Log Clustering-based Method for Repairing Missing Traces with Context Probability Information

Huan Fang, Wenjie Su
School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China, 232001

***Abstract*—In real business processes, low quality event logs due to outliers and missing values tend to degrade the performance of process mining related algorithms, which in turn affects the correct execution of decisions. In order to repair the missing values in event logs under the condition that the reference model of the process system is unknown, this paper proposes a method that can repair consecutive missing values. First, the event logs are divided according to the integrity of the trace, and then the cluster algorithm is applied to complete logs to generate homogeneous trace clusters. Then match the missing trace to the most similar sub log, generate candidate sequences according to the context of the missing part, calculate the context probability of each candidate sequence, and select the one with the highest probability as the repair result. When the number of missing items in the trace is 1, our method has the highest repair accuracy of 97.5 percent in the $Small\_log$ and 93.3 percent in the real event logs $bpic20$. Finally, the feasibility of this method is verified on four event logs with different missing ratios and has certain advantages compared with existing methods.**

Keywords*—*Trace clustering; log repairing; process mining; context semantics; conditional probability

I. Introduction

In the past few years, process mining has evolved into a discipline focusing on the discovery, monitoring, and enhancement of real processes [1]. Process mining bridges the gap between traditional data analysis techniques like data mining and business process management analysis [2]. One of the key areas of process mining is process discovery, it aims at generating process models that describe the behavior of process event logs as accurate as possible. Once a model is discovered, process analysis and enhancement can be performed to detect potential improvements [3].

Generally, an event log is composed of a set of traces, while each trace is a sequence of events that occur in business systems. We can label each event by an identifier, named case ID, and all events with the same case ID constitute a trace, where event are arranged by time series. Therefore, each event contains several attributes, such as case ID, activity, resources, and etc. All of event attributes reflect the actual execution information of business processes. Process mining construct model discovery frameworks based on various log processing technologies [4], [5].

In the field of process discovery, most process mining algorithms (such as heuristic mining algorithms, inductive mining, etc.) assume that behaviors related to the execution of underlying processes are correctly stored in event logs [6]–[8]. However, in real business processes, event data inevitably contains noise, and there are many reasons for this situation. For example, manual recording, machine malfunctions, system errors, and network delays, among others. In healthcare systems, errors in medical process event logs are mainly due to manual recording, where the frequency of missing or incorrect case IDs, resource information, and activity tags is higher than that of missing or abnormal timestamps [9], [10]. Thus, low-quality event logs due to outliers and missing values tend to degrade the performance of process mining related algorithms, which in turn affects the correct execution of decisions. It is necessary to address the challenge of improving the quality of event logs, achieving higher-quality business process analysis.

This paper proposes an approach of log repairing method that incorporates context probability information, i.e., the context semantics of event log. The method uses trace clustering technology and is able to repair logs for multiple missing value scenarios. Specifically, all logs are first divided into logs containing only complete traces and logs containing only missing traces. Then, the Levinstein Edit Distance is used to measure the similarity between traces, and a bottom-up hierarchical clustering approach is used to partition complete logs into k sub-logs. Finally, the cluster with the highest similarity to the missing trace is identified from the k sub-logs, and all possible sequences of behaviors containing the missing part are constructed based on the context of the missing part. The optimal repair sequence is selected by solving the context probabilities of each repair activity.

The contributions of our work is focused on:

- A clustering-based approach is proposed to repair multiple consecutive missing activities.
- Behavioural relationships between contexts and activities of arbitrary length in the log are considered.
- Calculate the contextual probability of each repairing activity to select the optimal repair sequence to improve repairing results.

The remainder of this paper is structured as follows. Section II introduces the related work, Section III presents an illustrative motivation example. Section IV reviews some basic concepts and notations, and Section V introduces the proposed method of this work, Section VI conducts experiments and analyses the experimental results. Finally, Section VII concludes this paper.

II. Related Work

In order to improve the quality of process mining analysis, the work in [11] developed a process mining methodology as a guide for projects using event logs for process mining, with a focus on data cleaning steps. Similarly, the work in [12]

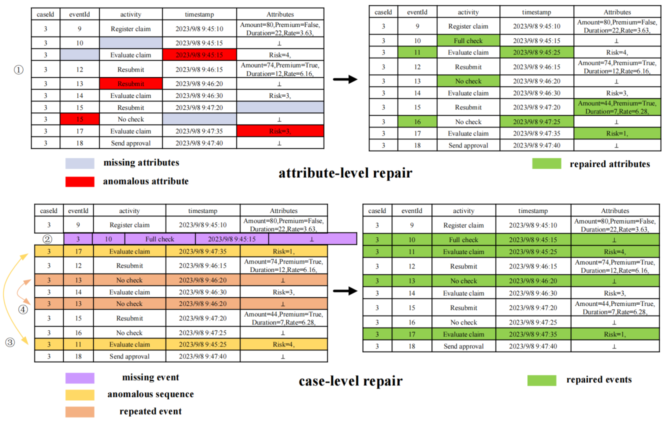

Fig. 1. The framework of proposed Log repairing method.

categorized event log quality problems into four types: missing data, incorrect data, imprecise data, and irrelevant data. In this section, we mainly categorize the existing repairing log work into two types: attribute-level repair and case-level repair.

A. *Attribute-level Repair*

The repair at the attribute level involves explicitly identifying the missing positions of attributes, including activity attributes or determining the positions of abnormal attributes based on anomaly detection of attributes. Specifically, as shown in problem 1 in Fig. 1, the entire table represents a case in the log, where each row represents an event and each cell represents the attributes contained in the event. The red cells in Fig. 1 indicate detected abnormal attributes, while the gray cells indicate missing attributes.

