Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

A Repairing Missing Activities Approach with Succession Relation for Event Logs

Jie Liu · Jiuyun Xu · Ruru Zhang · Stephan Reiff-Marganiec

Received: 16 Dec 2019/ Revised:12 Jul 2020/ Accepted: 11 Oct 2020

Abstract In the field of process mining, it is worth noting that process mining techniques assume that the resulting event logs can not only continuously record the occurrence of events but also contain all event data. However, like in IoT systems, data transmission may fail due to weak signal or resource competition, which causes the company's information system to be unable to keep a complete event log. Based on a incomplete event log, the process model obtained by using existing process mining technologies is deviated from actual business process to a certain degree. In this paper, we propose a method for repairing missing activities based on succession relation of activities from event logs. We use an activity relation matrix to represent the event log and cluster it. The number of traces in the cluster is used as a measure of similarity calculation between incomplete traces and cluster results. Parallel activities in selecting pre-occurrence and post-occurrence activities of missing activities from incomplete traces are considered. Experimental results on real-life event logs show that our approach performs better than previous method in repairing missing activities.

Keywords Process mining · Information system · Activity relation matrix · Incomplete event logs

1 Introduction

1.1 Background

Business Process Management (BPM) plays an increasingly important role in today's enterprises because of its advantages of cost savings, improvement of work quality and process optimization. The existing methods and tools in BPM can support the discovery, analysis and improvement of business process [31]. The execution of a business will be documented by the enterprise, and eventually will be converted into event log format that contains some attributes such as case id, activity, timestamp, resource, etc. Event logs recorded in modern information system, as the starting point of process mining, provide the precondition for process model discovery, conformance check and enhancement [3]. On the one hand, process discovery in the process mining domain is the crucial learning task whose goal is to automatically discover a process model represented by Petri nets or BPMN in order to analyze actual business execution process [1]. Based on the analysis of event logs, more and more scholars are working on different methods to obtain high-quality process models [13, 21, 38, 5, 2]. On the other hand, conformance checking is associating events in the event log with activities in the process model in order to expect to reveal and diagnose commonalities and differences between the modeling behaviors and the observed behaviors[24, 25]. Therefore, event logs should be treated as "first-class citizens"[4].

Jie Liu
College of Computer Science and Technology, China University of Petroleum(EastChina), Tsingdao, China
E-mail: liujie_ran@163.com

Jiuyun Xu: corresponding author
College of Computer Science and Technology, China University of Petroleum(EastChina), Tsingdao, China
E-mail: jyxu@upc.edu.cn

Ruru Zhang
The ChinaMobile(suzhou) Software Technology Company, Suzhou, China
E-mail: zhangruru@cmss.chinamobile.com

Stephan Reiff-Marganiec
School of Electronics, Computing and Maths, University of Derby, Derby, UK
E-mail: sreiffmarganiec@ieee.org

1.2 Motivation

However, it is worth noting that process mining techniques like α-algorithm [1], Genetic Miner [27], ILP Miner [19],Heuristic Miner [33], in most situations, assume that the resulting event logs can not only continuously record the occurrence of events, but also contain all event data. As we all know, an organization's business execution processes are becoming more complex and flexible to accommodate different business requirements, which will result in recording massive volumes of data in their own information systems. In this case, there is no guarantee that all data can be logged in the event logs. For example, event logs from Internet of Things(IoT) devices fails to store all running data for some reason such as temporary loss of network signal, a fault of a sensor or synchronization problems. To our knowledge, few researchers have studied the incomplete event log problem in process mining.

If we apply traditional process discovery algorithms to obtain process models from incomplete event logs,the resulting process models are likely to deviate from the actual business process models. For example, in an IoT system developed by an IT company for elderly people, sensors collect information such as heart rate, location, room temperature, carbon dioxide, and lighting to sent to fog nodes with computing power. Then a series of data processing task is carried out in the cloud and transmitted to the Web application through the Internet. Stakeholders, family members, caregivers, health care professionals and emergency services, can monitor the elderly's behavior in real time. The entire business process is shown in the figure1.

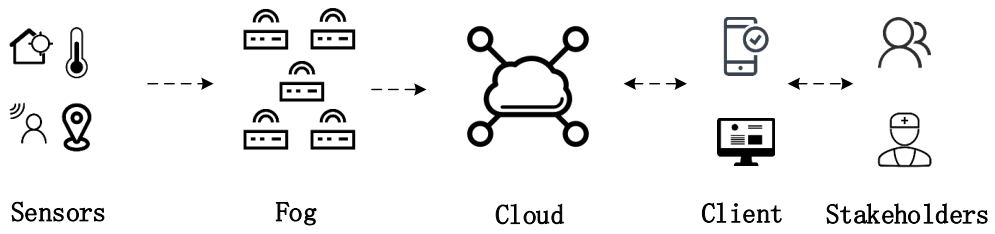


Fig. 1: data processing architecture distributed with the different locations

In order to represent the process of this business logic succinctly and clearly, we assume that there are ten active tasks, that is, task(A) collecting information, tasks(B, C, D, E and F) fog equipment calculation, task (G) cloud computing, tasks (H, I) application devices data interface, and task (J) stakeholders accepting information. Hypothetically, the company has a complete event log with 8 activity traces denoted by L =[ABCGFIL, ABFCGIKL, ACBFGIJL, ACBGFIJKL, ABCFDGIKL, ACDBFGIKL, ABCDGFIKL, ABCDGFIJL]. Based on this event log, a process model(see Figure 2) can be discovered from this event log using a fuzzy mining algorithm [16] In this Figure, the numbers in each box mean the edge cutoff in ProM 6.7 [1], which is a fuzzy process mining kit for process model analysis, to generate the process model shown. However, Data may be lost due to unstable transmission signal of fog devices. When in the fog device C and D faults occur on the penultimate and last track respectively, the event log is incomplete. Consequently, we get another process model based on this incomplete log, which is shown in Figure 3. Comparing the two process models, visually we see that two red arcs disappear in Figure 2 and two green arcs appear in Figure 3. Besides, the corresponding numbers in the other activities except A and J are different. Therefore, this will have some negative impact on the analysis of the process model for business executives or domain experts. For example, there is a B to C process in the complete event log, but the process model from incomplete log can not express this execution process.

