IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 2, 2019, accepted January 15, 2019, date of publication January 24, 2019, date of current version February 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894905

A Profile Clustering Based Event Logs Repairing Approach for Process Mining

JIUYUN XU, (Member, IEEE), AND JIE LIU
College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China
Corresponding author: Jiuyun Xu (jyxu@upc.edu.cn)

ABSTRACT Process discovery, as the most crucial learning task in the process mining, builds some highly complex process models such as ''spaghetti-like'' from event logs contained large amounts of data. To enhance the process discovery method for all of flexible environments, many researchers tried to exploit trace clustering approaches to split the logs into several homogeneous sub-logs, which are used to generate the corresponding sub-process model, respectively. However, their works are based on the assumption that the event logs are complete without missing any data values. On the contrary, the data in an event log may be lost due to some reasons such as system failure and human error. In this paper, we propose a method to deal with incomplete logs so as to discover the process model. First, we split up the event logs based on trace clustering. Then, the missing traces are assigned to the most similar clustering results, respectively. After that, with supplementing the missing data in the trace, a corresponding sub-process model is mined using the proposed method. At last, some experimental results on three real-life complex event logs demonstrate the feasibility and effectiveness of our method.

INDEX TERMS Process mining, highly flexible environments, trace clustering, incomplete event logs.

I. INTRODUCTION

Process mining including business model discovery, conformance checking and enhancement of business processes has been proven that the execution of business processes can be well represented and analyzed by constructing process models based on event logs [1].The process mining algorithm was first proposed by Coke and Wolf, who put forward a data analysis technique called process discovery, as well as presented three major methods, namely the pure algorithm, the pure statistical and theory combined with algorithm [2]. Process discovery in the process mining domain is the paramount learning task that aims at automatically discovering a process model in order to explain the execution process in an event log [3], which has led some scholars to study how to get accurate process models by analyzing event logs [4]–[7]. In the process mining, all of activities recorded in an event log are employed to automatically discover associations within activities, to analyze the actual running process of the business. Furthermore, in event logs of information systems, there are not only activities but other attributes as well, such as, the event id, the timestamp, and resource, etc.

In recent years, most existing approaches of process model discovery can focus on how to discover accurate and comprehensible process model for well-structured processes [7]–[9]. However, many event logs from real-world business processes contain fairly complex workflow process in highly flexible environments such as healthcare, product development, and customer support, which lead to discovery complex process model, such as 'spaghetti-like', that is too complex to comprehend easily even for domain experts [10]–[12]. Greco *et al.* [13] proposed a novel approach based on the clustering of log traces to discover expressive models by analyzing complex logs from flexible environments in 2006. In the literature review, there are many approaches of clustering in data mining domain [14]. Similarly, some researchers, in process mining, try to improve trace clustering approaches in order to generate more accurate and expressive process models [15]–[21].

A. MOTIVATION

Currently, most process discovery techniques rely on a complete event log that contains all the data values. As a large amount of data is recorded in the event log, the likelihood of missing data becomes higher [22]. This suggests that it is unrealistic to assume that these complex event logs from highly flexible environment are complete in process mining. Due to various reasons such as

human negligence or database errors, some activities in the event logs may be lost, which will make the event log incomplete. Even more, using incomplete event logs to discovery process models makes the discovered model being deviated from the complete event logs. To illustrate this situation clearly, we will describe such a scenario as an example. Supposing there are twelve activity tasks in the given business process(called *OrderManagement*) [13]. They are, task(A) receiving the order, task(B) authenticating the client, task(C) checking the availability of the required product in the stock, task(D) verifying the availability of external supplies, task(E) registering the client in the company databases, task(F) evaluating the trustworthiness of the client, task(G) evaluating the production plan, task(H) rejecting the order, task(I) accepting the order, task(J) contacting the mail department in order to speed up shipment of the goods, task(K) applying some discount, and task(L) preparing the bill. Suppose there is such an event log containing an incomplete log that is transformed into seven groups of complete traces $L = [ABCGFIL, ABFCGIKL, ACBFGIJL, ACBGFIJKL, ABCFDGIKL, ACDBFGIKL, ABC\ DGFIKL]$ and a missing trace $[A, B, C, -, G, F, I, J, L]$. The process model shown in Figure 1 is obtained from this incomplete event log using heuristic mining algorithm. We add this completion missing trace to the incomplete log to make it a complete log. Figure 2 shows the process model based on the complete log $L_1 = [ABCGFIL, ABFCGIKL, ACBFGIJL, ACBGFIJKL, ABCFDGIKL, ACDBFGIKL, ABC\ DGFIKL, ABCDGFIJL]$. Comparing the two process models, it can be seen that the two red arcs in Figure 1 disappeared in Figure 2. A new arc of J to L was added in Figure 2. So this will have some negative impact on the analysis of the process model. For example, there is a J to L process in the complete event log, but the process model from incomplete log in Figure 1 cannot express this process execution.

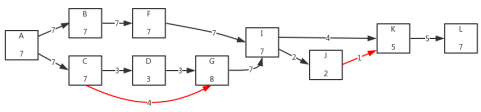

FIGURE 1. Process model based on the incomplete event log L.

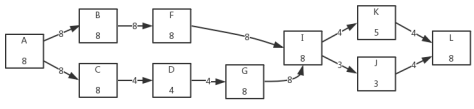

FIGURE 2. Process model based on the complete event log L_1.

In this paper, to solve this problem, we propose a novel approach to complementing the missing data in complex logs. First, we split all of event logs into two parts that one part includes the complete log, and the other is the incomplete log. Then these two parts are configured into their log profiles (which was first introduced by Song *et al.* [15]). After that, using a clustering method clusters the complete profiles into k clusters. Then, the distance of similarity between the traces of missing data and classify clusters are calculated so as to confirm which cluster the trace logs belong to. Finally, the traces with missing data are successfully categorized into different clusters and missing data will be predicted. So, we can discover simple and accurate process models from complete sub-logs, to a certain extent, which enhances the quality of process mining model in real flexible environment. It is worth noting that we are not using all the event log data in our method, rather than the sub-logs obtained by clustering to predict the missing activity.