In [13], detecting abnormal values and reconstructing missing values at the attribute level in event logs are focused on. It consists of two processes: log cleaning and reconstruction. In event log cleaning, abnormal values are those with high reconstruction errors in the autoencoder decoding step. In event log reconstruction, the autoencoder is used to reconstruct missing values in the input dataset. This method does not rely on any prior knowledge of the business process that generates the event log and has shown significant performance in terms of activity labels and timestamps in artificial event logs. In PROELR [14] and SRBA [16], trace clustering methods are used to cluster complete traces, which are traces without any missing activity labels. Each incomplete trace, referring to traces with missing activity labels, is assigned to the nearest cluster. Subsequently, the incomplete traces are repaired based on the characteristics of the corresponding trace clusters. It is worth noting that both [14] and [16] can only repair missing values at the individual activity level. MIEC [17] is a likelihood-based multiple imputation technique for repairing missing data in event logs. In addition to repairing missing activity labels, it can also repair all other missing attributes in the event log. MIEC relies on the dependencies between event attributes. For example, certain activities may always occur on weekends or be performed by specific groups of people. When such dependencies do not exist or the event log contains limited attribute data, effective repair of the event log may not be possible. In [18], a decision tree learning algorithm is proposed to discover rules for missing values in event logs. In [19], a novel classification event imputation method is proposed, which can recover missing categorical events by learning structural features observed in the event log. In [20], an LSTM-based prediction model that uses the prefix and suffix sequences of events with missing activity labels as input to predict the missing labels is proposed, demonstrating high repair capability. In [21], the BERT4Log model and weak behavior profile theory, combined with a multi-layer multi-head attention mechanism is introduced, for interpretable repair of low-quality event logs. In [22], a convolutional neural network model that incorporates trace behavior features to repair missing activities in traces is proposed, and its core idea is to transform the event log of a business process transition into spatial data based on the dimensions of time attributes and activity attributes, convert it into an image matrix, and train a convolutional neural network model to predict the missing activities.

B. *Case-level Repair*

The existing techniques for case-level repair mainly focus on solving problem 2 in Fig. 1, where there is a missing entire event in a case. A method combining random Petri nets, alignment, and Bayesian networks was proposed in [23] to recover missing activities and timestamps in event logs. The work in [24] developed advanced indexing and pruning techniques to reduce the search space, and the work in [25] utilizes process decomposition techniques and heuristic methods to effectively prune unfeasible sub-processes that fail to produce minimal repairs. Both works of [24] and [25] aimed at minimizing the search space as much as possible to improve the efficiency of repairing events.

C. *Summary of Existing Work*

The attribute-level repair techniques are effective in repairing known anomalies or missing attributes but are unable to handle cases where the missing information is unknown or when there are sequence anomalies and activity repetitions, as shown in problems 3 and 4 in Fig. 1. On the other hand, case-level repair techniques rely heavily on process models and may not perform well in the absence of a process model.The specific comparison of existing techniques is shown in Table I, where the symbol ✓ represents the scope of techniques considered. The specific meanings of the symbols are as follows:

F1: Deterministic repair for known anomalies or missing attributes, where the position of the anomaly or missing attribute is clearly identified.

F2: Recovery of a single attribute's missing values in a trace, mainly activity names.

F3: Recovery of multiple attributes' missing values in a trace.

F4: Incorporation of process models.

F5: Uncertain repair for missing attributes, where the position of the missing attribute is unknown.

F6: Uncertain repair, where the position of the missing attribute and the existence of event repetitions or sequence changes are unknown.

F7: Interpretable missing attribute repair.

Based on the aforementioned issues, this paper introduces the concept of an activity feature graph to detect and repair the mentioned problems. It utilizes activity feature graphs to compare abnormal behaviors with repaired behaviors, thereby identifying the causes of anomalies. Additionally, it enables further analysis of the impact of data. Furthermore, this paper also proposes the recovery of average behavior characteristics for missing values in the activity feature graph.

III. Motivation

Table II presents an event log containing 9 activities $a, b, c, d, e, f, g, h, i, j$ and 10 trace variants. The superscript of a trace denotes its occurrence frequency, and the symbol $-$ represents missing value. Take the missing trace $\langle a, i, e, -, g, f\rangle$ for example, if the repair method of [14] is used, the repair result should be $\langle a, i, e, d, g, f\rangle$, as the highest frequency activity occurring between activities e and g is d. The repair method of [14] repairs the missing activity based on the highest frequency of occurrence between the predecessor and successor of the missing activity. However, this result is incorrect. The reason is that the method of [14] can only repair one missing activity based on its predecessor and successor activities, which may not represent the key information of the missing trace. If the method from this paper is used with a context length of 2, the correct repair result can be obtained as $\langle a, i, e, k, g, f\rangle$, which is the correct one.

Furthermore, the repairing work of [8] selects the most frequently occurring segment between the preceding and succeeding contexts of the missing portion to repair the missing trace. The drawback of this method is that, although it considers more contextual information, it can only repair the missing trace variants occurring more than 2 times, and ignores the behavioral relationships between activities. For example, for the missing trace $\langle a, i, e, -, -, h, f\rangle$, the original log in Table I does not find the corresponding length of the missing segment when the method of [8] is directly used. Comparatively, when the proposed method in this paper is used, a kind of behavioral graphs of activities is constructed, and then the repaired result based on contextual probabilities is calculated as $\langle a, i, e, d, k, h, f\rangle$.

Therefore, in order to extend the research content of existing research, this paper proposes a clustering-based method that can repair multiple consecutive missing activities. The proposed method not only considers arbitrary lengths of context in the logs but also takes into account the behavioral relationships between activities. Experimental results show that compared to existing research, the proposed method has significant advantages in repairing missing activities.

IV. Preliminaries

In this section, we briefly review a couple of terminologies such as events, traces, event log, log clustering and missing trace, in order to ease the readability of this paper.

A business process is a set of activities executed in a given setting to achieve predefined business object [26]. An activity is an expression of the form $Act(a_1, a_2, \cdots, a_{n_A})$, where Act is the activity name and each a_i is an attribute name. We call n_A the arity of A. The attribute names of an activity are all distinct, but different activities may contain attributes with matching names.

