Such logs often contain complex trace behaviours which might be far beyond the expression ability of existing BPMD approaches [9]. Figure 1 shows an example event log L_1 and the model generated by carrying out the Heuristics Miner (HM) [3] on this log. The mined model for L_1 has a relatively low fitness (0.7752) due to the existence of inexpressible process behaviours for HM in L_1. Several pioneering approaches [6–9] have been put forward in academia for solving the problem mentioned above. These proposed methods are able to help mine high-fitness models expressed by Petri net [1]. However, few efforts have been made to help the HM (that expresses process model by Heuristics net) get better mining results. According to [10], the HM is one of the most important and widely utilised BPMD tools in the ProM framework [1] for dealing with real-life event logs. Developing an auxiliary method to help it mine high-fitness process models is far from trivial.

In this paper, we put forward a novel heuristic method named HIF which transforms the fitness improvement problem for the non-fitting models mined by HM into the problem of locating the inexpressible process behaviours of HM in

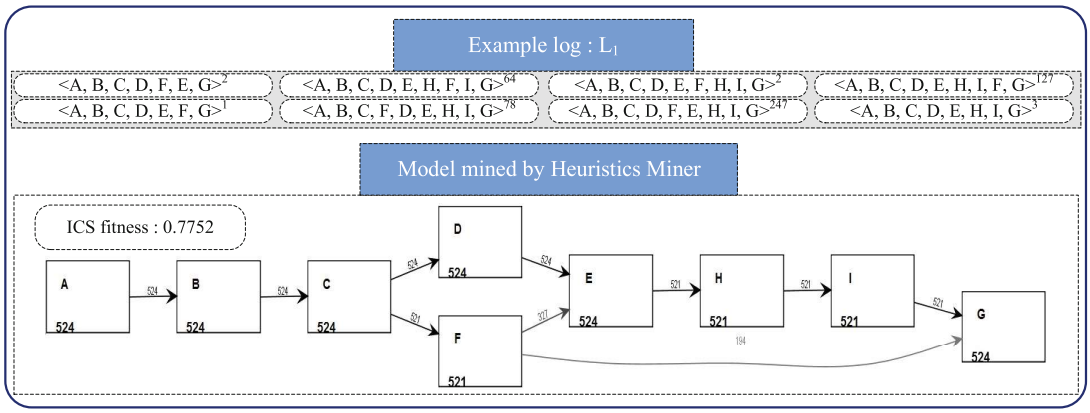

Fig. 1. The process model mined by executing Heuristics Miner on an example log L_1.

event logs and converting these found behaviours into expressible behaviours. As shown in Fig. 2, an element B_k in the process behaviour space (PBS) represents a kind of process behaviour extracted from a specific event log L. Afterwards, it is assessed whether B_k can be expressed by HM. If B_k cannot be expressed then all the process behaviours that pertain to B_k in L will be converted into expressible behaviours. Given an event log, how to build the PBS relevant to this log and how to locate the inexpressible behaviours in the PBS and then transform them into expressible behaviours for HM are the main problems that this paper is going to solve.

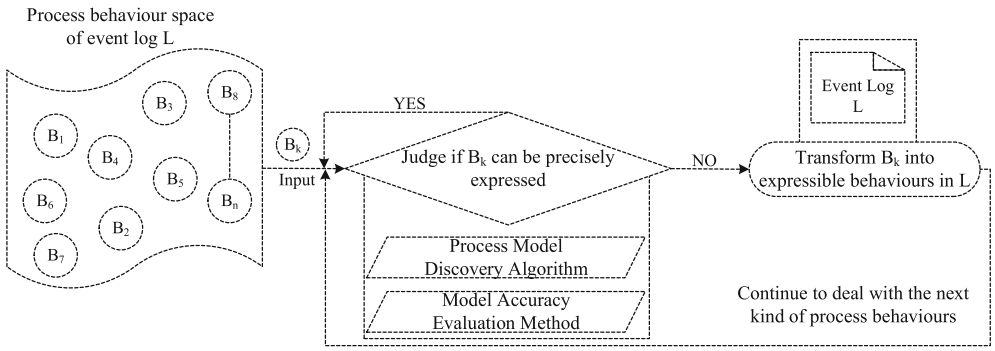

Fig. 2. The process for dealing with the inexpressible process behaviours in event logs.