In this paper, we focus on the event log with one missing activity in each trace and propose a general framework for dealing with missing traces to get complete log. To complement these missing traces, we proposed a framework which explores the context of full traces in the incomplete log with trace profile generation, clustering, matching, complementing and mining.

B. THE CONTRIBUTIONS OF THIS PAPER

The contributions of this paper are as follows:

- A general framework for repairing missing activities in the event log are proposed for process mining.
- We not only consider the distance measure but also the number of traces in the cluster result for calculating similarity between missing trace and clusters.
- With real world experiments, we conducted the performance analysis of our method.

To best of our knowledge,

The rest of this paper is structured as follows: First, related work of process mining is given in Section II. And some preliminary knowledge presented in Section III. Next, Section IV describes an overview of the proposed method and two of the most critical steps in detail. Section V gives a case to explain our approach. Then some experimental results on real event logs are discussed in Section VI. Finally, Section VII draws conclusion and future work.

II. RELATED WORK

Retrospect to the Business Process Management, process mining has been attracted to many researchers to carry out process model discovery, conformance checking and enhancement of business process. In 2004, van der Aalst *et al.* [3] 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 α-algorithm, however, still has some problems when generating process models. For example, some noises and loops path constructs appeared in the event log cannot be handled exactly. Hence, based on α -algorithm, [23] introduces an α^+-algorithm aiming to mine short loops processes and Wen *et al.* [24] presented an α^{++}-algorithm focused on mining non-free-choice constructs, which can be able to figure out explicit and implicit tasks from event logs. Gu *et al.* [25], similarly,

also extended the α algorithm called $\alpha^{\#}$ to deal with cyclic tasks in business process preferably. We can see that these algorithms extract certain traces from the event log and build the process model directly. Although other methods described in [8] and [26] have similar processes when collecting traces, they also considered the frequency of traces in the event log when dealing with noise and incompleteness-related problems. This makes the discovered process model available for important tasks but does not support overly complex process models and since the unimportant tasks are abandoned so that the restored model is inaccurate.

In addition, there are two other methods of discovery that are applied to process mining. One is called computational intelligence technique including genetic algorithms, neural networks, ant colony algorithms and so on. Instead of converting an event log into a process model directly, this algorithm simulated the natural evolution process by using iterative approach. For genetic process mining [27], [28], it can not only deal with the noise of log information but also mine repetitive tasks and hidden tasks in process mining. However, there is a disadvantage, in that the calculation is still performed when the noise constitutes a condition for stopping the program. The other is called local computing method. These techniques mentioned above focused on the discovery of end-to-end models representing the behavior of the event log traces. In fact, we can also concentrate on discovering rules and frequent patterns in event logs. Reference [29] described a new approach, *episodes miner*, that discovery frequently occurring episodes to mine local patterns in complex business processes. An episode is a collection of events among consecutive partial order. Although *episodes miner* can find frequent events, it needs heuristics techniques to accelerate the result of the mining process models. The combination of heuristic mining and region-based technology is often a better choice in the field of process mining, because only regional-based technologies often have problems when dealing with noise from the event logs. Tax applied three different heuristic methods for the event logs through log projections so as to yield process model fragments in [30]. For the anomalous frequent behaviors occurring in parallel behaviors, Genga *et al.* [31] introduced an approach to find the anomalous frequent local process models that can not only express parallel traces but also explain repeated deviations correlatively from partially ordered event logs.

These algorithms are able to generate expressible process models for well-structured business processes, but may find some bad process models that are quite complex and difficult to understand in the face of the complex event logs and unstructured business processes from flexible environments. As a result, researchers explored some new approaches based on trace clustering to copy with such problem [4], [13], [16]–[18], [20]. Greco *et al.* [13] first came up the idea of trace clustering to discovery sub-process models, where they represent trace logs using a vector and use the K-means algorithm to cluster event logs. Based on trace profiles, any clustering algorithm can divide the event log into several similar sub-logs so that improve the expression of the process model [4]. In addition, Song also described how to calculate the similarity between trace profiles and four clustering algorithm.

In the literature of process mining using clustering technique, De Weerdt *et al.* [17] indicated that some trace clustering technique can cause divergence between the clustering bias and the evaluation. Consequently, they employ a novel activity trace clustering approach, called ActiTraC, in order to improve the accuracy of process discovery. The ActiTrac algorithm including three phases: selection, look ahead and residual trace resolution is to cluster an event log into a set of clusters, each of which contains a collection of similar event log traces. The sub-process models, however, generated by existing traditional trace clustering approaches are inaccurate to a certain extent [16]. Therefore, the accuracy and complexity of yielding sub-process models by Compound Trace Clustering(CTC) technique presented in their papers are considered. In contrast to other trace clustering methods, CTC method can improve the accuracy of process models by reducing the complexity of sub-process models.

Although process mining techniques list above can discover process models from complete event logs, there is no guarantee that all event logs will be complete, especially in highly complex and flexible environments. In this paper, we are committed to researching on incomplete event logs so that the resulting process model can adequately reflect the business execution process.

III. PRELIMINARIES

In this section we describe some essential process mining notations including eXtensible Event Stream(XES) and log profiles, which are used in later sections.

Let Ψ be the set of all event identifiers for the event space. The event can be described by different attributes in an event log L. For example, an event may have some attributes, such as *concept:name*, *org:resource*, *lifecycle:transition*, etc. Let AN be the set of all attributes for the events. For an event $e \in \Psi$ and an attribute $n \in AN$, $\#_n(e)$ denote the value of the attribute n of the event e, e.g. $\#_{activity}(e)$ is the activity associated with event e. In this context, a sequence of events e are represented by a trace σ, e.g. the trace $\sigma = \langle e_1, e_2, e_3 \rangle$ consists of three different events. For example, $L = [\langle a, b, c \rangle, \langle a, c, b \rangle]$ shows that there are 2 traces in the log. With the assumption of $|L|$ as the number of occurrences of traces in a log L, so $|L| = 2$.