We assume a finite set A of activities, all with distinct names; thus, activities can be identified by their name, instead of by the whole tuple. Every attribute a_i of an activity A is associated with a type $D_A(a_i)$, i.e., the set of values that can be assigned to a_i when activity is executed.

An event is the execution of an activity and is formally captured by an expression of the form $e = A(v_1, v_2, \cdots, v_{n_A})$, where $A \in Act$ is an activity name with $v_i \in D_A(a_i)$.The set of events is denoted as $Event$.

A trace is formally defined as finite sequences of events $\sigma = \langle e_1, e_2, \cdots, e_n\rangle$ with $e_i = A_i(v_1, v_2, \cdots, v_{n_{A_i}})$. Traces model process executions, i.e., the sequences of activities performed by a process instance CID. A finite collection of executions into a set L of traces is called an event log.

Definition 1 (Levenshtein Distance): Let $\sigma_1, \sigma_2 \in L$ be two traces in the log L, $Lev(\sigma_1, \sigma_2)$ denotes the minimum number of edit operations required to transform σ_1 to σ_2, which is the edit distance. There are three types of edit operations: delete, insert, and replace.

For example, for $\sigma_1 = \langle a, c, d, e, h\rangle$, $\sigma_2 = \langle a, b, d, e, g\rangle$, after two replacement operations on σ_1, it becomes σ_2, thus the edit distance $Lev(\sigma_1, \sigma_2) = 2$.

Definition 2 (Log Clustering): Let $CN = \{C_1, C_2, \cdots, C_m\}$ be a set of clusters, and m be the number of the clusters. If CN is the clustering result of log L, then CN is a clustering of L if and only if $C_i \bigcap C_j = \emptyset$ $(1 \le i, j \le n \land i \ne j)$ and $\bigcup_{i=1}^{n} C_i = L$.

Definition 3 (Missing Trace): A missing trace be $\sigma^* = \langle e_1, e_2, \cdots, e_n$ and n be the length of σ^*, if and only if $\exists i \in [1, n] : e_i =' -'$, where $'-'$ denotes the missing value $null$. The incomplete log L^* is the set of all missing traces.

For example, $L_1 = \{\langle a, d, b, -, h\rangle^{56}, \langle a, d, c, -, -, c, d, e, h\rangle^{8}, \langle a, d, c, -, -, -, b, e, g\rangle^{2}\}$ is an incomplete log.

V. The Proposed Log Repairing Method

This section introduces the proposed method for repairing multiple consecutive missing activities based on clustering and context integration. The research framework is shown in Fig. 2, where the original log is first divided into complete logs and missing logs, and then clustering methods are used to segment the complete log into several sub-logs. Subsequently, each missing trace in the missing log is matched with the most similar sub-log. Next, a behavioral graph is constructed based on the context of the missing parts and the behavioral relationships of similar clusters, generating candidate repair sequences. Finally, the conditional probabilities of each candidate sequence are calculated in conjunction with the context, and the candidate sequence with the highest probability is selected to repair the missing trace.

A. Log Clustering

Clustering is used to partition process logs into trace clusters, which helps reduce heterogeneity and improve un-

TABLE I. THE COMPARISON OF EXISTING TECHNIQUES

Repair type	Existing technologies	F1	F2	F3	F4	F5	F6	F7
Attribute-level repair	[13], [17], [19], [20]	✓		✓				
	[14], [16], [22]	✓	✓					
	[18]						✓	
	[21]	✓	✓					✓
Case-level repair	[23]–[25]			✓	✓	✓		
The methods in this paper				✓		✓	✓	

TABLE II. AN EXAMPLE OF EVENT LOG

ID	Traces
1	$\langle a, c, b, e, d, h, f\rangle$
2	$\langle a, c, b, e, d, g, f\rangle^2$
3	$\langle a, i, b, e, d, g, f\rangle$
4	$\langle a, i, e, k, g, f\rangle$
5	$\langle a, i, c, b, h, k, f\rangle^3$
6	$\langle a, b, c, g, d, j\rangle^2$
7	$\langle a, i, c, e, d, k, g, j\rangle$
8	$\langle a, c, b, e, b, d, k, g, f\rangle$
9	$\langle a, i, c, g, k, j\rangle$
10	$\langle a, b, c, i, k, h, j\rangle$

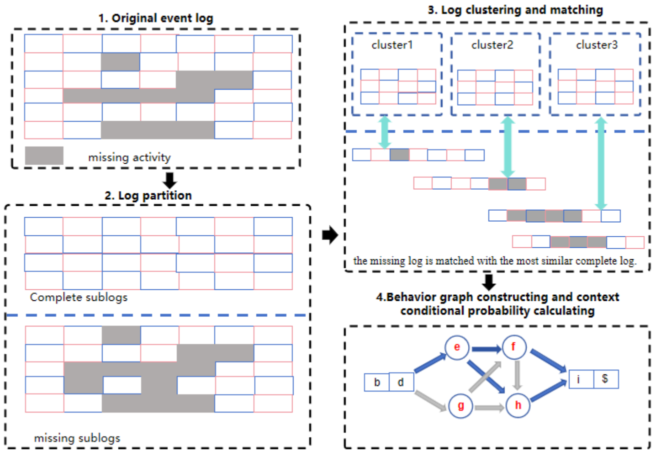
1. Original event log
missing activity
2. Log partition
Complete sublogs
missing sublogs
3. Log clustering and matching
cluster1
cluster2
cluster3
the missing log is matched with the most similar complete log.
4.Behavior graph constructing and context conditional probability calculating
b
d
e
f
g
h
i
$

Fig. 2. The framework of proposed log repairing method.

derstandability [16]. Trace clustering splits observed different behaviors into several groups of multiple sub-logs with similar behaviors. This paper aims to identify a group of activity sequences similar to the missing trace from the complete logs, in order to enhance the accuracy of missing trace repairing. Therefore, this paper uses trace clustering as a preprocessing stage for log repairing. Firstly, the traces are encoded using Bag of Activity (BOA) encoding, and the similarity between traces is measured using Euclidean distance to construct a similarity matrix of traces. Then, spectral clustering is used for clustering.