2 Approach Design

In this section, some important basic notations and concepts are introduced in Subsect. 2.1. The details of the proposed method are elaborated in Subsects. 2.2, 2.3, 2.4 and 2.5.

2.1 Notation

Let L^+ be the set of event logs, $\Omega : L^+ \rightarrow M^+$ be a BPMD algorithm, where M^+ is the set of process models. $\Phi : (M^+, L^+) \rightarrow V^+$ represents a process

model fitness evaluation mechanism which gets a process model together with its relevant event log as input and creates an assessed value from V^+ (the set of all possible values output by Φ) as output.

Definition 1 (Direct Activity Relation). Let SA_L be the set of activities from an event log L. Symbol $\succ_L$ represents a direct relation between two activities from SA_L. For two activities a and b from SA_L, $a \succ_L b = true$ if $|a \succ_L b| > 0$, where $|a \succ_L b|$ is the number of times that a is directly followed by b in L.

2.2 Build Process Behaviour Space (PBS)

How to effectively extract and organise process behaviours is the first challenge encountered by our approach. In this subsection, we present a method for collecting and structuring the process behaviours from event logs based on two concepts: *behaviour-related activity* and *behaviour-related sub-trace*.

Definition 2 (Behaviour-Related Activity). Let SA_L be the set of activities for event log L. Symbol $\Rightarrow_L$ represents a behaviour-based relation between any two activities from SA_L. For two activities $a, b \in SA_L$, $a \Rightarrow_L b = true$ if $a \succ_L b = true$ or $b \succ_L a = true$ and b is also called a behaviour-related activity (BRA) of a.

Definition 3 (Behaviour-Related Sub-trace). Let SA_L be the set of activities for event log L. Let t be a trace from L, $st \sqsubseteq t$ be a sub-trace of t and SA_{st} be the set of activities for st. Given an activity $a \in SA_L$, st is a behaviour-related sub-trace (BRST) of a if $\forall b \in SA_{st} \wedge b \neq a$ such that $a \Rightarrow_L b$ and $a \in SA_{st}$. And st is a maximal behaviour-related sub-trace (MRST) of a if $\nexists st'$ such that st' is a BRST of a and $st \sqsubset st'$.

Let's take event log L_1 depicted in Fig. 1 as an example. According to Definition 2, activity F has six BRAs which are activity D, E, H, I, G and C. Seven kinds of MRSTs for activity F can be discovered from L_1 (as shown in Fig. 3) according to Definition 3. It can be seen that every MRST of F contains activity F and all the other activities in the MRST are BRAs of F.

In our technique, the process behaviours recorded in an event log are divided into several groups where each group is relevant to a single activity from this log and the process behaviours for a group are stored in the MRSTs of its related activity. For instance, the PBS for log L_1 consists of nine sets of MRSTs where each set of MRSTs is relevant to a specific activity from L_1 (as shown in Fig. 3). Our technique is devised to detect each set of MRSTs stored in PBS iteratively for finding and converting inexpressible process behaviours for HM.

2.3 Activity Ranking

In this subsection, an activity ranking method is put forward in which the MRSTs related to the higher-ranked activities will be handled before the MRSTs

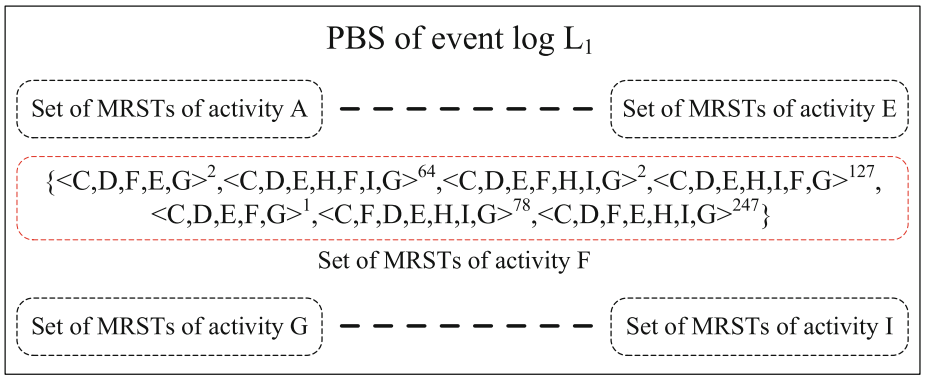

Fig. 3. The PBS built for the example event log L_1.