For the context of attributes n of the event e, we use $\#_{\bullet n}(e)$ and $\#_{n\bullet}(e)$ to denote the set of pre-occurrence and post-occurrence of $\#_n(e)$ directly, which consist of the number of occurrences is represented by $|\#_{\bullet n}(e)|$ and $|\#_{n\bullet}(e)|$ respectively. For example, we assume the $\#_n(e_1) = C$ and $\#_n(e_2) = G$. Observing the sequence of trace($T1 - T16$) in Table 4, we can get some results as shown in Table 1. The $C \bullet \cap \bullet G$ represents the intersection of $C\bullet$ and $\bullet G$ while the corresponding number of occurrences is the mean of $|C \bullet| + |\bullet G|$ for $|C \bullet \cap \bullet G|$.

TABLE 1. The results when $\#_n(e_1) = C$ and $\#_n(e_2) = G$.

$\bullet C = \{A, F, B\}$	$C\bullet = \{B, G, E, D, F\}$
$\lvert \bullet C\rvert = \{8, 3, 5\}$	$\lvert C \bullet \rvert = \{4, 6, 2, 3, 1\}$
$\bullet G = \{B, C, F, E, D\}$	$G\bullet = \{F, H, I, B\}$
$\lvert \bullet G\rvert = \{3, 5, 4, 1, 3\}$	$\lvert G \bullet \rvert = \{5, 2, 8, 1\}$
$C \bullet \cap \bullet G = \{B, D, E, F\}$	$\lvert C \bullet \cap \bullet G\rvert = \{3.5, 3, 1.5, 2.5\}$

In the current information industry setting, XES files are widely used in the filed of process mining as a semantic annotation version for structuring event logs because of their better scalability than Mining eXtensible Markup Language(MXML) and an event log consist of any number of traces that describe a list of sequence events. Some attributes such as *activity*, *resource* and *timestamp* may be recorded in an event log. For convenience, we map some fragment data in XES to Table 2 showing 19 events and 9 activities.

Definition 1 (Missing Trace, Complete Trace): Let $\sigma = \langle e_1, e_2, e_3, \ldots, e_n\rangle$ be a trace from an event log. For $\forall i \in \{1, 2, \ldots, n\}$, missing trace (MT) is defined, if and only if, $\exists e_i \in \sigma$ missing any attribute data. Otherwise, this trace is defined as a complete trace.

From the table 2, we can see that one of the activities of trace id 3 is missing. As a consequence, trace 3 is called a missing trace, that is, $MT = <A, B, D, -, H>$.

Definition 2 (Incomplete Log, Complete Log): Let $L = [\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_m]$. For $\forall i \in \{1, 2, \ldots, m\}$, incomplete log is defined if and only if $\exists \sigma_i \in L$ is a missing trace. Otherwise, this log is defined as a complete log.

As for the event log $L = [<A, B, D, E, F, G, I>, <A, B, C, D, E, G, I>, <A, B, D, -, H>]$ represented by table 2, this log is called incomplete log because there is a missing trace $<A, B, D, -, H>$.

TABLE 2. Example an incomplete event log.

trace id	*activity*	*resource*	*trace id*	*activity*	*resource*
1	A	John	2	D	Pete
1	B	Mike	2	E	Jane
1	D	Pete	2	G	Jane
1	E	Jane	2	I	Mona
1	F	Fred	3	A	John
1	G	Clare	3	B	Mike
1	I	Mona	3	D	Pete
2	A	John	3	-	Jane
2	B	John	3	H	John
2	C	Sue	...	...	...

In this paper, we use *profiles* to cluster traces. The *profiles* can describe traces of the different attributes in event logs, i.e. *activity profile* contains all the activity behavior trace vector. In this paper we focus on events' activities.

Definition 3 (Transformation of a Trace Into a Trace Vector): A trace vector($tv = <i_1, i_2, i_3, \ldots, i_n>$), where the position of each index corresponds to a fixed activity, denotes the number of occurrences of activities in a trace.

For instance, in the second row of the table 3, the trace vector $tv_1 = \langle 1, 1, 0, 1, 1, 1, 1, 0, 1\rangle$ which means that activities {A,B,D,E,F,G,I} occur once along with no occurrence of activities {C, H} in the trace.

TABLE 3. Activity profile from Table 2.

Trace	Activity Profile								
	A	B	C	D	E	F	G	H	I
T1	1	1	0	1	1	1	1	0	1
T2	1	1	1	1	1	0	1	0	1
MT1	1	1	0	1	0	0	0	1	0

Definition 4 (Complete Trace Vector, Missing Trace Vector): A complete(resp.missing) trace vector is defined if and only if it is transformed by a complete(resp. missing) trace.

Definition 5 (Full Trace Profiles(FTF), Missing Trace Profiles(MTF)): Full(resp.Missing) trace profiles is defined if and only if it consists of complete(resp. missing) trace vectors.

Table 3 shows the result of two types of trace profiles according to table 2. The second and third rows form full trace profiles and the fourth row form missing trace profiles.

It is well known that Self-Organizing Map(SOM) is one of Artificial Neural Network algorithms and can conduct unsupervised learning classification for high dimensional data to produce a low-dimensional representation [32]. So SOM is called a natural method of dimension reduction and with this property we can cluster profiles. SOM simulates the processing of information in the human brain neurons and automatically clusters the input patterns through self-training. It is composed of input layer and hidden layer based on competitive learning to apply neighborhood function so that can determine the topological structure of the input space. The purpose of SOM is to aggregate similar traces together and separate dissimilar traces.