Definition 4 (Similar Clusters): Let $\sigma^* \in L^*$ be a missing trace, then $C_i \in CN$ is the cluster most similar to σ^*, if and only if C_i satisfies eq (1).

$$C_i = argmin_{i=1}^{|CN|} \frac{\Sigma_{\sigma \in C_i} Lev^*(\sigma, \sigma^*)}{|C_i|} \tag{1}$$

Where, $Lev^*(\sigma, \sigma^*) = \frac{Lev(\sigma,\sigma^*)}{\max(|\sigma|,|\sigma^*|)}$ represents the normalized edit distance, and $Lev^*(\sigma, \sigma^*) \in [0, 1]$.

In Algorithm 1, lines 1-5 use Euclidean distance to calculate the similarity between traces represented by BOA encoding, using SM to store the similarity values between traces, and then use spectral clustering to divide the complete log into n sub-logs. Lines 8-13 select the most similar cluster based on the minimum edit distance SV from the missing trace to the traces in each cluster according to Eq. (1).

Algorithm 1: Locating the candidate cluster C^* that most similar to the missing trace MT

```
   Input: Original log $L'$, Number of clusters $n$,
          Missing trace $MT$
   Output: Candidate cluster $C^*$
1  $SM \leftarrow null$;
   // Initialize the similarity matrix
2  foreach $\sigma_i \in L'$ do
3      foreach $\sigma_j \in L'$ do
4          $SM(i, j) \leftarrow Euclidean(\sigma_i, \sigma_j)$;
5      end
6  end
7  $CN \leftarrow SC(SM)$;
8  $C^* \leftarrow null$;
9  $SV \leftarrow 0$;
10 foreach $C_i \in CN$ do
11     $S_i \leftarrow (\sum_{\sigma \in C_i} Lev^*(\sigma, MT))/|C_i|$;
12     if $SV > S_i$ then
13         $SV \leftarrow S_i$;
14         $C^* \leftarrow C_i$;
15 end
16 return $C^*$;
```

B. Repairing Missing Traces by Context Probability

In order to achieve more accuracy and reduce the complexity of repairing missing events, this paper defines the concepts of begin sequence and end sequence in relation to the behavior graph. The begin and end sequences refer to the activity sequence around the missing part, while the behavior graph stores the solution space that satisfies all possible activity behaviors from the start sequence to the end sequence in the similar cluster.

Definition 5 (begin sequence, end sequence): Given a missing trace $\sigma \in A^*$, the contextual behavior of the missing part is defined as $con(\sigma, l, r) = \{(\sigma', \sigma'')|\sigma', \sigma'' \subseteq \sigma \wedge |\sigma'| = l \wedge |\sigma''| = r\}$, where σ' corresponds to the begin sequence of σ with precede length l, and σ'' corresponds to the end sequence of σ with successor length r.

For example, given a missing trace $\sigma = \langle a, d, c, -, -, -, b, e, g\rangle$, the context of its missing parts is denoted as $con(\sigma, 2, 3)$, $con(\sigma, 2, 3) = (\langle d, c\rangle, \langle b, e, g\rangle)$, where $\langle d, c\rangle$ represents the Begin Sequence (BS) with precede length 2 and $\langle b, e, g\rangle$ represents the End Sequence (ES) with successor length 3.

To identify the activity sequences of the missing parts in the missing trace, first, in the candidate clusters, identify the activity sequences $SeqSet$ that satisfy from the BS to the ES, i.e., $SeqSet = \{\phi(\sigma_i, BS, ES)|\exists\phi(\sigma_i, BS, ES) \in \sigma_i \wedge \sigma_i \in C^* \wedge 1 \leq i \leq |C^*|\}$, where $\phi(\sigma_i, BS, ES) = \{\sigma_i(o, p)|o = PosB(\sigma_i, BS) \wedge p = PosE(\sigma_i, ES)\}$, $PosB(\sigma_i, BS)$ indicates the index where the begin sequence BS first appears in trace σ_i, $PosE(\sigma_i, ES)$ indicates the index where the end sequence ES last appears in trace σ_i, and $\sigma_i(o, p)$ represents the activity sequence in trace σ_i between indices o and p. Then, a behavior graph is constructed based on $SeqSet$.

Definition 6 (Behavior Graph): The behavior graph G is a directed graph, i.e., $G = DiGraph(V, E)$. The nodes set V represent the activities in $SeqSet$, the directed edges $E = \{(x, y)|\exists x \rightarrow y \in SeqSet\}$, $x \rightarrow y$ indicates that activity y immediately follows activity x.

For the missing trace $\langle \#, \#, a, c, d, e, f, b, d, -, -, \$, \$\rangle$, for ease of computation, we added the symbols $\#$ and $\$$ at the beginning and end of the trace, respectively, representing the precede context at the start and the successor context at the end.

Algorithm 2 describes the process of finding all possible valid behavior sequences from the initial sequence to the final sequence. Lines 1-10 define a method $deep\_search$, used to calculate a depth-first search starting from a certain activity with a length of $(n + 2)$. Line 14 indicates searching for the first activity in the initial sequence, for example, for the missing trace $mt = \langle \#, \#, a, c, b, e, -, -, f, \$, \$\rangle$, when the context lengths are set to 2, searching procedure starts from activity b in the original sequence, returning all found behavior sequences $repair\_seq = \{\langle b, e, d, g, f, \$\rangle, \langle b, e, d, h, f, \$\rangle\}$. Lines 15-17 filter the found behavior sequences based on the length of the final sequence and the missing trace, finally obtaining candidate repairing sequences $CT = \{\langle b, d, e, h, \$, \$\rangle, \langle b, d, e, g, \$, \$\rangle\}$.