relevant to the lower-ranked activities. The proposed activity ranking method is based on two concepts: *behaviour-related activity weight* (BAW) and *activity ranking weight* (ARW). Given an activity a from event log L, the BAW of a is defined as:

$$BAW_a = |\bullet \succ_L a| + |a \succ_L \bullet|. \tag{1}$$

In Eq. 1, $|\bullet \succ_L a|$ represents the total number of activities from L that are directly followed by a at least once and $|a \succ_L \bullet|$ represents the total number of activities which directly follow a in L at least once.

Axiom 1.
The larger the BAW of an activity from an event log L is, the more possible this activity will be the main factor that leads to the inexpressible process behaviours in log L.

According to Axiom 1, the BAW is employed by our technique to quantify the complexity induced by an activity on its related process behaviours (i.e. the MRSTs of this activity) recorded in the relevant event log. However, Axiom 1 might not be applicable in all situations, e.g. an activity that only joins a concurrent behaviour may also have a large BAW but it will not cause any inexpressible process behaviour as long as the utilised BPMD algorithm can model concurrency. These additional situations are also considered in our approach proposed in the next subsection.

Let a be an activity from event log L, the ARW (activity ranking weight) of a is defined as:

$$ARW_a = \frac{BAW_a}{BAW_{max}} \times \frac{|a|}{OF_{max}}. \tag{2}$$

where BAW_{max} stands for the BAW of a particular activity from L which has the largest BAW, $|a|$ stands for the occurrence frequency of activity a in log L and OF_{max} represents the occurrence frequency of an activity from L which has the largest frequency of occurrence. According to Eq. 2, the ARW of an activity consists of two parts. The fist part is based on the BAW of this activity while

the second part considers the influence level of activity on the fitness of the final mined model. In our method, the larger the ARW of an activity is, the higher ranking the activity will have.

2.4 Detection and Conversion of Inexpressible Process Behaviours

In this subsection, we first formalise a new concept called *environment item*. Then, a new method named DCIB is proposed.

Definition 4 (Environment Item). Let SA_L be the set of activities from event log L, activity a, b and c are three activities from SA_L, the tuple (b, c) is an environment item (EI) of activity a if $\exists t \in L$ such that $<b, <a \ldots>, c> \sqsubseteq t$, where t stands for a trace from L and $<a \ldots>$ represents a sub-trace that only consists of activity a (one or more).

According to Definition 4, the activity F in the example log L_1 has six EIs which are EI (D,E), (H,I), (E,H), (I,G), (E,G) and (C,D).

Axiom 2.
Converting an activity into a new activity under appropriate environment item will help reduce the complex process behaviours aroused by this activity.

Figure 4 shows an event log L_2 generated by converting activity F under environment (D,E) into a new activity F1 and converting F under environment (H,I) into a new activity F0 in log L_1. As illustrated in Fig. 4, the process model mined from the newly created log L_2 has a much higher fitness than the model mined from L_1. The main reason for such an improvement on fitness is that the conversion of activity F under environment (D,E) and (H,I) transforms the inexpressible (complex) process behaviours (related to F) for HM into expressible (simple) process behaviours.