Since the profile is composed of n-dimensional vectors, we can use distance measures to calculate the similarity between missing trace data and clusters. By using the SOM algorithm, we can get several sets of different clustering results that are represented by C_i contained similar traces. We also define the average trace vector for each cluster C_i as the follow formula. Notice that $|C_i|$ denotes the number of traces in the cluster.

$$\bar{tv_i} = \frac{\sum_{i=1}^{|C_i|} tv_i}{|C_i|} \tag{1}$$

In the paper, we choose Euclidean distance to calculate similarity between missing trace data and clusters, which is defined as follow:

$$ED(MT, \bar{tv_i}) = \sqrt{\sum_{j=1}^{n} |MT_{ij} - \bar{tv}_{ij}|^2} \tag{2}$$

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 *EDs* are close, the final similarity of a missing trace and cluster result C_i is related to the number of elements in the cluster, namely, the more the value is, the more similar it is. We use the following formula to determine similarity:

$$FS = \alpha * ED(MT, \bar{tv_i}) + \beta * |C_i| \tag{3}$$

where the value of α plus β is 1. We consider that when the similarity values of the two clustering results are very close, the number of traces in the cluster will play a decisive role. Therefore α (resp. β) is set to 0.4 (resp.0.6) in our experiment. This formula is guided by a distance threshold(dt). If the difference between any two *EDs* is less than set threshold manually, we will calculate *FS* to determine which cluster the current *MT* is assigned to. Otherwise, we only use the Euclidean distance to decide the similarity.

IV. AN EVENT LOG REPAIRING APPROACH BASED ON PROFILE CLUSTERING(PROELR)

Real-life event logs, extracted from business process in highly flexible environments, are not only considerably complex but its data volume is also huge. Therefore, these event logs are difficult to guarantee data integrity and may be missing some data values so that yielding sub-process models are inaccurate and incomprehensive by using trace clustering technique. In this paper, we describe an approach to copy with event logs with missing data in order to make sub-process models well represent the execution process of the business. In this section, we first present a general framework for completing missing event log. Second we describe the detail of two of the most critical steps(matching and complement) in the framework.

A. OVERVIEW OF THE PROPOSED FRAMEWORK

The framework is shown in figure 3 and contains the following five main steps:

1) **Generation:** Generate the initial two types of trace profiles, namely, the full trace profiles *FTF* and missing trace profiles *MTF*. By scanning the XES event log, we separate the complete traces and incomplete traces, and then form *FTF* and *MTF* separately.

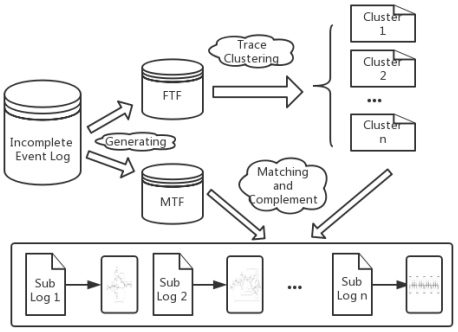

FIGURE 3. Framework for completing missing event log.

2) **Clustering:** Use the *FTF* for clustering. In this paper, Clustering involves the combination of the Euclidean distance and the SOM algorithm, which partitions *FTF* into a group of clusters C_i to ensure that the trace vectors within the cluster have high similarity and conversely any cluster is not sufficiently similar to other clusters.
3) **Matching:** Match missing traces to clusters. In the calculation of similarity, we simultaneously consider the distance and the number of elements in the cluster to jointly determine which cluster the missing trace data belongs to and add it to the cluster.
4) **Complementing:** Complete missing activity in the trace. The missing activity is complemented according to the event log corresponding to the profiles in the clusters and context environment. In this way, traces in each cluster are complete, that is, each sub-log is complete.
5) **Mining:** Mine sub-process models. These sub-process models that are mined using existing mining algorithm are more accurate than those obtained without the completion of the process.

B. CANDIDATE CLUSTERS MATCHING

Algorithm 1 describes the process of finding candidate clusterings of missing traces, that is, the third step of the framework. There is one point to note when calculating similarity of missing traces and clusters. First, we need to calculate the average trace vector using Formula 1 for each clustering result(line 1-2).

Algorithm 1 Match Candidate Clustering of MT_j

Input: n clustering results, m missing traces, dt
Output: candidate clustering results of missing traces

```
1  for i ← 1 to n do
2      t̄v_i = (Σ_{i=1}^{|C_i|} tv_i) / |C_i| ;
3  for j ← 1 to m do
4      for i ← 1 to n do
5          ED_i ← calculate Euclidean distance between
           MT_j and t̄v_i;
6      maxED ← max(ED_i) ;
7      for i ← 1 to n do
8          df ← calculate difference between maxED and
           ED_i;
9          if df <= dt then
10             candidate clustering C_i of MT_j
```

We calculate the Euclidean distance between missing trace and each cluster(line 3-5). Note that the previous methods usually considered only distance parameter when computing similarity between any two trace vectors. But, in this paper, we also considered the number of elements in the cluster as a measure of the standard. We think that it is possible

that this missing activity trace is assigned to one of the clusters when the distance are not much different. Based on this point, we therefore take the number of elements in the cluster as another important calculation parameter. In other word, the distance metric and the number of activity traces in the cluster determine which cluster the missing activity belongs to.

C. MISSING ACTIVITIES COMPLEMENTING

The approach to computing the missing activity is formalized in Algorithm 2. we choose whether to use Formula 2 or Formula 3 to determine the final similarity result based on the distance threshold. This missing activity trace is eventually assigned to the most similar cluster(line 2). We next find the input and output activity of the missing activity namely $\#_{activity}(e_1)$ and $\#_{activity}(e_2)$ (line 3). Then the cluster corresponding to the missing trace is scanned and $\#_{activity\bullet}(e_1)$ and $|\#_{activity\bullet}(e_1)|$ are counted respectively. The analogue process is used to compute $\#_{\bullet activity}(e_2)$ and $|\#_{\bullet activity}(e_2)|$ (line 4). Next, there are three situations that will be considered for $\#_{activity\bullet}(e_1)$ and $\#_{\bullet activity}(e_2)$:

1) If the intersection of this two sets is not empty, then the activity with the largest number of intersections is selected as the missing activity (lines 6-8).
2) If the intersection of this two sets is empty and one of them is not empty, then the activity with the largest number of this two sets is selected as the missing activity (lines 9-11).
3) If both sets are empty, then we will discard this missing trace. In other words, this missing trace will not appear in any clustering results (lines 12-13).