Algorithm 2 can generate candidate repair sequences for missing traces. In order to select the most suitable missing sequence, this paper adopts eq. (2) to calculate the next activity under the current window based on the context of the missing parts.

Definition 7 (Context probability): Given a sliding window size of w, a candidate trace $ct \in CT$, and the cluster C that is most similar to ct, the context probability of act under the window w is denoted as $CoverProbably(act)$.

$$CoverProbably(act) = \frac{|Wact(ct, w) \cup Next(Wact(ct, w)) \in C|}{|Wact(ct, w) \in C|} \quad (2)$$

Algorithm 2: Generating candidate repair sequences

```
Input: Behavior Graph $G$, begin sequence $BS$, end
       sequence $ES$, number of missing values $n$
Output: Candidate repair traces $CT$
1  def $deep\_search(h, k, current, G)$: // search
       from $current$, $h$ is the current
       depth
2  $repair\_seq \leftarrow \{\}$;
3  if $h \leq k$ then
4      foreach $sct \in G.node()$ do
5          if $\exists current \rightarrow act$// Nodes with paths
               to the current activity
6          then
7              $temp\_seq \leftarrow$
                 $deep\_search(h + 1, k, act, G)$;
8              foreach $seq \in temp\_seq$ do
9                  $misseq \leftarrow current \cup \{seq\}$;
10                 $repair\_seq.add(misseq)$;
11             end
12     end
13 return $repair\_seq$;
14 $CT = \{\}$;
15 $Bact = Firact(BS)$;
   // The first activity in the initial
       sequence
16 $Elen = len(ES)$; $Blen = len(BS)$;
   // length of the begin sequence, end
       sequence
17 $repair\_seq = deep\_search(0, n + 2, Bact, G)$;
18 foreach $seq \in repair\_seq$ do
19     if $seq[Elen :] == ESandlen(seq) == n + 2$
       then
20         $CT.add(seq)$;
       // Evaluate candidate sequences
21 end
22 return $CT$;
```

Where $act \in A$, $A = \{ct(i)|w \leq ilen(ct)\}$, $Wact(ct, w)$ represents the activity sequence in the candidate trace ct under the window size of w. $Next(Wact(ct, w))$ represents the next activity in this sequence. The probability of the candidate trace ct sliding backward is denoted as $BP(ct)$, shown as eq. (3).

$$BP(ct) = \Pi_{act \in A} CoverProbably(act) \quad (3)$$

Similarly, the probability of the candidate trace ct sliding forward is denoted as $FP(ct)$, $FP(ct) = BP(ct^{-1})$, where ct^{-1} represents the reverse order of the candidate trace activity sequence ct. Therefore, the final context probability of the candidate trace ct is calculated as $P(ct)$, $P(ct) = \frac{BP(ct)+FP(ct)}{2}$.

Algorithm 3 describes the process of calculating the conditional probability to select the final repair result from the candidate repair sequences. Lines 1-5 initialize parameters. Lines 6-9 calculate the probability of the next activity under the current window from forward and backward directions. The conditional probability calculated by the forward sliding is saved in FP, and the conditional probability calculated by the backward sliding is saved in BP. Then, the average of

Algorithm 3: Context probability calculating algorithm

```
Input: most similar cluster $C$, Sliding window size $w$, Candidate repair sequence $CT$
Output: Predicted sequence $pre\_ct$
1  $pre\_ct \leftarrow null$;
2  $max\_P \leftarrow 0$;
3  foreach $ct \in CT$ do
4      $BP \leftarrow 1$;
5      $FP \leftarrow 1$;
6      while $Next(Wact(ct, w)! = null)$ do
7          $BP \leftarrow BP * CoverProbably(Next(Wact(ct, w))$;
8          $FP \leftarrow FP * CoverProbably(Next(Wact(ct^{-1}, w))$;
9      end
10     $P = (BP + FP)/2$;
11     if $max\_P < P$ then
12         $max\_P \leftarrow P$;
13         $pre\_ct \leftarrow ct$;
14 end
15 return $pre\_ct$;
```

the probabilities calculated from both directions is taken as the repair probability P for that candidate repair sequence. Lines 10-13 select the candidate sequence with the highest probability as the repair result and return it.

In order to illustrate the main idea of proposed method, an example is taken here. Suppose the missing trace $mt = \langle \#, \#, a, c, b, e, -, -, f, \$, \$ \rangle$ is generated from Table I that to be repaired. Fig. 3 shows the distance matrix obtained by calculating the Euclidean distance after encoding the logs in Table II using BOA, and then generates a weighted undirected graph based on spectral clustering, where nodes represent the IDs of 10 variants, edges represent the distances between traces. The original event log is divided into 2 clustering results, with the most similar cluster $\{1, 2, 3, 4, 6, 8\}$ determined based on eq. (3). Next, setting the context of the missing trace as $con(mt, 2, 2) = (\langle b, e \rangle, \langle f, \$ \rangle)$, and candidate repair sequences are obtained as $CT = \{\langle b, e, d, g, f, \$ \rangle, \langle b, e, d, h, f, \$ \rangle\}$ through Algorithm 2. Finally, Algorithm 3 is used to calculate the conditional probability.

Tables III and IV record the context probability of candidate repair sequences with a window size of 2, with the probability of the candidate repair sequence $ct_1 = \langle b, e, d, g, f, \$ \rangle$ being $3/5$, and $ct_2 = \langle b, e, d, h, f, \$ \rangle$ being $1/5$. Therefore, the final repair result is $\langle \#, \#, a, c, b, e, d, g, f, \$, \$ \rangle$.

VI. Experimental Evaluation

In this section, a series of experiments are conducted to validate the feasibility of the proposed method. Firstly, the repair effectiveness of traces with different missing ratios is tested on four kinds of event logs. Secondly, the impact of different context lengths and whether clustering preprocessing is conducted on repairing missing traces is compared.