[1] http://www.promtools.org/doku.php?id=prom67

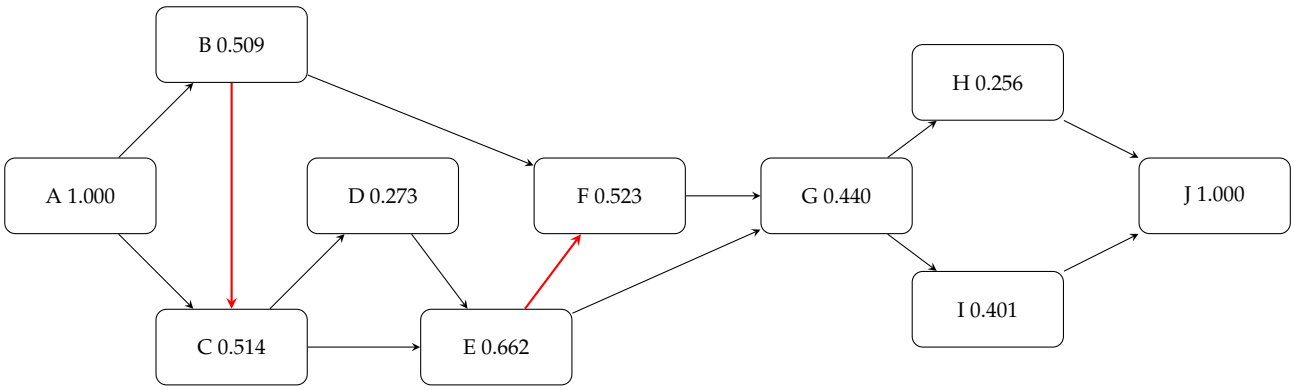


Fig. 2: Process model based on a complete event log.

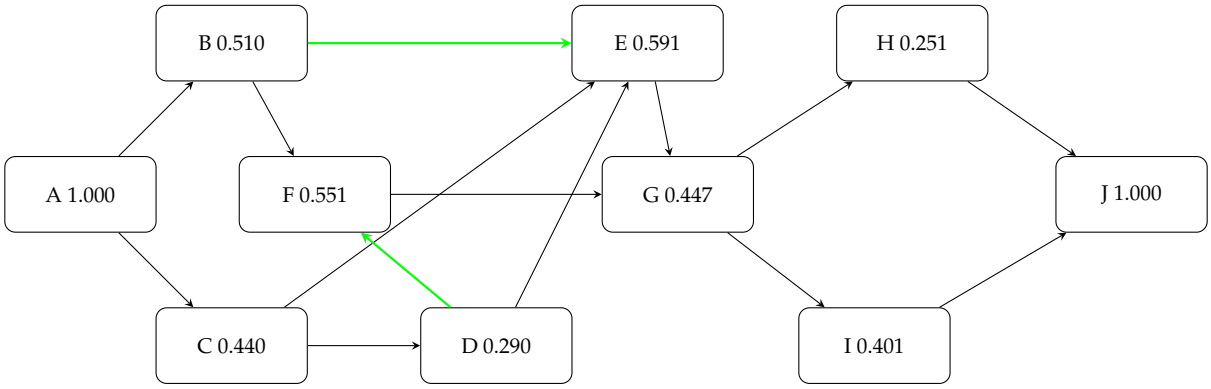


Fig. 3: Process model based on an incomplete event log.

Let's take another look at the results of an incomplete event log and complete event log on conformance checking, respectively. First of all, we use one of the simple complete event log [2] containing eight activities and six different traces (42 events in total) as input for conformance analysis. Secondly, we turn this complete log into an incomplete log by randomly deleting any four of the 42 events. To identify which are the most right traces for the conformance, we finally apply a conformance approach called *Replay a Log on Petri Net for Conformance Analysis* in ProM Lite 1.2 [3], which is a lightweight tool similar to ProM 6.7, and show some results in the table 1. As can be seen from table 1, all values except for the calculation time decrease. Of the four quality measures (fitness, simplicity, precision, generalization) in process mining, fitness is the most relevant measure of conformance. When conformance analysis is carried out for the same process model from complete and incomplete log, fitness for the process generated from the incomplete log is lower, meaning that we must expect the resulting process to be flawed.

Table 1: Results of conformance anlysis from complete and incomplete event logs

Property	Complete Event Log	Incomplete Event Log
Trace Fitness	1.0	0.94
Trace Length	7.0	6.33
Max Fitness Cost	12.0	11.33
Max Move-Log Cost	7.0	6.33
Calculation Time(ms)	1.67	1.67

Consider the above two aspects, a complete event log is critical to process mining techniques. In the field of data mining, there are many methods to repair missing data, such as regression, Bayesian formalization, decision tree induction, etc. However, due to the complex and flexible nature of the business execution process and the concurrency, circular and causal relationships between events, the above methods cannot be directly used to repair missing data from the incomplete event log in process mining. In our previous [35] work, a method, which explores the context of full traces in the incomplete log with trace profile, was proposed to repair missing activities. However,

[2] http://www.processmining.org/event_logs_and_models_used_in_book

[3] http://www.promtools.org/doku.php?id=promlite12

trace profile the initial relationships among activities are extracted from trace profile, without further analysis of circular and causal relationships between events, the accuracy of proposed method is one of problems which we are addressing in this paper.

In this paper, we focus on incomplete event logs and propose a method of repairing missing activities to restore the most realistic event log. The goal is that after repair all existing process mining techniques can be used with confidence. Initially, we extract complete traces and incomplete traces from an event log. Dealing with these complete traces will result in some relational groups and each trace, including incomplete traces, is represented by multiple relation groups. Unlike our previous representation of the event log, first, we transform a log into a matrix by activity succession relationships rather than trace profiles. Next the complete traces are divided into different clusters, and then we assign each incomplete trace to the most similar clustering results. The concurrency relation of activities is considered when looking for direct pre-occurrence and post-occurrence activities of missing activities. Finally, the missing activities in incomplete traces are repaired according to the relationship between the activities from the event log. The accuracy of our approach is better than others and has been empirically evaluated and compared on different real-life event logs.

The contributions of this paper are as follows:

- A process repair method which is based on the succession relation of events is proposed in this paper, which is quite different from our previous paper which is based profile clustering.
- Experimental results on real-life event logs show that the accuracy of our approach is better than previous method in repairing missing activities.

The rest of the paper is structured as follows: Section 2 discusses related work. In Section 3, we provide the basic concepts used in this paper. Our method for predicting missing activities from incomplete traces is described in Section 4. Experimental results and comparisons on several real-life event logs with respect to previous method are provided in Section 5. Section 6 draws conclusions and discusses future work.

2 Related Work

Process mining builds an important bridge between data mining and business process modeling and analysis. The purpose of process mining is to extract useful information from event logs, so that actual business processes can be discovered, and improved. Aalst et al.[1] proposed an α algorithm to mine a simple event log. It is one of the first algorithms to deal with concurrent processes in process mining. The disadvantage of the α algorithm is that it fails to deal with noise and loop structures in an event log. The authors of [8, 34] extended the α algorithm to mine short loop processes and non-free-choice constructs. In addition, Medeiros et al. [27] introduced a framework for a genetic process mining method, which includes four main steps: initialization, selection, reproduction and termination. Genetic Miner can handle noise of log information and be easily adapted and extended. Some areas have borrowed the idea of genetic mining for achieving better quality assurance like the garment industry [20] and business process monitoring [11]. At the same time, genetic process mining is not very effective in the face of large event logs. It can take a lot of time for finding a good fitness function. Effendi et al. [12] presented a process model discovery technique by considering the overlapping rules which makes fitness without losing too much precision.