Algorithm 2 Complete Missing Activities of Traces

Input: candidate clustering results of missing traces
Output: complete sub-logs

```
1  for j ← 1 to m do
2      C ← select the most similar clustering result
       of MT_j;
3      X, Y ← find input and output of the missing activity
       in MT_j;
4      compute X•, •Y, |X • |, | • Y| in C;
5      XY, |XY| ← X • ∩ • Y, |X • | ∩ | • Y|;
6      if XY is not NULL then
7          select the activity corresponding to the
           maximum number in |XY| as the missing
           activity;
8          add MT_j to C;
9      else if XY is NULL and (X• is not NULL or •Y is
       not NULL) then
10         select the activity corresponding to the
           maximum number in |X • | and | • Y| as the
           missing activity;
11         add MT_j to C;
12     else
13         discard MT_j;
```

V. CASE STUDY

Let's given a case study of the proposed method with the *OrderManagement* example in Section I. We select 16 intact traces and two missing traces as shown in Table 4. These two missing traces are artificially formed in the experiment. In fact, with the complete traces corresponding to two missing traces of $<A, B, C, D, G, F, I, J, L>$ and $<A, C, B, F, I, J, K, L>$ respectively, we just deleted one of the activity task of them. In this section, we describe in detail how to get the missing activity in $MT1$. These traces first are combined into a complete activity profile and a missing activity profile respectively. The result of the activity profile see Table 5. Next, Figure 4 shows the clustering result where these 16 intact traces vectors are divided into four clusters by using SOM algorithm. The result of 4 clusters respectively are $C_1 = \{T4, T5, T6, T7, T13, T14, T15\}$, $C_2 = \{T8, T10\}$, $C_3 = \{T1, T2, T3\}$, and $C_4 = \{T9, T11, T12, T16\}$.

TABLE 4. An instance of some activity traces.

Trace	Sequence of activities
$T1$	$< A, C, B, G, F, H >$
$T2$	$< A, B, F, C, G, H >$
$T3$	$< A, C, G, B, F, H >$
$T4$	$< A, B, C, G, F, I, L >$
$T5$	$< A, B, F, C, G, I, K, L >$
$T6$	$< A, C, B, F, G, I, J, L >$
$T7$	$< A, C, B, G, F, I, J, K, L >$
$T8$	$< A, B, C, E, G, F, I, J, L >$
$T9$	$< A, B, E, F, C, G, I, L >$
$T10$	$< A, C, G, B, E, F, I, J, L >$
$T11$	$< A, B, C, E, D, F, G, I, L >$
$T12$	$< A, C, D, B, E, F, G, I, L >$
$T13$	$< A, B, C, F, D, G, I, K, L >$
$T14$	$< A, C, D, B, F, G, I, K, L >$
$T15$	$< A, B, C, D, G, F, I, K, L >$
$T16$	$< A, C, B, F, D, G, I, L >$
$MT1$	$< A, B, C, -, G, F, I, J, L >$
$MT2$	$< A, -, B, F, I, J, K, L >$

We can find candidate clusterings of missing traces by Algorithm 1. First, we need to calculate the average trace vector using Formula 1 for each clustering result. For example, the $\bar{tv} = \langle 1, 1, 1, 0.429, 0, 1, 1, 0, 1, 0.286, 0.714, 1\rangle$ for C_1. We then calculate the Euclidean distance between $MT1$ and each cluster, namely, $ED_1 = 0.477$, $ED_2 = 0.500$, $ED_3 = 0.333$, and $ED_4 = 0.407$. As a result, C_1 and C_2 are selected as the candidate clustering of $MT1$ because the difference value of two distances is less than the minimum threshold(0.08).

We use the formula 3 to get the final result $FS_1 = 4.931 > FS_2 = 3.2$ for the candidate clustering of $MT1$ in

TABLE 5. Activity profiles from Table 4.

Trace	Activity Profile											
	A	B	C	D	E	F	G	H	I	J	K	L
$T1$	1	1	1	0	0	1	1	1	0	0	0	0
$T2$	1	1	1	0	0	1	1	1	0	0	0	0
$T3$	1	1	1	0	0	1	1	1	0	0	0	0
$T4$	1	1	1	0	0	1	1	0	1	0	0	1
$T5$	1	1	1	0	0	1	1	0	1	0	1	1
$T6$	1	1	1	0	0	1	1	0	1	1	0	1
$T7$	1	1	1	0	0	1	1	0	1	1	1	1
$T8$	1	1	1	0	1	1	1	0	1	1	0	1
$T9$	1	1	1	0	1	1	1	0	1	0	0	1
$T10$	1	1	1	0	1	1	1	0	1	1	0	1
$T11$	1	1	1	1	1	1	1	0	1	0	0	1
$T12$	1	1	1	1	1	1	1	0	1	0	0	1
$T13$	1	1	1	1	0	1	1	0	1	0	1	1
$T14$	1	1	1	1	0	1	1	0	1	0	1	1
$T15$	1	1	1	1	0	1	1	0	1	0	1	1
$T16$	1	1	1	1	0	1	1	0	1	0	0	1
$MT1$	1	1	1	0	0	1	1	0	1	1	0	1

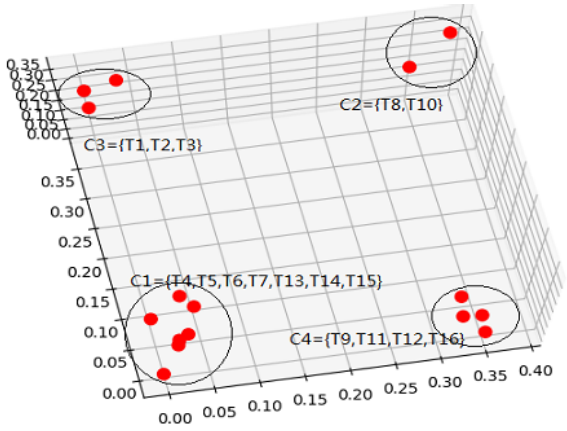

FIGURE 4. Result of trace clustering.