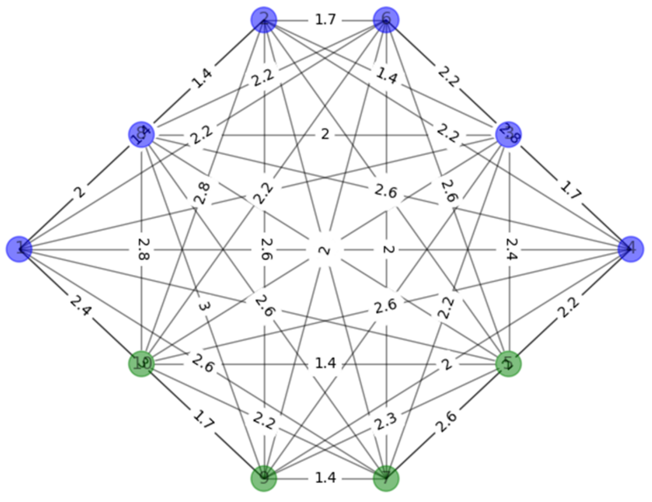

Fig. 3. Weighted undirected graph generated by spectral clustering method.

TABLE III. The Contextual Conditional Probability between Candidate Traces $ct_1 = \langle b, e, d, g, f, \$ \rangle$

Direction	Sliding window	Conditional probability	Result
Backward sliding	$be \to d$	$4/5$	$3/5$
	$ed \to g$	$3/4$	
	$dg \to f$	1	
	$gf \to \$$	1	
Forward sliding	$\$f \to g$	$4/5$	$3/5$
	$fg \to d$	$3/4$	
	$gd \to e$	1	
	$de \to b$	1	

A. Logs Information

Table V describes the basic information of one synthetic log and three real logs. These logs come from different environments to compare the feasibility of the proposed method in different environments. $Small\_log$ is an artificial log generated by the plg tool that comes from the literature [15]. BPI Challenge 2020 ($bpic20$) contains event data related to travel expense reimbursement over two years. $sepsis$ records sepsis case events from a hospital's ERP system, with each trace corresponding to a case. BPI Challenge 2013, incidents ($bpic13\_inc$), consists of event logs of Volvo IT incidents and problem management.

B. Effectiveness Evaluation

In this paper, each experimental log is divided into complete logs containing complete traces and incomplete logs containing missing traces. The experiment divided them in this way three times, with the proportion of incomplete logs in each division being 5%, 10%, and 15% of the entire log respectively. Secondly, this paper randomly deletes some activities from each trace, in order to simulate the situation of lost activities in a real environment, which generates the incomplete logs. In experimental evaluation, three random deletion ratios have been performed on the traces in the incomplete logs of each log, with each deletion removing 1 to 3 activities from the trace.

TABLE IV. The Contextual Conditional Probability between Candidate Traces $ct_2 = \langle b, e, d, h, f, \$\rangle$

Direction	Sliding window	Conditional probability	Result
Backward sliding	$be \to d$	4/5	1/5
	$ed \to h$	1/4	
	$dh \to f$	1	
	$hf \to \$$	1	
Forward sliding	$\$f \to h$	1/5	1/5
	$fh \to d$	1	
	$hd \to e$	1	
	$de \to b$	1	

TABLE V. Description of Event Logs

Event log	number of traces	Trace variants	Average length of traces	Number of activities
small_log	2000	12	14	14
*Bpic*20	2099	202	8.693	29
*Bpic*13_*inc*	7554	2278	8.675	13
sepsis	1050	846	14.49	16

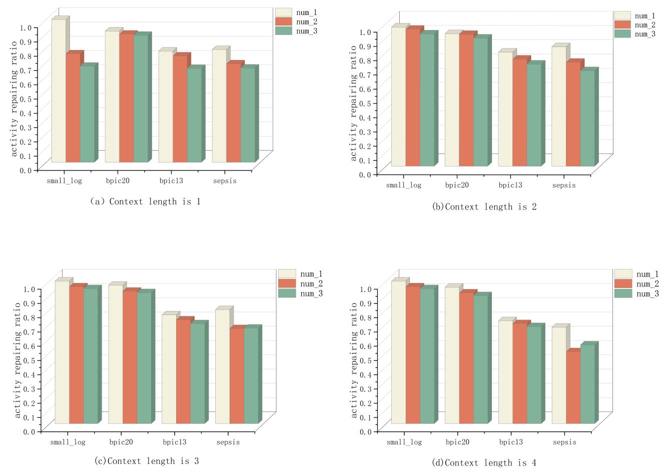

Fig. 4. Repair effects with different numbers of missing items under 10% missing ratio.

Fig. 4 depicts the repair situation of each event log using the method proposed in this paper under different missing ratios and varying numbers of missing activities per trace. From the Fig. 4, it can be seen that the best repair effect is achieved when each trace is missing one activity. As the number of missing activities increases, the accuracy of repair gradually decreases. This result is expected because with more missing activities in the trace, there are potentially multiple behavior combinations that could exist, making it challenging to obtain the true repair sequence solely based on the control flow. Additionally, from this result, it can be observed that the repair effect of synthetic logs using the method proposed in this paper is superior to that of real event logs. This is because real logs are generated by complex systems, characterized by high concurrency, a wide range of event types, and inherent abnormal situations, all of which can influence the accuracy of repair.