For well-structured business processes and small event logs, most process discovery algorithms can generate simple and expressive process models. As a matter of fact, an event log will record huge amounts of data due to the complexity of the business process. Based on large event logs, resulting process models can be 'spaghetti-like' and difficult to understand. Therefore, Greco et al. [14] presented the idea of trace clustering to split a complex event log into several simple sub-logs and discovery sub-process models. The key to trace clustering is the approach to convert data flow into a numerical representation. If a set of numerical representations cannot well represent the running relationship of events in the trace, the similarity between traces in the clustering results may be very low, which ultimately affects the quality of the process model. Researchers explored more methods to improve the result of trace clustering and discover accurate and simple business process models from event logs [9, 17, 10, 30, 32].

Split miner [6], based on a directly-follows graph, can identify combinations of split gateways that capture the concurrency, conflict, and causal relation between events to discovery accurate and simple business process models from event logs. In the face of some more complex event logs, the split miner method may get a bad process model. If an event log includes some low-level events, process models discovered by process mining techniques cannot be easily understood. Therefore, Mannhardt et al. [26] presented a Guided Process Discovery method (GPD) to find relation between low-level events and high-level activities based on domain knowledge. GPD can discovery a high-level process model from the abstracted event log created through alignment between activity patterns and low-level event log. In the paper, the GPD approach has been shown to be successfully applied to big event logs form complex and flexible business processes. And yet, if multiple optimal alignments appear in event groups, GPD just chooses one of them and does not give a better selection strategy. Besides, a large and complex event log with a large amount of data is more likely to miss some data [7]. In this case, they did not indicate whether the activity patterns analyzed by GPD deviated from the actual one.

Existing process discovery techniques including α algorithm [1], ILP Miner[19], Heuristic Miner[33], etc. can satisfactorily discover process models from both simple and complex event logs under different constraints. They are all based on the assumption that the event logs are complete. Vast quantities of data is recorded in the enterprise

information system every day and it is impossible to guarantee that the data can be recorded correctly and without omission. Leemans et al.[22] introduced a new algorithm to discovery block-structured process models from incomplete event logs. They improved Inductive Miner [21] to deal with incomplete logs: the divide-and-conquer step is preserved, whereas the activity partition substituted by an optimization problem from probabilistic behavioral relations. The so-called incomplete log defined in the paper is that one cannot assume to have seen all possible process executions.

Conformance checking can be used to improve compliance with business processes, organizations, and information systems in process mining. It has the ability to identify the difference between the process model and the process execution recorded in the event log. And by analyzing these deviations, one can develop work efficiency or change management system [1]. In [28], Rozinat et al. introduced an incremental approach to check the conformance of a process model and an event log based on monitoring real behavior and define multiple metrics to allow for the quantification of conformance. However, they use only an activity perspective (i.e. control-flow) on the business process and do not consider the dependencies of events or even the loss of activity in the event log.

A new method in which all perspectives were taken into account was reported in [23] to align log traces and process models for conformance checking. Sometimes, the result of alignment is not optimal so that deviations cannot be comprehensively interpreted. Based on a customizable cost function, Mannhardt et al. [25] presented a balanced multi-perspective technique for business process conformance checking. Unlike other approaches that take control-flow as a priority perspective, they balance the deviations about all perspectives from event logs to circumvent misleading results.

A majority of process mining techniques extract knowledge from complete event logs to discovery process models and check the conformance between event logs and process models. Zakarija [36] used an example to show how to select process mining techniques to discover process models when the event log data is incomplete and Zareh-Farkhady [37] also introduced a two phase approach to directly mine process models form incomplete and noisy logs, but without considering how to repair missing data form event logs. As mentioned in Section 1, when an incomplete event log is used to generate process model, the result will deviate from the actual business processes. In our previous work, we proposed an event log repairing approach based on profile clustering (PROELR). In this paper, we focus on predicting missing activities from incomplete event logs in process mining using a new method and compare the accuracy with our previous method. The goal is to restore the incomplete event log to the original record as perfectly as possible so that it can serve all process mining technology.

3 Preliminaries

This section describes some essential concepts used in the paper. An event log record some event information about the business execution process. Each event can be described by different attributes. For example, an event might have a timestamp corresponding to an activity, be executed by a specific person, have associated costs, and so on.

3.1 Event logs and re-represention of traces

An event log consists of a set of cases (traces) that are a specific event sequence. For simplicity, we often use activity names to represent events. For example, an event log $L = [\sigma_1^3, \sigma_2, \sigma_3^2]$, where $\sigma_1 = \langle$ a,b,c,d $\rangle$, $\sigma_2 = \langle$ a,b,b,b,d $\rangle$, and $\sigma_3 = \langle$ a,c,b,d,e $\rangle$, contains six traces and 27 events. In this paper, we focus on repairing missing activities in traces from incomplete log. The event log and trace are defined formally as follows:

Definition 1 ***(Trace, Event log).*** *Let A be a set of activities. A^* is a collection of all finite sequences of activties. A trace $\sigma \in A^*$ is a sequences of activities. An event log L is a multiset of traces.*

Definition 2 ***(Complete trace, Complete event log).*** *A complete trace ($c\sigma$) consists of a complete sequence of activities. A complete event log CL is a multiset of complete traces.*

Consider an event log L = [abcd, acbd, abbd, acbde, ab-def], where A is {a, b, c, d, e, f} and complete traces include $c\sigma_1 = \langle a,b,c,d\rangle$, $c\sigma_2 = \langle a,c,b,d\rangle$, $c\sigma_3 = \langle a,b,b,d\rangle$ and $c\sigma_4 = \langle a,c,b,d,e\rangle$. So the CL = [$c\sigma_1$, $c\sigma_2$, $c\sigma_3$, $c\sigma_4$].

Nowadays, vast quantities of information are generated daily from enterprise information system, which could result in the loss of some data due to various reasons such as human error and system failure so that business execution process cannot be fully recorded in the event log. Furthermore, events are not just stored in dedicated log files (XES, MXML) and may be recorded in database tables, message logs, transaction logs, server logs, and other data sources. We need to translate these events into a formal event log form as a starting point for process mining. During this transformation, data loss may also occur.

Definition 3 ***(Incomplete trace, Incomplete event log).*** *An incomplete trace $i\sigma = \langle e_1, e_2, ..., e_n\rangle$, if and only if, for $i \in$ [1,n], $\exists e_i = NULL$. An incomplete event log $IL = [\sigma_1, \sigma_2, ..., \sigma_m]$, if and only if, for $j \in$ [1,m], $\exists \sigma_j = i\sigma$.*

Consider again this event log L = [abcd, acbd, abbd, acbde, ab-def]. It is defined as an incomplete event log on account of existing an incomplete tracec $i\sigma = \langle a,b,-,d,e,f\rangle$.

Definition 4 ***(Start activities(T_s), End activities(T_e), Single loop activities(T_l)).*** *Let L be a complete event log, A be the set of activities, and σ=$\langle e_1, e_2, e_i, ..., e_n \rangle$.*

- $T_s = \{a \in A | \exists\ \sigma \in L \wedge a = e_1\}$.
- $T_e = \{a \in A | \exists\ \sigma \in L \wedge a = e_n\}$.
- $T_l = \{a \in A | \exists\ \sigma \in L \wedge a = e_i = e_{i+1}, i \in [1, n-1]\}$.