Algorithm 2. This missing activity trace $MT1$ is eventually assigned to the first cluster including seven traces(C_1). we can compute these results including $C\bullet = \{G, B, F, D\}$, $|C\bullet| = \{2, 2, 1, 2\}$, $\bullet G = \{C, F, B, D\}$, $|\bullet G| = \{2, 2, 1, 2\}$, $C\bullet \cap \bullet G = \{B, F, D\}$ and $|C\bullet \cap \bullet G| = \{1.5, 1.5, 2\}$ since the missing activity is the output of activity C and the input of activity G. We finally determine that the missing activity in the $MT1$ is D through the context of these seven traces. This complementary trace exactly matches the trace where we deleted one of the activities previously. It is worth noting that if we do not consider the number of elements of the cluster but only the distance when we calculate the similarity, the missing activity becomes E. Obviously, this result is not what we want, which further explains that our method is feasible and accurate for predicting missing activities.

VI. EXPERIMENTS AND RESULTS

In order to prove the feasibility of our approach, we have evaluated several real-life event cases in our experiment. Since the ProM[1] is a professional tool including a large number of process discovery algorithms to discovery process models in the field of process mining, we use heuristic mining algorithm [8] in the ProM to obtain process models. In this section, we focus on explaining the Hospital Billing cases collected by Eindhoven University of Technology in 2017[2] and the experimental results of the other two data sets are also briefly introduced.

The real Hospital Billing event log is provided by the financial modules of the ERP system of a regional hospital. The event log records medical services billing data from 2012 to 2016 in the hospital and has 100000 traces and 451359 events. In addition, the log also contain three global attributes, namely, *concept* : *name*, *lifecycle* : *transition*, and *time* : *timestamp*. All of the events identifications and the attribute values have been anonymized for privacy reasons. Note that, we only consider the *activity profile* that is related to the *concept* : *name* containing 18 activities. In our experiment, we need to preprocess the Hospital Billing event log in order to turn it into an incomplete log with missing activities. In this data set, we conducted four groups of experiments in which the missing traces were 100, 500, 1000, 10000 and 15000 respectively. To make the experiment more credible, these missing traces are randomly selected from this logs. In this way, we get four different incomplete event logs.

In each group of incomplete logs, we performed 10 experiments. By analyzing the experimental results, we concluded that the predicted success rate of the missing data in the four groups was 79.20%, 78.04%, 77.76%, 77.27% and 76.68% respectively. We add these completed traces to the corresponding sub-logs, while the unpredictable traces are omitted. Figure 5 shows the process model generated by the heuristic mining algorithm for the Hospital Billing event log without missing data. Besides, we also present several sub-process models for sub-logs completed by our approach. These three sub-models with 9, 5 and 17 activities are respectively described in Figure 6. We add completed traces to the corresponding sub-logs, which makes sub-process models more accurate and comprehensive. For example, the two models with 9 and 5 activities in Figure 6(a) and Figure 6(b) have less activity than the model in Figure 5. Although the model in Figure 6(c) and the model in Figure 5 have almost the same number of activities, they are considerably different in simplicity. Table 6 has some basic information about the Hospital Billing case, for example, the missing traces percentage indicates the proportion of the number of missing traces to total traces and the success percentage means that the number of traces complemented by our method accounts for the percentage of missing traces.

[1] http://www.promtools.org/doku.php?id=prom68

[2] https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

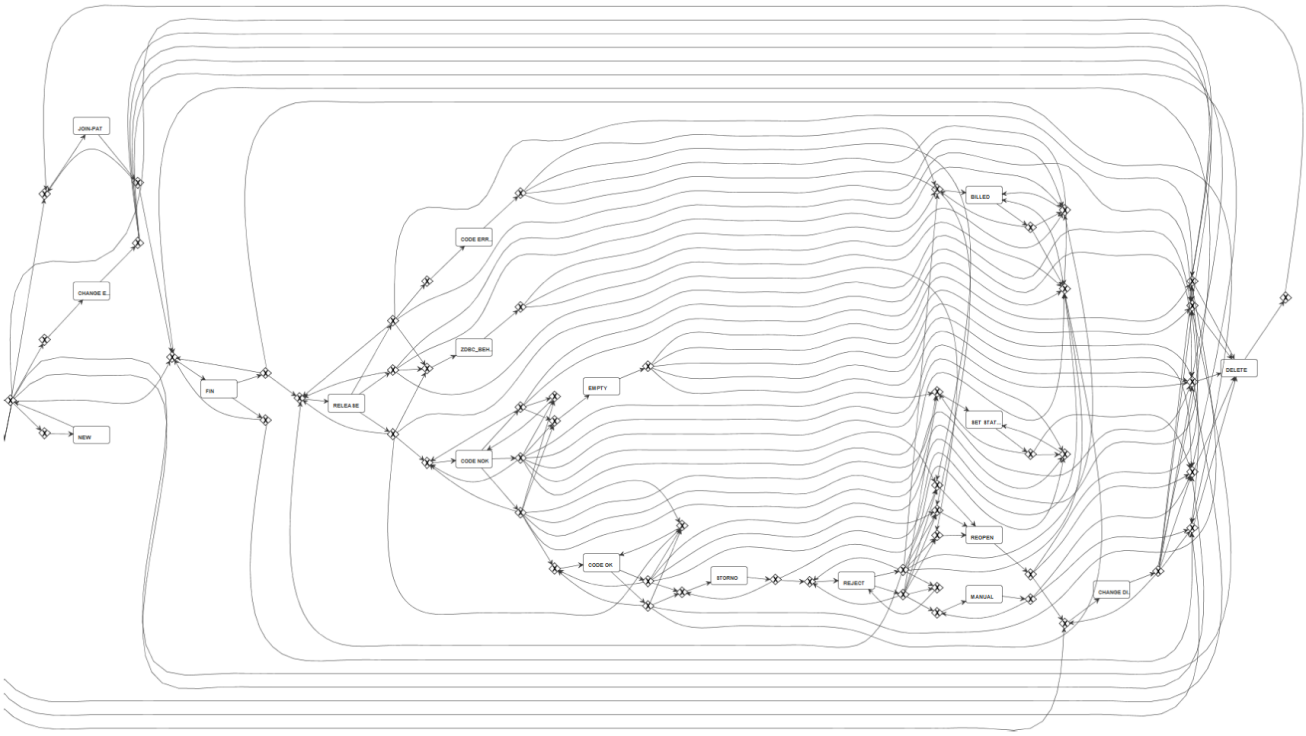

FIGURE 5. Process model based on the Hospital Billing log.