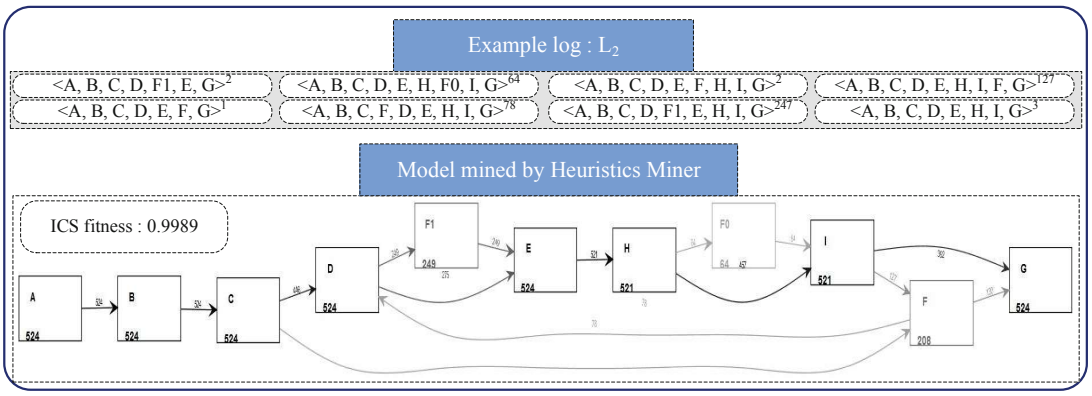

Fig. 4. The process model mined from newly generated log L_2.

The method DCIB (that will be) proposed in this subsection is able to assist in detecting the suitable EIs for a specific activity under which transforming the

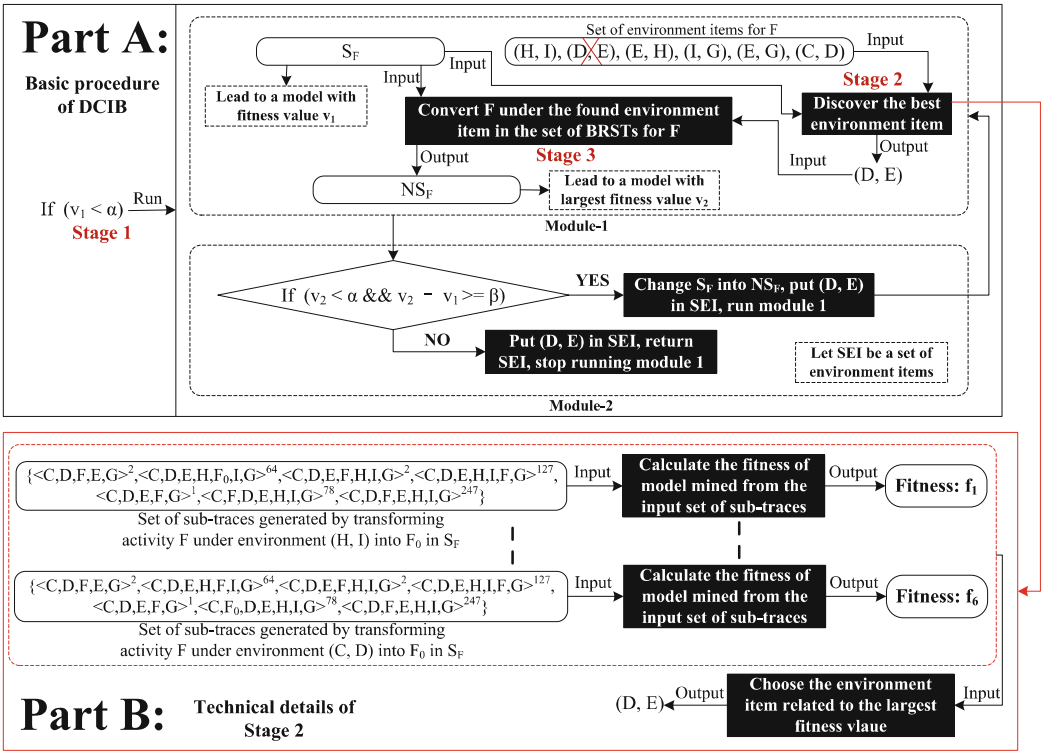

Fig. 5. The basic procedure for technique DCIB.