A trace represents the order in which activities occur, while does not adequately indicate the dependencies between activities. For example, we can conclude that succession relations are (a,b), (b,c), (c,d) from $c\sigma_1$. However, the CL characterizes the process where a is followed by b and c in parallel. So the pair of activities (b,c) is not succession but concurrency relation. Based on this point, we need to re-represent the event log through the direct succession relation between activities. Next, we introduce several kinds of relations based on the event log.

Definition 5 *Let L be a complete event log and A be the set of activities.*

- *Order relation($\succ_L$): $a \succ_L b$ is a order relation if and only if $\exists \sigma = \langle e_1, e_2, e_i, e_j, ..., e_n \rangle$, i,j ∈ [1, n] and $i < j$, $\sigma \in L$, a,b ∈ A so that $e_i = a$ and $e_j = b$.*
- *Concurrency relation($\|_L$): $a \|_L b$ is a concurrency relation if and only if $\exists \sigma = \langle e_1, e_2, e_i, ..., e_n \rangle, \bar{\sigma} = \langle e_1, e_2, e_j, ..., e_m \rangle$, $i \in$ [1, n-1], $j \in$ [1, m-1], $\sigma, \bar{\sigma} \in L$, a,b ∈ A so that $e_i = a$, $e_{i+1} = b$ and $e_j = b$, $e_{j+1} = a$.*
- *Transitive relation($\Rightarrow_L$): $a \Rightarrow_L c$ is a transitive relation if and only if ∃ a,b,c ∈ A so that $a \succ_L b$ and $b \succ_L c$.*

Definition 6 ***(Re-representation of trace(ρ)).*** *Let tuple $\rho = (T, \rightarrow_\sigma)$ be a new representation for the trace σ, where:*

- *$T = (T_s, T_e, T_l)$, and*
- *$\rightarrow_\sigma$ is the succession relation, i.e. $\rightarrow_\sigma = (\succ_\sigma \setminus \|_L) \setminus \Rightarrow_\sigma$.*

Based on the above definitions of special relations, a trace can be re-expressed. Let us consider the complete trace $c\sigma_1$ from complete log CL. First, we compute the order relation $\succ_L$ from all traces and get concurrency relation set {(b,c)}, and note that (b,c) is equivalent to (c,b). Activities in concurrency relation are called parallel activities. In the first step we have got the order relation set of $c\sigma_1$, namely $\succ_{c\sigma_1}$ {(a,b), (a,c), (a,d), (b,c),(b,d), (c,d)}. Second, we remove the concurrency relation {(b,c)} from the $\succ_{c\sigma_1}$. Thrid, the transitive relation $a \Rightarrow_{c\sigma_1} d$ also is removed from $\succ_{c\sigma_1}$. Finally, the $c\sigma_1 = \langle a, b, c, d \rangle$ is re-represented as the succession relation set $\rightarrow_{c\sigma_1}$= $\{a \rightarrow_L b, a \rightarrow_L c, b \rightarrow_L d, c \rightarrow_L d\}$. Similarly, we obtain $\rightarrow_{c\sigma_2}$= $\{a \rightarrow_L b, a \rightarrow_L c, b \rightarrow_L d, c \rightarrow_L d\}$, $\rightarrow_{c\sigma_3}$= $\{a \rightarrow_L b, b \rightarrow_L b, b \rightarrow_L d\}$, $\rightarrow_{c\sigma_4}$= $\{a \rightarrow_L b, a \rightarrow_L c, b \rightarrow_L d, c \rightarrow_L d, d \rightarrow_L e\}$. The set of succession relation is used in trace clustering and predicting missing activities.

3.2 Traces clustering

Greco et al.[14] first proposed an approach based on the traces clustering to handle complex and large event logs from flexible environments to discover accurate and expressive sub-models. Using the idea of traces clustering, we first cluster the complete event log so that similar traces are clustered in the same cluster, and then calculate the distance between the incomplete trace and each clustering result. Song et al.[29] introduced log profiles to partition event logs. However, each row in the activity profile defines the number of times each activity in trace and does not take into account the relation between activities.

In this paper, we construct an activity relation profile, where each row corresponds to the succession relation vector of one trace. A trace is transformed into a succession relation vector by setting corresponding relation to 1 if it exists in the trace and 0 otherwise.

Based on activity relation profile, we can apply any clustering algorithm to separate complex event logs. Four clustering algorithms including K-means, Quality Threshold, Agglomerative Hierarchical Clustering, and Self-Organizing Maps(SOM) are introduced and the conclusion is drawn that SOM algorithm performs better than others in[29].

It is well known that SOM is one of Artificial Neural Network algorithms and can conduct unsupervised learning classification for high dimensional data to produce a low-dimensional representation [18]. SOM is called a natural method of dimension reduction and with this property it can be employed for dividing the log on the basic of activity relation matrix. The purpose of using traces clustering in our paper is to find the most similar traces set for the incomplete trace, in which the missing activity rather than in the entire traces is predicted.

3.3 Similarity calculation

By using the SOM algorithm, we can get several sets of different clustering results that are represented by C containing some similar traces. We define the average succession relation vector for each cluster C in order to calculate similarity between incomplete trace and clusters as the follow formula.

$$\overline{V} = \frac{\sum_{i=1}^{|C|} v_i}{|C|} \tag{1}$$

Notice that $|C|$ denotes the number of traces in the cluster and v_i is a succession relation vector in a trace from the cluster result.

In this paper, we choose Euclidean distance to calculate similarity between incomplete traces and clusters, which is defined as follow:

$$ED(I\sigma, \overline{V}) = \sqrt{\sum_{j=1}^{n} |I\sigma_j - \overline{V}_j|^2} \tag{2}$$

$I\sigma$ represents the active relation vector of the incomplete trace iσ. It is crucial that we not only consider the distance measure but also the number of traces in the cluster for calculating similarity in our approach. In a case where any two ED are close, the final similarity of a missing trace and cluster result C is related to the number of elements in the cluster, namely, the more the value is, the more similar it is. We therefore use the following formula to determine similarity: $FS = \alpha * ED + \beta * |C|$, where α(resp. β) is set to 0.4(resp. 0.6) [35]. This formula is guided by a distance threshold(dt). If the difference between any two ED is less than set threshold manually, we will calculate FS to determine which cluster the current $i\sigma$ is assigned to. Otherwise, we only use the Euclidean distance to decide the similarity.

4 Our Approach

We already observed that it is difficult to guarantee completeness of the data recorded in the real-life event logs from modern complex information systems due to reasons such as system failure and human error. If an event log including missing data is used for process discovery and conformance checking, there will be some deviation between the results and actual business model. Therefore, ensuring that the event log is complete is critical for process mining.

Our approach for repairing missing activities in event logs consists of the following steps: 1) pre-process an event log to get a complete and incomplete event log respectively; 2) cluster the complete event log; 3) match missing traces from incomplete event log to clusters; 4) repair missing activities in the incomplete traces; and 5) mine process models and analyse conformance with complete event log. The framework of repairing missing activities is shown in Figure 4. Our approach has two main steps: one is the representation of event logs, and the other is the selection of pre-occurrence and post-occurrence of the missing activity, which are covered in detail in the following sections.