TABLE 6. The table shows some basic information about Hospital Billing event log case and the results of each experiment were found in ten runs.

missing traces percentage	number of traces	number of events	number of activities	number of missing traces	average number of missing traces of completion	success percentage
0.1%	100000	451359	18	100	79.20	79.20%
0.5%				500	390.20	78.04%
1%				1000	777.60	77.76%
10%				10000	7727	77.27%
15%				15000	11502	76.68%

TABLE 7. The table shows some basic information about real-life log of a Dutch academic hospital and the results of each experiment were found in ten runs.

missing traces percentage	number of traces	number of events	number of activities	number of missing traces	average number of missing traces of completion	success percentage
8.75%	1143	150291	624	100	95.10	95.10%
17.50%				200	190	95.00%
26.25%				300	280.41	93.47%
43.74%				500	455.80	91.16%

Correspondingly, four groups of experimental results are also shown in Figure 7.

Next, let us have a look at the experimental results for the other two data sets (the real-life log of a Dutch academic hospital(BPIC 2011)[3] and BPIC 2018).[4] As shown in Table 7 and Table 8, we respectively did four experiments for BPIC 2011 with 8.75%, 17.50%, 26.25% and 43.74% of missing traces and BPIC 2018 with 0.46%, 2.28%,11.41% and 18.26% of missing traces. By analyzing the results in different missing traces percentage we have the conclusion that our approach proposed in this paper can complement the missing data from event logs. Figure 8 (a)and Figure 8 (b) show results for the BPIC 2011 and BPIC 2018. To save space, we omitted the display of the sub-process models for the two event logs.

[3]https://data.4tu.nl/repository/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

[4]https://data.4tu.nl/repository/uuid:3301445f-95e8-4ff0-98a4-901f1f204972

TABLE 8. The table shows some basic information about BPI Challenge 2018 and the results of each experiment were found in ten runs.

missing traces percentage	number of traces	number of events	number of activities	number of missing traces	average number of missing traces of completion	success percentage
0.46%	43809	2514261	41	200	200	100%
2.28%				1000	990	99%
11.41%				5000	4950	99%
18.26%				8000	7920	99%

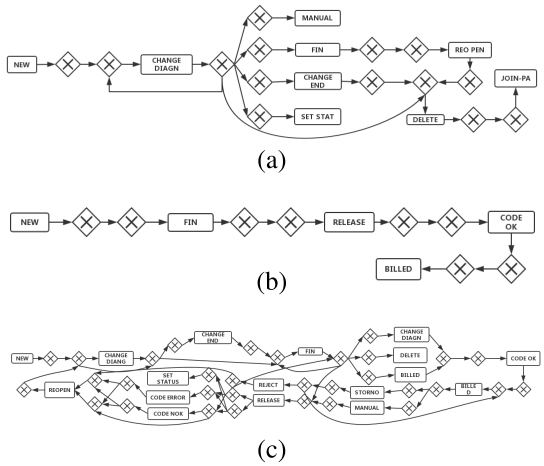

FIGURE 6. The process models for three complete sub-logs.

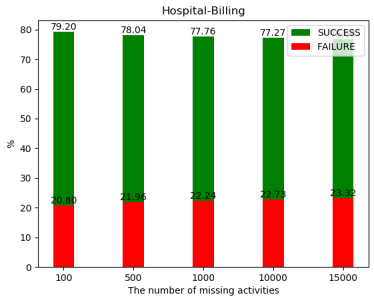

FIGURE 7. Results for Hospital Billing log with 0.1%, 0.5%, 1%, 10% and 15% of missing traces.

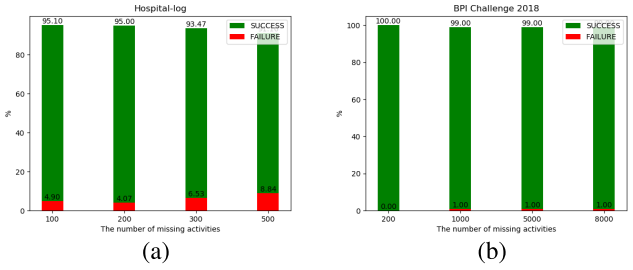

FIGURE 8. Results for real-life log of Hospital and BPIC 2018.

VII. CONCLUSION AND FUTURE WORK

Currently, existing process mining technologies are well represented in the face of some structured business processes. However, when unstructured business processes are produced in some flexible environments, trace clustering approaches can be used to solve this problem effectively. In practice, these logs are not guaranteed to be complete and they may lose some data for a variety of reasons. To best of our knowledge, no work has been done on how to discover process models on missing data logs.

In this paper we proposed a method to effectively deal with missing data logs from complex environments for obtaining business process models. Since these logs are complex considerably and unstructured, we first used the trace clustering method to partition them into several similar sets of sub-logs. Then we can predict the missing data by calculating the similarity between each missing trace and the sub-logs and taking into account the number of traces. As a result, more comprehensive and accurate sub-process models can be derived. Through the evaluation results from different real-life cases we demonstrated the validity of our method proposed in this paper.

In our experiment, we directly omitted some traces with missing data value that could not be predicted, which may have a slight impact on the results. In the future, we will pay attention to a future research direction to determine these missing data by other algorithms. Furthermore, the similarity threshold parameter is set by the value of the similarity matrix (the distance between the missing data traces and sub-logs). Since the similarity threshold parameter is artificially modified according to the change of similarity matrix, we will also consider how to set the similarity threshold parameter automatically in the future.

REFERENCES

[1] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Berlin, Germany: Springer, 2011.