activity into new activities can help simplify the complex process behaviours led by this activity. The storage structure of process behaviours in PBS provides a basis for DCIB to fulfil such a function. Specifically speaking, for each time DCIB discovers the qualified EIs for a certain activity by detecting its MRSTs stored in the relevant PBS[1]. Let's take the activity F from log L_1 as an example to explain the primary procedure for DCIB. Let S_F stand for the set of MRSTs for activity F (the details of S_F are exhibited in Fig. 3), v_1 represent the fitness value of the process model mined from S_F, SEI be a set of EIs, α be a target fitness and β be a minimum fitness improvement threshold. As illustrated in part A of Fig. 5, DCIB contains three stages and two modules. In stage 1, it judges whether v_1 is less than the target fitness α. If it is not, DCIB stops because the negative influences aroused by the inexpressible process behaviours related to activity F is acceptable. In stage 2 and stage 3 (that belong to Module–1), DCIB searches for the best EI ((D,E) in our example) of activity F among all its EIs (mentioned above) under which converting F into a new activity will generate a new set of MRSTs NS_F for F where the fitness of the model mined from NS_F has the largest value (i.e. v_2) compared with the models mined from other set of MRSTs generated by transforming F under other EIs of F. Then the found EI (D,E) is removed from the original set of EIs for activity F. The part B of Fig. 5 shows the details for realising the stage 2 of DCIB. In Module–2, DCIB

[1] Detecting the qualified EIs in the MRSTs of an activity instead of in the whole process behaviours recorded in the event log will greatly reduce the detection time (i.e. the algorithm efficiency is improved).

judges if $v_2 < \alpha$ and $v_2 - v_1 \geq \beta$. If it is, put the found EI (D,E) in SEI, replace S_F by using NS_F and continue running Module–1. If $v_2 \geq \alpha$, DCIB stops because the EIs found so far are enough to help decrease the negative influence led by the complex process behaviours aroused by F to a certain extent (indicated by α). DCIB will also stop running if $v_2 - v_1 < \beta$. Because adding new activities will help improve the accuracy of the potential model but may also increase the complexity of the model at the same time. It is not worth to add new activities if the model fitness cannot be improved to a certain extent.

2.5 A Heuristic Method for Improving the Fitness of Mined Business Process Models (HIF)

In this subsection, we propose a heuristic method named HIF based on the discussions in the former subsections for improving the fitness of process models mined through HM. The details about HIF are shown in Algorithm 1.

Algorithm 1. HIF

Input: an event log L, the set of activities SA_L of log L, a target fitness α, a model fitness improvement threshold β, a threshold μ for the number of newly added activities.

Let x be a variable of type Integer.
Let PBS_L be a set of sets of behaviour-related sub-traces.
Let *SEI* be a set of environment items.
Let *LRA* be a list of ranked activities.
Let *CPBS* be the PBS building method introduced in Subsection 2.2.
Let *AR* be the activity ranking method introduced in Subsection 2.3.

1: $SEI \leftarrow null$, $x \leftarrow |SA_L|$ # x records the total number of activities in log L
2: $PBS_L \leftarrow CPBS(L, SA_L)$ # create the process behaviour space for event log L
3: $LRA \leftarrow AR(SA_L)$ # rank activities from SA_L
4: **repeat**
5: get the activity a that has the highest ranking out of *LRA*
6: $SEI \leftarrow$ DCIB$(a, PBS_L, \alpha, \beta)$ # detect qualified environment items for activity a
7: **repeat**
8: get an environment item ei out of *SEI*
9: convert activity a under ei into a new activity in log L
10: put the newly generated activity for a in SA_L
11: **until** $((|SA_L| - x) == (\mu \times x)$ || $\Phi(\Omega(L), L) \geq \alpha$ || $|SEI| == 0)$
12: **until** $((|SA_L| - x) == (\mu \times x)$ || $\Phi(\Omega(L), L) \geq \alpha)$

Output: a business process model with higher accuracy $M = \Omega(L)$.

Firstly, the number of activities in the given log L is stored in variable x (step 1). Then, algorithm HIF creates the PBS for log L (step 2) and also a ranking list LRA for the activities in L (step 3) according to the method proposed in Subsect. 2.3. Next, HIF chooses an activity a which has the highest ranking in LRA and removes a from LRA (step 5). The inexpressible process

behaviours aroused by a will be first handled which means that HIF always give priority to the main contradiction. Then, HIF searches for the qualified EIs for activity a through technique DCIB (introduced in Subsect. 2.4) and the found EIs are put in set SEI (step 6). Afterwards, for each environment item $ei \in SEI$, HIF changes the activity a into a new activity under environment ei in log L (this action will help improve the fitness of the model mined from L as demonstrated in the last subsection), removes ei from SEI and put the newly generated activity in the set of activities SA_L for log L (steps $7-11$). In HIF, a threshold μ is used to limit the number of the newly added activities because adding too many new activities might increase the complexity of the final model. If the number of the newly added activities is larger than $\mu \times x$ then HIF stops (step 11 and 12). The activity ranking procedure described in step 3 makes sure that the accuracy of the mined model could be improved as much as possible under the limitation given by μ. Furthermore, if the fitness of the model mined from L is larger than or equal to the given target fitness α then HIF also stops (step 11 and 12). Finally, a process model M with higher fitness value is output by HIF.