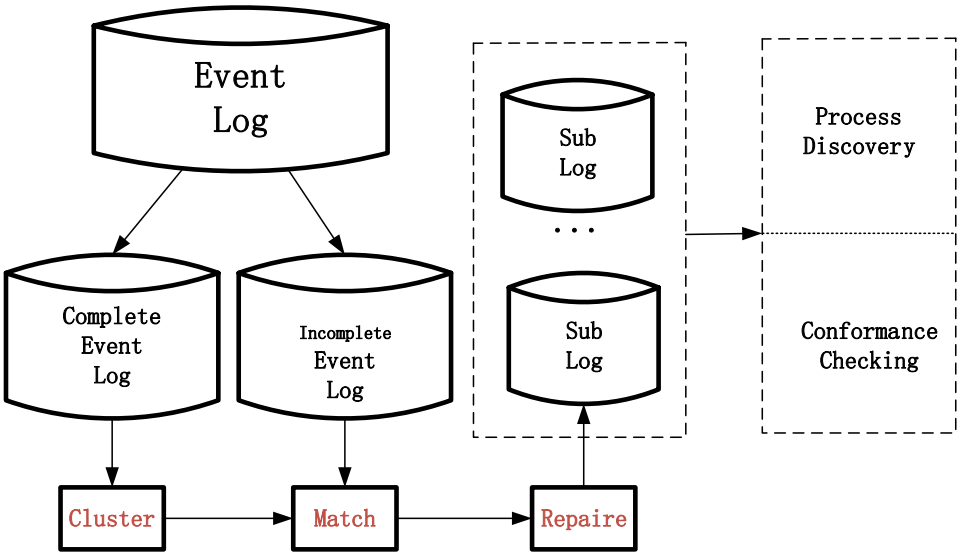


Fig. 4: The framework for repairing missing activities.

4.1 Transforming An Event Log into A Succession Relation Matrix

Given an event log, we first split it into a complete event log and an incomplete event log. Then, the event log needs to be converted to a matrix for clustering and matching operations. Unlike the previous log representation, our method

uses the activity succession relation to represent the log. From Definition 6, we can re-represent each trace from the event log. For example, the event log L = [abcd, acbd, abbd, acbde] is transformed into a activity relation profile as shown in Table 2. The following processes of clustering and matching are similar to our previous method. The event log is clustered into several different sub-logs by using SOM algorithm based on activity succession relation matrix. Incomplete traces are matched to the most similar sub-logs. As discussed in Section3.3, determining the similarity calculation between the incomplete trace and the cluster is crucial, because if the incomplete trace cannot be accurately assigned to the most similar clustering result, it may lead to the prediction of the erroneous missing activity.

Table 2: Activity succession relation matrix from the event log CL.

Trace	Succession relation					
	$a \rightarrow_L b$	$a \rightarrow_L c$	$b \rightarrow_L d$	$c \rightarrow_L d$	$d \rightarrow_L e$	$b \rightarrow_L b$
$c\sigma_1$	1	1	1	1	0	0
$c\sigma_2$	1	1	1	1	0	0
$c\sigma_3$	1	0	1	0	0	1
$c\sigma_4$	1	1	1	1	1	0

4.2 Selecting Direct Pre-occurrence and Post-occurrence activities of missing activities in Incomplete Traces

After determine which cluster is most similar to the incomplete trace, we start to repair the missing activity in the incomplete trace according to the succession relation between the activities in the traces of the cluster. When we choose the pre-occurrence and post-occurrence activities of missing activity from missing trace, there are three situations:

1. If the pre-occurrence and post-occurrence activities directly are both not parallel activities, they will be chosen as pre-occurrence and post-occurrence activities directly of missing activity in an incomplete trace.
2. If the pre-occurrence activity directly is a parallel activity, then select another pre-occurrence activity in front of it until this activity is not a parallel activity.
3. If the post-occurrence activity directly is a parallel activity, then select another post-occurrence activity behind it until this activity is not a parallel activity.

For the incomplete trace $i\sigma = \langle a, b, -, d, e, f \rangle$, since parallel activities include b and c in the event log L, the pre-occurrence and post-occurrence activities are a and d respectively. We define a variable(P) to hold the pre-occurrence and post-occurrence activities of the missing activity which are parallel activities for the event log. If activities in the predicted candidate activity set exist in P, we will remove them.

The main consideration based on the predecessor and successor activities method is to count the most frequent activities from the candidate activities as missing activities in our previous method(PROELR) [35]. However, the method based on the succession relation between activities(SRBA) considers the direct dependencies of activities in the event log. Analogously, we also first compute the direct pre-occurrence activity(a) and post-occurrence activity(d) of missing activity in the incomplete trace. Secondly, and in contrast to the approach in which $a\cdot$ and $\cdot d$ of all traces are calculated in a cluster, this approach allows for the direct succession relations between activities and generates a collection of candidate activities. Algorithm1 describes the specific solving steps. Finally, an activity from the candidate activity set could be selected as the missing activity.

The number of activities in the candidate activity set can be 0 and more than 1. For the former, we randomly select a trace from the cluster and then determine the missing activity. We chose the most frequent succession activity to determine the missing activity for the latter. In our experiment, however, the probability of these two cases happening is small.

Let's introduce the example event log L and look at Algorithm 1 again. Because we figured out that $X = a, Y = d$ and $P = \{b\}$ (line 2), the candidate activity set $xy = \{b, c\}$ (line 4-10). Afterwards, we remove activity b from the candidate activity set. So the activity c is considered a missing activity in the incomplete trace. The missing activities in the trace are completed by our method. This allows process mining techniques to use event logs as their inputs with 'greater confidence'.

5 Experimental Result Evaluation

To illustrate the effectiveness of the proposed approach for several real-life event logs(Lfull[4],Hospital Billing[5],BPI Challenge 2012[6],and Conformance Checking Challenge 2019(CCC19)[7]) and compare the PROELR and SRBA algorithms.

[4] http://www.processmining.org/event_logs_and_models_used_in_book
[5] https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
[6] https://data.4tu.nl/repository/uuid:3926db30-f712-4394-aebc-75976070e91f
[7] https://data.4tu.nl/repository/uuid:c923af09-ce93-44c3-ace0-c5508cf103ad

Algorithm 1: Based on the succession relation between activities in the log

Input: Candidate clustering results of missing traces
Output: Candidate activity set xy