[2] J. E. Cook and A. L. Wolf, "Discovering models of software processes from event-based data," *ACM Trans. Softw. Eng. Methodol.*, vol. 7, no. 3, pp. 215–249, 1998.

[3] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[4] W. M. P. van der Aalst *et al.*, "Business process mining: An industrial application," *Inf. Syst.*, vol. 32, no. 5, pp. 713–732, 2007.

[5] D. Fahland and W. M. P. van der Aalst, "Repairing process models to reflect reality," in *Proc. Int. Conf. Bus. Process Manage.*, 2012, pp. 229–245.

[6] 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 *Proc. Int. Conf. Appl. Theory Petri Nets Concurrency*, 2013, pp. 311–329.

[7] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa, "Beyond tasks and gateways: Discovering BPMN models with subprocesses, boundary events and activity markers," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 101–117.
[8] A. J. M. M. Weijters and J. T. S. Ribeiro, "Flexible heuristics miner (FHM)," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Apr. 2011, pp. 310–317.
[9] Y. Sun and B. Bauer, "A novel heuristic method for improving the fitness of mined business process models," in *Proc. Int. Conf. Service-Oriented Comput.*, 2016, pp. 537–546.
[10] R. S. Mans, M. H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker, "Application of process mining in healthcare—A case study in a dutch hospital," in *Biomedical Engineering Systems and Technologies*. Berlin, Germany: Springer, 2008, pp. 425–438.
[11] P. De Koninck, J. De Weerdt, and S. K. L. M. vanden Broucke, "Explaining clusterings of process instances," *Data Mining Knowl. Discovery*, vol. 31, no. 3, pp. 774–808, 2017.
[12] M. L. Sebu and H. Ciocârlie, "Applied process mining in software development," in *Proc. IEEE 9th Int. Symp. Appl. Comput. Intell. Inform.*, May 2014, pp. 55–60.
[13] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, "Discovering expressive process models by clustering log traces," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1010–1027, Aug. 2006.
[14] J. Han, *Data Mining: Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2005.
[15] M. Song, C. W. Günther, W. M. P. van der Aalst, *Trace Clustering in Process Mining*. Berlin, Germany: Springer, 2008.
[16] Y. Sun, B. Bauer, and M. Weidlich, "Compound trace clustering to generate accurate and simple sub-process models," in *Proc. Int. Conf. Service-Oriented Comput.*, 2017, pp. 175–190.
[17] J. De Weerdt, S. K. L. M. vanden Broucke, J. Vanthienen, and B. Baesens, "Active trace clustering for improved process discovery," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2708–2720, Dec. 2013.
[18] Q.-T. Ha, H.-N. Bui, and T.-T. Nguyen, *A Trace Clustering Solution Based on Using the Distance Graph Model*. Springer, 2016.
[19] R. P. J. C. Bose W. M. P. van der Aalst, "Context aware trace clustering: Towards improving process mining results," in *Proc. SDM SIAM*, 2009, pp. 401–412.
[20] P. Delias, M. Doumpos, E. Grigoroudis, and N. Matsatsinis, "A non-compensatory approach for trace clustering," *Int. Trans. Oper. Res.*, to be published. doi: 10.1111/itor.12395.
[21] P. Wang, A. Tang, and K. Hu, "A novel trace clustering technique based on constrained trace alignment," in *Proc. Int. Conf. Hum. Centered Comput.* Cham, Switzerland: Springer, 2017, pp. 53–63.
[22] M. L. Brown and J. F. Kros, "Data mining and the impact of missing data," *Ind. Manage. Data Syst.*, vol. 103, no. 8, pp. 611–621, 2003.
[23] A. K. A de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process mining: Extending the α-algorithm to mine short loops," Univ. Technol., Eindhoven, Eindhoven, The Netherlands, BETA Working Paper Series, WP 113, Tech. Rep., 2004.
[24] L. Wen, W. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 145–180, 2007.
[25] C.-Q. Gu, H.-Y. Chang, and Y. Yi, "Workflow mining: Extending the aalgorithm to mine duplicate tasks," in *Proc. IEEE Int. Conf. Mach. Learn. Cybern.*, Jul. 2008, pp. 361–368.
[26] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining—Adaptive process simplification based on multi-perspective metrics," in *Proc. Int. Conf. Bus. Process Manage.*, 2007, pp. 328–343.
[27] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic process mining: An experimental evaluation," *Data Mining Knowl. Discovery*, vol. 14, no. 2, pp. 245–304, 2007.
[28] C. K. H. Lee, K. L. Choy, G. T. S. Ho, and C. H. Y. Lam, "A slippery genetic algorithm-based process mining system for achieving better quality assurance in the garment industry," *Expert Syst. Appl.*, vol. 46, pp. 236–248, Mar. 2016.
[29] M. Leemans and W. M. P. van der Aalst, *Discovery of Frequent Episodes in Event Logs*. Springer, 2014.
[30] N. Tax, N. Sidorova, W. M. P. van der Aalst, and R. Haakma, "Heuristic approaches for generating local process models through log projections," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Dec. 2017, pp. 1–8.
[31] L. Genga, M. Alizadeh, D. Potena, C. Diamantini, and N. Zannone, "Discovering anomalous frequent patterns from partially ordered event logs," *J. Intell. Inf. Syst.*, vol. 51, no. 2, pp. 257–300, 2018.
[32] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer, 1997, pp. 85–144.

JIUYUN XU (M'90) received the Ph.D. degree in computer science from the China University of Posts and Telecommunications, in 2004. He is currently a Professor with the China University of Petroleum (Eastern China). He has published in excess of 40 international conferences and journals papers. His research interests include service computing and the Internet of Things. He is a member of the IEEE and ACM, and a Senior Member of CCF. He is a Reviewer for some prestigious journals, including the IEEE TSC, and IJIMS. He has also served on the technical program committees for numerous conferences including ICWS and SCC.

JIE LIU received the B.S. degree in software engineering from the University of Yantai, Yantai, in 2016. He is currently pursuing the master's degree with the China University of Petroleum (Eastern China). His research interests include process mining and next-generation network technology.