TABLE VI. Repairing Effects under Different Missing Ratios

Number of missing items in traces	Missing ratio	*Small_log*	*Bpic*13	*Bpic*20	*sepsis*
1	5%	0.975	0.828	0.933	0.754
	10%	0.957	0.8	0.929	0.838
	15%	0.92	0.783	0.857	0.81
2	5%	0.945	0.782	0.967	0.698
	10%	0.925	0.751	0.924	0.729
	15%	0.902	0.731	0.917	0.655

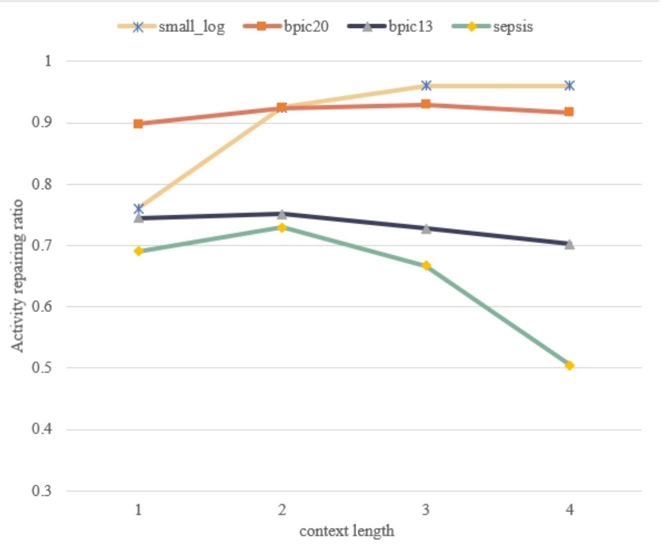

Fig. 5. Repairing effects under different context lengths.

Table VI respectively displays the repair effects of traces missing 1 and 2 activities under log missing rates of 5%, 10%, and 15%. From the data in Table VI, it can be observed that as the missing rate of the log increases, the repair rate of the activities gradually decreases. Moreover, as the number of missing activities per trace increases, the repair effect decreases as well. Therefore, both the missing rate of the log and the number of missing activities per trace will have an impact on the log's repair.

Fig. 5 illustrates the impact of different context lengths on repair. Four different context lengths were set in this experiment, and the results show that the accuracy of repair is lower when the context length is set to 1. This is because it is difficult to determine the main information of the current trace with just one start and end activity. As the context length increases, the repair effect gradually improves, as a certain length of context contains the main information of the trace and can filter out activity sequences more similar to the missing trace. For logs *bpic*13 and *sepsis*, a decrease in repair rate occurs when the context length reaches 4. This is because these two logs have a high number of variants, and longer context lengths filter out a large number of candidate traces, making it difficult to find similar behavioral relationships.

Fig. 6 presents the comparison between the method of

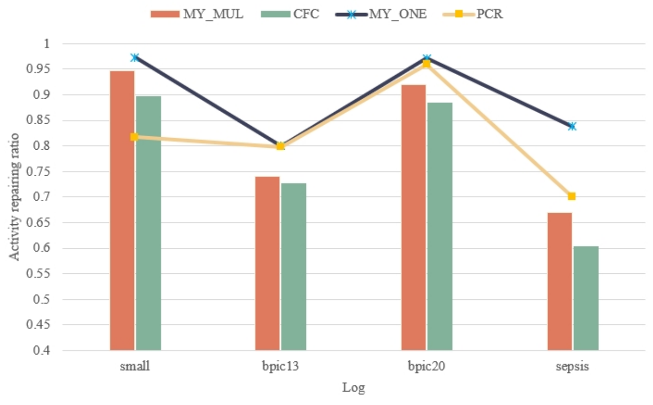

Fig. 6. Comparison of repairing methods.

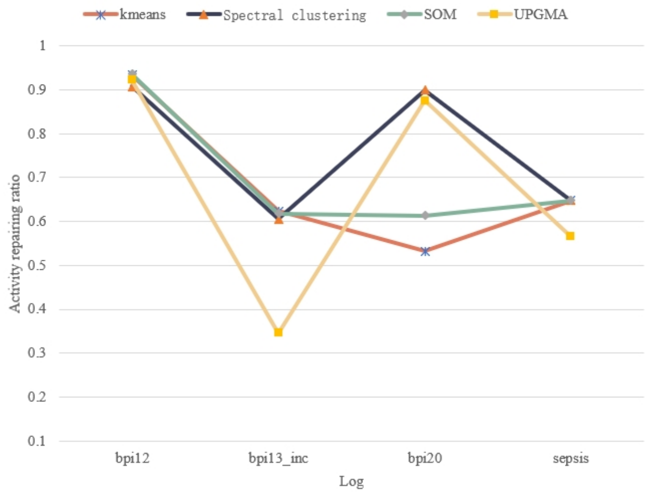

Fig. 7. The impact of different clustering methods on repairing effectiveness.

repairing multiple missing activities (MY_MUL) proposed in this paper and the method of repairing single activity (MY_ONE), compared with the methods named CFC[13] and PCR [12], respectively.It can be seen that our proposed method outperforms existing methods both in the case of repairing multiple activities or a single activity. And the method of spectral clustering has a more significant improvement over other existing clustering methods for preprocessing of missing activity repair. Our proposed method has a good advantage in repairing multiple consecutive missing events, but for the time being, only the direct consecutive relationship between the activities is considered, and the long-term dependency between them is not taken into account. Furthermore, Table VII illustrates the improvement in repair after clustering preprocessing. Dividing the logs and matching the missing traces to more similar clusters can reduce the impact of unnecessary trace pairs on the repair effect.

TABLE VII. The Impact of Clustering Preprocessing on Repairing Effectiveness

Method	$Small\_log$	$Bpic13$	$Bpic20$	$sepsis$
clustering	0.957	0.751	0.924	0.729
No clustering	0.92	0.557	0.8	0.562
Enhancement	4%	34.8%	15.5%	29.7%

Fig. 7 describes the impact of using different clustering methods, such as $kmeans$,$spectralclustering$, SOM, $UPGMA$[17] method, on the repairing effectiveness of logs. It can be seen from the Fig. 7 that the repair method based on spectral clustering is better in the four logs.

VII. Conclusion

This paper proposes a method for repairing multiple missing activities in event logs without relying on process models. We use spectral clustering to partition the complete event log into sub-logs with similar behaviors, identifying the most similar cluster based on the minimum Levenshtein edit distance between the missing trace and traces in these clusters.