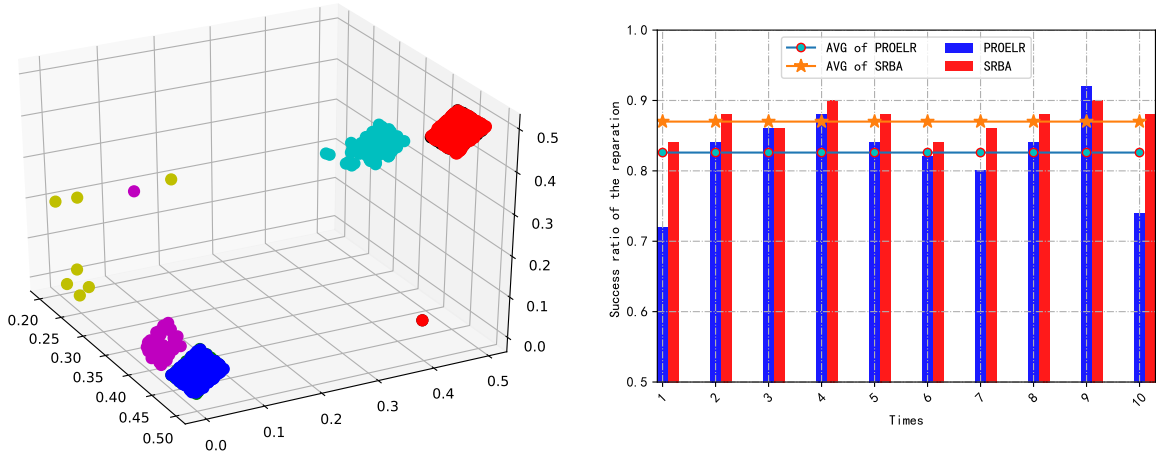
C: The most similar clustering result of $i\sigma$ selected;
X, Y: The direct pre-occurrence and post-occurrence activities of the missing activity in $i\sigma$ found;
$\rightarrow_\sigma$: The set of all direct succession relations from C to be computed;
if *X is not NULL and Y is not NULL* **then**
    $xy \leftarrow$ compute the direct successor of X and direct predecessor of Y from $\rightarrow_\sigma$;
    if *len(xy)==2 and X in xy and Y in xy* **then**
        if *X, Y not in T_l* **then**
            $xy \leftarrow xy \backslash X \backslash Y$;
    else if *len(xy)>2* **then**
        $xy \leftarrow xy \backslash X \backslash Y \backslash T_s \backslash T_e$;
else if *X is NULL and Y is not NULL)* **then**
    $xy \leftarrow$ and direct predecessor of Y from $\rightarrow_\sigma$;
    if *len(xy) ==0 and s in T_l and s != Y* **then**
        $xy \leftarrow add(s)$;
    else
        $xy \leftarrow xy \backslash Y$; $y \leftarrow set()$;
        if *s in T_l and s in xy* **then**
            $y \leftarrow add(s)$;
        if *len(y) != 0* **then**
            $xy.clear()$; $xy \leftarrow y$;
else
    $xy \leftarrow$ and direct successor of X from $\rightarrow_\sigma$;
    if *len(xy) ==0 and s in T_l and s != X* **then**
        $xy \leftarrow add(s)$;
    else
        $xy \leftarrow xy \backslash X$; $x \leftarrow set()$;
        if *s in T_l and s in xy* **then**
            $x \leftarrow add(s)$;
        if *len(x) != 0* **then**
            $xy.clear()$; $xy \leftarrow x$;

With these event log sources, some comparing experiments conducted are as followings. First, the complete event logs are adopted from all of above data sources, and deleting one of activities from the complete event traces are to get produces the corresponding incomplete event log traces. Next, we describe in detail the calculation of the repair missing activities on the real-life event log Lfull. The event log described in Table3 contains 1391 traces and 7539 events.

Table 3: Some basic information about the Lfull event log

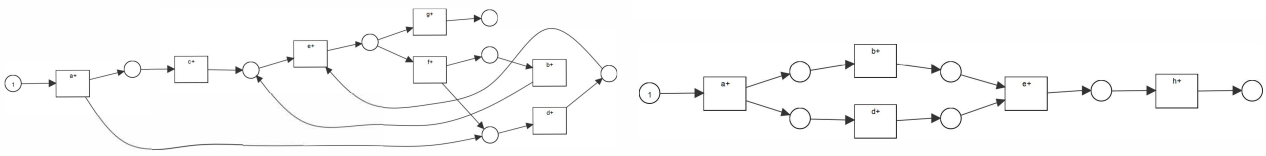
Traces	1391	Events	7539
Activities	8	Start Activities(T_s)	a
End Activities(T_e)	g,h	Single loop Activities(T_l)	None
Order relation($>_L$)	1391	Concurrency relation($\|\|_L$)	(b,d),(c,d)
Direct successor activities	16	Succession relation($\rightarrow_L$)	38

Since the event log is complete, an incomplete log is obtained from the complete event log by randomly deleting one of events for our experiment. We select 50 traces from the 1391 traces and delete one of the activities in each trace, both of which are randomly chosen, which resulted in an incomplete log containing the missing 50 activities. First, we re-represent the traces by Definition 4, 5 and 6 from the complete log including 1341 complete traces and then we get a succession relation matrix(1341*38). Second, by using the SOM algorithm, we obtain seven different clustering results shown in Figure 5(a). Third, calculate the most similar clustering results for each incomplete trace according to the introduction in Section 3.3. For example, the incomplete trace id 16 is most similar to the first clustering result in our experiment. Finally, we repair the missing activity in each incomplete trace based on SRBA and compared the 50 activities that were previously deleted to conclude that 44 activities were predicted to succeed, that is, the accuracy was 0.86. Similarly, the prediction rate obtained by PROELR method is 0.84. For these two methods, we conduct 10 trials respectively and obtain the prediction accuracy of each time and average value as shown in Figure 5 (b). Intuitively, we can see that the SRBA performs better than the PROELR method for the Lfull event log. When the missing activities is repaired, we add complete traces to the corresponding sub-logs. The process models are then obtained using process discovery techniques. Figure 6 shows two of the seven process models with seven sub-logs for the Lfull event log by α^+-algorithm [15].



(a) seven different groups of sub-logs (b) Lfull event log containing 50 missing activities

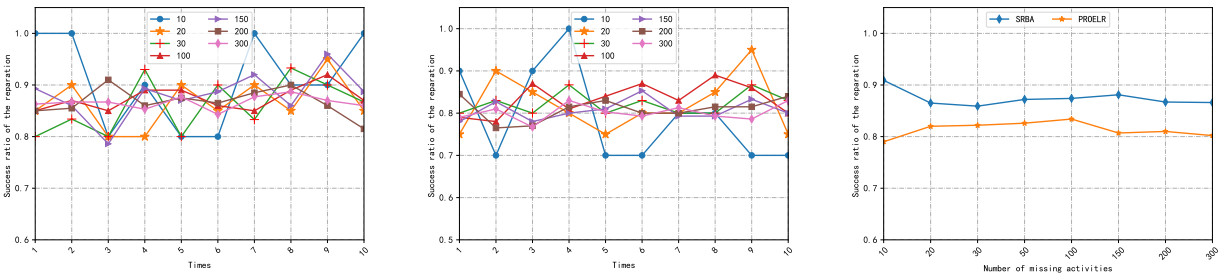
Fig. 5: The result of clustering and the comparison of the accuracy of the two methods.



(a) Process model 1 (b) Process model 2

Fig. 6: The process models for two different sub-logs.