3 Evaluation

In our experiment, the ICS fitness [4] is used for evaluating the accuracy of mined models. The Extended Cardoso Metric (E-Cardoso) [5] and Place/Transition Connection Degree (PT-CD) [2] are employed for evaluating the impact of our method on the complexity of the mined models. We tested the effectiveness of HIF on four real-life event logs: the repair log (Repair) from [1], the log of the loan and overdraft approvals process (LOA) from Business Process Intelligence Challenge (BPIC) 2012, the log of Volvo IT incident and problem management (VIPM) from BPIC 2013 and log of CRM process (MCRM) from [2].

Table 1. Evaluation results for the proposed model fitness improvement method HIF.

Model	ICS-fitness	Precision	E-Cardoso	PT-CD	Event types	Time (s)
M–Repair	0.6768	0.426	31	2.3656	12	
$M\text{–}Repair_N$	**0.9989**	**0.7252**	**49**	**2.3611**	**16**	**6.39**
M–LOA	0.7878	0.6717	148	3.1478	36	
$M\text{–}LOA_N$	**0.9826**	**0.6511**	**178**	**2.9149**	**47**	**399.71**
M–MCRM	−0.1379	0.7454	64	2.4545	22	
$M\text{–}MCRM_N$	**0.8802**	**0.7641**	**79**	**2.3621**	**29**	**25.953**
M–VIPM	0.3594	0.86	54	2.8848	13	
$M\text{–}VIPM_N$	**0.8539**	**0.6**	**68**	**2.979**	**17**	**274.51**

In the experiment for HIF on the four logs, the target fitness α is set to 1, the model fitness improvement threshold β is set to 0.03 and the threshold for the

number of newly added activities μ is set to 0.3. Table 1 shows the evaluation results. In Table 1, *M–Repair* represents the model generated by directly mining the original event log Repair and $M{-}Repair_N$ stands for the model output by HIF (the same applies to the other models). It can be seen that the technique HIF can improve the fitness of the mined models to a large extent, while for most of the models output by HIF their precision and complexity are kept within an acceptable range compared with their original models. The model $M{-}Repair_N$ only has four more activities than the model $M{-}Repair$ but the fitness for $M{-}Repair_N$ has been greatly improved (the same to the model $M{-}VIPM_N$). This benefits from the activity ranking method presented in Subsect. 2.3.

4 Conclusion

In this paper we proposed the technique HIF for helping improve the fitness of the models mined from event logs. The proposed technique is able to detect the inexpressible process behaviours recorded in event logs for HM and transform the found behaviours into expressible behaviours. As a result, more fitting process models can be generated. Through the evaluation results from Sect. 3 we demonstrated the effectiveness of HIF. Our future work will mainly be focused on adapting our method HIF to other BPMD techniques so as to help them mine better process models. In the meantime, we will also validate HIF on some other real-life cases.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
2. Weerdt, J.D., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
3. Weijters, A., van der Aalst, W.M.P., Alves de Medeiros, A.K.: Process mining with the heuristics algorithm. In: BETA Working Paper Series 166, TU Eindhoven (2006)
4. de Medeiros, A.A.: Genetic process mining. Ph.D. thesis, Eindhoven University of Technology (2006)
5. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for workflow nets. Inf. Softw. Technol. **51**, 610–626 (2009)
6. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75183-0_27
7. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 368–387. Springer, Heidelberg (2008)

8. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013)
9. Fahland, D., Aalst, W.M.P.: Repairing process models to reflect reality. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 7481, pp. 229–245. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32885-5_19
10. Claes, J., Poels, G.: Process mining and the ProM framework: an exploratory survey. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 132, pp. 187–198. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36285-9_19