Behavior sequences are then constructed in the context of the missing parts. Using a bottom-up hierarchical clustering method, we refine the sub-logs and identify clusters most similar to the missing trace. The best repair sequence is determined by solving the contextual probability of each repair activity, examining relationships between contexts and activities of arbitrary lengths. A sliding window technique predicts the next activity based on the current context, averaging forward and backward probabilities to select the sequence with the highest likelihood.

Our method effectively repairs multiple consecutive missing events by considering direct dependencies and contextual semantics. However, this paper only considers control flow dependencies between events and ignores data flow dependencies. In future work, in order to consider data flow dependencies of events, some more complicated sequential patterns of traces will be further investigated.

Acknowledgment

This work was supported by the National Key R&D Program of China(Nos. 2023YFC3807500 and 2023YFC3807501).

We also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] Van Der Aalst, Wil MP, *Process Mining: Data Science in Action*, Springer Publishing Company, Incorporated, 2016.

[2] Van Der Aalst, Wil MP, *Using Process Mining to Bridge the Gap between BI and BPM*, Computer, 44(12), 2011, pp. 77-80.

[3] Chapela-Campa D , Mucientes M , Lama M, *Understanding complex process models by abstracting infrequent behavior*, Future Generation Computer Systems, 2020.

[4] Sani M F, Zelst S J V , Van Der Aalst, Wil MP. *Improving Process Discovery Results by Filtering Outliers Using Conditional Behavioural Probabilities*, Springer, Cham, 2017.

[5] Wang J, Song S, Lin X, et al, *Cleaning structured event logs: A graph repair approach*, 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015.

[6] Wang J, Song S, Zhu X, et al, *Efficient Recovery of Missing Events*, IEEE Transactions on Knowledge & Data Engineering, 2016, 28(11), pp.2943–2957.

[7] Song W, Xia X, Jacobsen H A, et al, *Heuristic Recovery of Missing Events in Process Logs*, IEEE International Conference on Web Services, IEEE, 2015, pp. 105–112.

[8] Fani Sani M, Zelst S J, van der Aalst W M P, *Repairing outlier behaviour in event logs*, International Conference on Business Information Systems, Springer, Cham, 2018, pp. 115–131.

[9] Song W, Jacobsen H A , Zhang P. *Self-Healing Event Logs*, IEEE Transactions on Knowledge and Data Engineering, 2019, (99), pp. 1–1.

[10] Reijers H A, Mendling J, Dijkman R M, *Human and automatic modularizations of process models to enhance their comprehension*, Information Systems, 2011, 36(5), pp. 881–897.

[11] Van Eck, Maikel L and Lu, Xixi and Leemans, Sander JJ and Van Der Aalst, Wil MP, *PM: a process mining project methodology*, International conference on advanced information systems engineering, Springer, 2015, pp. 297–313.

[12] Bose, RP Jagadeesh Chandra and Mans, Ronny S and Van Der Aalst, Wil MP, *Wanna improve process mining results?*. 2013 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, 2013, pp.127–134.

[13] Nguyen, Hoang Thi Cam and Lee, Suhwan and Kim, Jongchan and Ko, Jonghyeon and Comuzzi, Marco, *Autoencoders for improving quality of process event logs*, Expert Systems with Applications, Elsevier, 2019, 131, pp.132–147.

[14] Xu, Jiuyun and Liu, Jie, *A profile clustering based event logs repairing approach for process mining* IEEE Access, 2019, 7, pp. 17872–17881.

[15] A. Weijters, J. T. S. Ribeiro, Flexible heuristics miner (fhm), in: 2011 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, 2011, pp. 310–317.

[16] Liu J, Xu, J, Zhang R and R M Stephan, *A repairing missing activities approach with succession relation for event logs*, Knowledge and Information Systems, Springer, 2021, 63(2), pp.477–495.

[17] Sim Sunghyun, Bae Hyerim and Choi, Yulim, *Likelihood-based multiple imputation by event chain methodology for repair of imperfect event logs with missing data*, 2019 International Conference on Process Mining (ICPM), IEEE, 2019, pp.9–16.

[18] Horita Hiroki, Kurihashi Yuta and Miyamori, Nozomi, *Extraction of missing tendency using decision tree learning in business process event log*,Data, MDPI, 2020, 5(3), pp. 82–82.

[19] Sim Sunghyun, Bae Hyerim and Liu, Ling, *Bagging recurrent event imputation for repair of imperfect event log with missing categorical events*, IEEE Transactions on Services Computing, IEEE, 2023, 16(1), pp.108–121.

[20] Lu Y, Chen Q and Poon Simon K, *A deep learning approach for repairing missing activity labels in event logs for process mining*, Information, 2022, 13(5), pp. 234–234.

[21] Li B, Fang H and Mei Z, *Interpretable Repair Method for Event Logs Based on BERT and Weak Behavioral Profiles*, Computer Science, 2023, 50, pp.38-51.

[22] Liu W, Fang H and Zhang S, *Missing activity log repair method based on image data using CNN*, Computer Integrated Manufacturing Systems, 2023.

[23] R S Andreas, Mans R S, van der Aalst, Wil MP and Weske Mathias, *Improving documentation by repairing event logs*, The Practice of Enterprise Modeling: 6th IFIP WG 8.1 Working Conference, PoEM 2013, Riga, Latvia, November 6-7, 2013, Springer, pp. 129–144.

[24] Wang J, Song S, Zhu X and Lin X, *Efficient recovery of missing events*, Proceedings of the VLDB Endowment, 2013, 6(10), pp. 841–852.

[25] Song W and Xia X, Jacobsen Hans-Arno, Zhang P and Hu H, *Heuristic recovery of missing events in process logs*, 2015 IEEE International Conference on Web Services, IEEE, 2015, pp. 105–112.

[26] Fang H, Liu W, Wang W and Zhang S, *Discovery of process variants based on trace context tree, Connection Science*, 35(1), 2023, pp.1–29.