Next, we generate seven incomplete logs of different sizes by removing 10, 20, 30, 100, 150, 200, and 300 activities respectively from the log Lfull. We just delete one of the activities in a trace selected randomly. For each incomplete log we perform 10 experiments and get the results as shown in the Figure 7(a)(resp.,Figure 7(b)) using the SRBA(resp.,PROELR) approach. Although the results of each experiment are different, they all fluctuate within a range. For example, most of the accuracy of repair exist between 0.85 and 0.9 shown Figure 7(a). By comparing the average accuracy of these two methods, it can be concluded that the SRBA method performs better than PROELR for the Lfull event log as a whole from Figure 7(c).

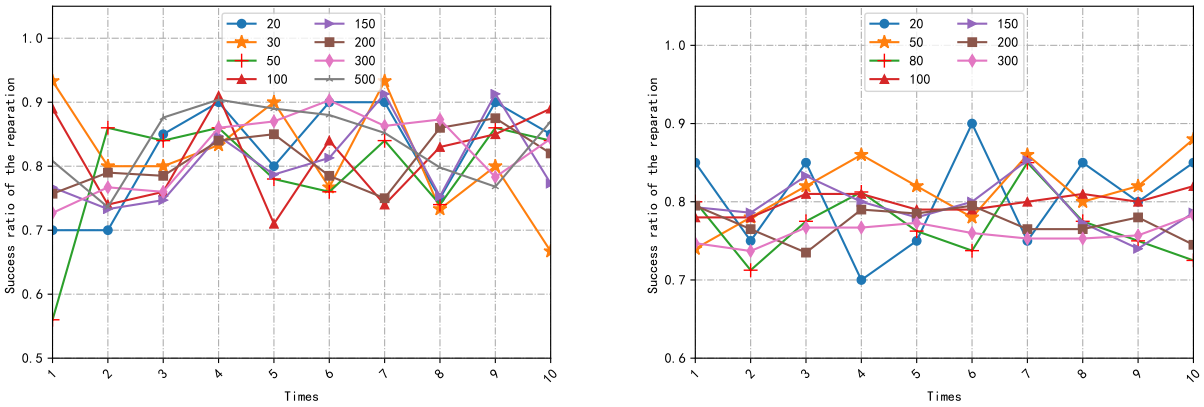


(a) (b) (c)

Fig. 7: Results for the event log of Lfull containing 10, 20, 30, 100, 150, 200, and 300 missing activities respectively.

In order to prove the ability of our approach to repair missing activities in incomplete traces, we altered two complex event logs, namely, Hospital Billing and BPI Challenge 2012. Before the experiment, we first pre-processed the Hospital Billing log and remove all traces including only two activities. The results are shown in Figure 8, where missing activities are randomly selected. Polylines of different shapes in Figure 8 indicate the number of remove activities from event log and the abscissa is the order of experiments. It can be seen from Figure 8 that the accuracy of repair each time remains around 0.8. The mean of accuracy iu shown in Table 4. The accuracy of our method is

about 0.17 higher than the PROELR method for the Hospital Billing log and about 0.34 for the BPI Challenge 2012 log.



(a) Hospital Billing log

(b)BPI Challenge 2012 log

Fig. 8: Accuracy of repair activities based on SRBA in different incomplete event logs.

We also selected the CCC19 log, as the input for consistency checking, to verify the accuracy of our method. The event log of the Conformance Checking Challenge will focus on a medical training process. To compare the performance of PROELR and SRBA algorithm, we generated several different incomplete logs with the same way in which conducted 10 experiments. The mean of accuracy as shown in Table 4. From the results, we can see that the proposed SRBA method boosted overall the accuracy of PROELR.

Table 4: The accuracy of different method for repairing missing activities on three real-life event logs.

Hospital Billing Log			BPI Challenge 2012 Log			Conformance Checking Challenge 2019 Log		
Method / Number	PROELR	SRBA	Method / Number	PROELR	SRBA	Method / Number	PROELR	SRBA
20	0.645	**0.825**	20	0.450	**0.805**	3	0.467	**0.835**
30	0.623	**0.817**	50	0.449	**0.816**	5	0.500	**0.860**
50	0.634	**0.794**	80	0.446	**0.770**	6	0.433	**0.833**
100	0.644	**0.816**	100	0.441	**0.800**	7	0.471	**0.785**
150	0.650	**0.805**	150	0.463	**0.790**	8	0.538	**0.763**
200	0.635	**0.811**	200	0.432	**0.772**	10	0.440	**0.780**
300	0.668	**0.825**	300	0.438	**0.760**	-	-	-
500	0.616	**0.838**	-	-	-	-	-	-

6 Conclusion and Future Work

In this paper, our goal is to provide more reliable and complete event logs for process mining techniques. We propose an approach to repair missing activities in incomplete event logs based on the succession relationships between activities in process mining. First, complete traces and incomplete traces are converted into matrix by extracting succession relations. Second, we use the SOM method to separate complete traces into different clustering results. And each incomplete trace is assigned to the most similar clustering result by computing the distance measure and the number of traces. Finally, missing activities in incomplete traces are repaired according to the relations of activities from complete traces in clustering results. Evaluation results on several real-life event logs demonstrate that our approach is better than the previous PROELR approach.

Our method currently focuses on a missing activity in each trace. One avenue of our future work is to predict more than one missing activity. Besides, we may also consider other attributes such as resource or timestamp to together determine missing activities based on the succession relations of activities.

References

1. Van der Aalst W, Weijters T, Maruster L (2004) Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9):1128–1142

2. van der Aalst WM (2018) Process discovery from event data: Relating models and logs through abstractions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(3):e1244
3. Aalst WMPVD (2011) Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Publishing Company, Incorporated
4. van der Aalst et al (2012) Process mining manifesto. In: Daniel F, Barkaoui K, Dustdar S (eds) Business Process Management Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 169–194
5. Augusto A, Conforti R, Dumas M, La Rosa M, Bruno G (2016) Automated discovery of structured process models: Discover structured vs. discover and structure. In: Comyn-Wattiau I, Tanaka K, Song IY, Yamamoto S, Saeki M (eds) Conceptual Modeling, Springer International Publishing, Cham, pp 313–329
6. Augusto A, Conforti R, Dumas M, La Rosa M, Polyvyanyy A (2019) Split miner: automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems 59(2):251–284, DOI 10.1007/s10115-018-1214-x, URL https://doi.org/10.1007/s10115-018-1214-x
7. Brown ML, Kros JF (2003) Data mining and the impact of missing data. Industrial Management & Data Systems 103(8):611–621
8. De Medeiros AA, Van Dongen BF, Van der Aalst WM, Weijters A (2004) Process mining: Extending the α-algorithm to mine short loops pp 145–180
9. De Weerdt J, Vanden Broucke S, Vanthienen J, Baesens B (2013) Active trace clustering for improved process discovery. IEEE Transactions on Knowledge and Data Engineering 25(12):2708–2720
10. Delias P, Doumpos M, Grigoroudis E, Matsatsinis N (2019) A non-compensatory approach for trace clustering. International Transactions in Operational Research 26(5):1828–1846
11. Di Francescomarino C, Dumas M, Federici M, Ghidini C, Maggi FM, Rizzi W, Simonetto L (2018) Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Information Systems 74:67–83
12. Effendi YA, Sarno R (2017) Discovering process model from event logs by considering overlapping rules. 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) pp 1–6
13. Fahland D, van der Aalst WMP (2012) Repairing process models to reflect reality. In: Barros A, Gal A, Kindler E (eds) Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 229–245
14. Greco G, Guzzo A, Pontieri L, Saccà D (2004) Mining expressive process models by clustering workflow traces. In: Dai H, Srikant R, Zhang C (eds) Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 52–62
15. Gu CQ, Chang HY, Yi Y (2008) Workflow mining: Extending the αalgorithm to mine duplicate tasks. In: 2008 International Conference on Machine Learning and Cybernetics, IEEE, vol 1, pp 361–368
16. Gunther CWC, Der Aalst WMPV (2007) Fuzzy mining: adaptive process simplification based on multi-perspective metrics pp 328–343
17. Ha QT, Bui HN, Nguyen TT (2016) A trace clustering solution based on using the distance graph model. In: Nguyen NT, Iliadis L, Manolopoulos Y, Trawiński B (eds) Computational Collective Intelligence, Springer International Publishing, Cham, pp 313–322
18. Jaeger D, Jung R (eds) (2015) Self-Organizing Maps, Springer New York, New York, NY, pp 2655–2655. DOI 10.1007/978-1-4614-6675-8_100525, URL https://doi.org/10.1007/978-1-4614-6675-8_100525
19. Lamma E, Mello P, Riguzzi F, Storari S (2008) Applying inductive logic programming to process mining. In: Blockeel H, Ramon J, Shavlik J, Tadepalli P (eds) Inductive Logic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 132–146
20. Lee C, Choy KL, Ho GT, Lam CH (2016) A slippery genetic algorithm-based process mining system for achieving better quality assurance in the garment industry. Expert Systems with Applications 46:236–248
21. Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs - a constructive approach. In: Colom JM, Desel J (eds) Application and Theory of Petri Nets and Concurrency, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 311–329
22. Leemans SJJ, Fahland D, van der Aalst WMP (2014) Discovering block-structured process models from incomplete event logs. In: Ciardo G, Kindler E (eds) Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, pp 91–110
23. de Leoni M, van der Aalst WMP (2013) Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: Daniel F, Wang J, Weber B (eds) Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 113–129
24. Lu X, Fahland D, van der Aalst WMP (2015) Conformance checking based on partially ordered event data. In: Fournier F, Mendling J (eds) Business Process Management Workshops, Springer International Publishing, Cham, pp 75–88
25. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WMP (2016) Balanced multi-perspective checking of process conformance. Computing 98(4):407–437, DOI 10.1007/s00607-015-0441-1, URL https://doi.org/10.1007/s00607-015-0441-1
26. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WM, Toussaint PJ (2018) Guided process discovery–a pattern-based approach. Information Systems 76:1–18
27. de Medeiros AKA, Weijters AJMM, van der Aalst WMP (2007) Genetic process mining: an experimental evaluation. Data Mining and Knowledge Discovery 14(2):245–304, DOI 10.1007/s10618-006-0061-7, URL https:

//doi.org/10.1007/s10618-006-0061-7
28. Rozinat A, Van der Aalst WM (2008) Conformance checking of processes based on monitoring real behavior. Information Systems 33(1):64–95
29. Song M, Günther CW, van der Aalst WMP (2009) Trace clustering in process mining. In: Ardagna D, Mecella M, Yang J (eds) Business Process Management Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 109–120
30. Sun Y, Bauer B, Weidlich M (2017) Compound trace clustering to generate accurate and simple sub-process models. In: Maximilien M, Vallecillo A, Wang J, Oriol M (eds) Service-Oriented Computing, Springer International Publishing, Cham, pp 175–190
31. Van Der Aalst WMP (2013) Business process management: a comprehensive survey. ISRN Software Engineering 2013:37, URL http://dx.doi.org/10.1155/2013/507984
32. Wang P, Tan W, Tang A, Hu K (2018) A novel trace clustering technique based on constrained trace alignment. In: Zu Q, Hu B (eds) Human Centered Computing, Springer International Publishing, Cham, pp 53–63
33. Weijters A, Ribeiro J (2011) Flexible heuristics miner (fhm). In: 2011 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, pp 310–317
34. Wen L, van der Aalst WMP, Wang J, Sun J (2007) Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery 15(2):145–180, DOI 10.1007/s10618-007-0065-y, URL https://doi.org/10.1007/s10618-007-0065-y
35. Xu J, Liu J (2019) A profile clustering based event logs repairing approach for process mining. IEEE Access 7:17872–17881
36. Zakarija I, Skopljanac-Macina F, Blaskovic B (2015) Discovering process model from incomplete log using process mining. 2015 57th International Symposium ELMAR (ELMAR) pp 117–120
37. ZarehFarkhady R, Aali SH, Branch B (2012) A two phase approach for process mining in incomplete and noisy logs
38. van Zelst SJ, van Dongen BF, van der Aalst WMP (2018) Event stream-based process discovery using abstract representations. Knowledge and Information Systems 54(2):407–435, DOI 10.1007/s10115-017-1060-2, URL https://doi.org/10.1007/s10115-017-1060-2

Jie Liu, received his master degree from China University of Petroleum (Eastern China) 2020. the B.S. degrees in Software Engineering from the University of Yan'tai, Yan'tai, in 2016. His research interest are Process Mining and Machine Learning. Now, he is a PhD candidate of Beijing University of Posts and Telecommunications.

Jiuyun Xu, Professor, received Bachelor Degree in Computational Mathematics from Shandong University in 1990, Master Degree in Computer Application from China University of Petroleum in 1997 and the Ph.D. degree in Computer Science from Beijing University of Posts and Telecommunications in 2004.

He is a Professor of China University of Petroleum. His research interests include service computing and Internet of Things. He has published in excess of 50 International conferences and Journals papers. He has as co-author awarded two best papers of International Conference on Service Science and National Conference of Service Computing of CCF. Prof. Xu is a reviewer for some prestigious journals including IEEE TSC, IJIMS, etc. He has also served on the technical program committees for numerous conferences including ICWS, SCC, etc. He is a member of IEEE and ACM, and a senior member of CCF.

Ruru Zhang, received her Master degree in Computer Technology from The China University of Petroleum in 2016. Currently, she is a Software Engineer of China Mobile R&D Center (Suzhou). Her research interests include service computing and cloud computing. She has published in several International Journals papers.

Stephan Reiff-Marganiec, Professor, PhD, the Head of School of Computing and Engineering at the University of Derby. Prior, he worked in the computer industry in Germany and Luxembourg for several years and as researcher at the University of Glasgow, while at the same time reading for a PhD in Computing Science investigating features and associated conflict resolution techniques. He was co-Chair of the 8th and 10th International Conference on Feature Interactions in Telecommunications and Software Systems, and programme chair of the International Conference on Web Services 2016. Stephan was leader of workpackages and tasks in the EU funded projects Leg2Net, Sensoria and inContext focusing on automatic service adaption, context aware service selection, workflows and rule based service composition. Stephan co-edited the Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions published in 2011. Stephan has published in excess of 100 papers in international conferences and journals and has been a member of a large number of programme committees.