IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2024, accepted 4 June 2024, date of publication 7 June 2024, date of current version 19 June 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3411089*

RESEARCH ARTICLE

# Enhancement in Process Mining Model by Repairing Noisy Behavior in Event Log

**SHABNAM SHAHZADI[1], WALID EMAM[2], USMAN SHAHZAD[3,4], SOOFIA IFTIKHAR[5], ISHFAQ AHMAD[4], AND GAURAV SHARMA[6]**

[1]Department of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 230001, China
[2]Department of Statistics and Operations Research, Faculty of Science, King Saud University, Riyadh 11451, Saudi Arabia
[3]Department of Statistics, PMAS-Arid Agriculture University at Rawalpindi, Rawalpindi 46300, Pakistan
[4]Department of Mathematics and Statistics, International Islamic University at Islamabad, Islamabad 44000, Pakistan
[5]Department of Statistics, Shaheed Benazir Bhutto Women University, Peshawar 00384, Pakistan
[6]Department of Computer Science and Engineering, Seth Jai Parkash Mukand Lal Institute of Engineering and Technology, Haryana 135133, India

Corresponding author: Shabnam Shahzadi (shabnamsarwar2@gmail.com)

This work was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSPD2024R749.

**ABSTRACT** Companies and organizations aim to improve the performance of their business processes to stay competitive. Recently, researchers have shown significant interest in process mining, particularly its ability to extract accurate information from process-related data. Process enhancement is a crucial aspect of process mining, involving the extraction of information from the actual process event log to extend or improve existing processes. Enhancement can be classified into two types: extension and repair. This paper specifically focuses on the repair type of enhancement. Information systems commonly encounter logging errors or exhibit special behaviors that introduce noise into the event log. In this research, we investigate the process mining model in the presence of noise in the event log. We propose a method for repairing event logs by decomposing them into sub-logs and eliminating the noisy behavior within these sub-logs using covering probability. The repaired sub-logs are then integrated into the original event log at the appropriate location. Additionally, we propose a probabilistic method that considers the frequency of occurrence for activities in a given situation. This method allows for the removal of noisy and abnormal behavior from the event log, providing an overall perspective on the process. To validate our approach, we generate artificial event logs with the presence of noisy behavior using the ProM framework. By using the RapidMiner-based ProM Extension, we generate a test set to illustrate how various types of noisy behavior in an event log can be identified and repaired. Our findings clearly demonstrate that repairing the event log improves the performance of a process mining model.

**INDEX TERMS** Covering probability, enhancement, outliers, process mining, repaired event log.

## I. INTRODUCTION

Process mining is a data analysis method related to data mining in business process management. It combines knowledge from information systems and management to improve process models. The objective of process mining is to extract process-related information from an event log to enhance the model [1].

There are three types of process mining models: process discovery, conformance checking, and enhancement. Process discovery determines the actual process from the event log, conformance checking checks if the process model is consistent with the event log, and enhancement improves the process mining model based on the event logs.

Process mining methods assume that the behavior of the basic process is accurately recorded in the event log and that the log captures the complete behavior of the process. However, in reality, event data often contains noise, outliers, and infrequent behaviors. These can lead to complex and

The associate editor coordinating the review of this manuscript and approving it for publication was Dominik Strzalka.

inaccurate results in process mining algorithms. Examples of noise and infrequent behaviors in event logs include:

## A. NOISY BEHAVIOR

***i. Outliers:*** It is an incident that noticeably departs from the general trends throughout the process. These anomalies may be the result of mistakes, malfunctions in the system, or specific circumstances.

***ii. Incomplete or Incorrect Data:*** Event data that is erroneous or missing can introduce noise. This could occur because of system failures, human error in event logging, or technological problems during data gathering.

***iii. Duplicate Events:*** Repeated or duplicate events recorded in the dataset, which could skew the results of the process flow analysis and interpretation.

## B. INFREQUENT BEHAVIOR

***i. Rare Paths:*** It is a series of occurrences that are rare in relation to many process instances. These unusual routes could be exceptional instances, departures from the standard, or different process flows.

***ii. Uncommon Activities:*** Rarely occurring actions within the process instances. These actions may point to process exceptions, edge cases, or unexpected behaviors.

***iii. Unusual Timing:*** Events that, in contrast to the conventional process flow, happen at odd timestamps or at odd intervals. These variations in timing could be an indication of process abnormalities or inefficiencies.

To address this issue, a process mining model is developed to include a pre-processing phase that aims to identify and eliminate traces containing undesirable behavior. However, this manual process is expensive and time-consuming. The presence of noisy, infrequent, and incomplete behavior negatively impacts the process mining model. As a result, researchers are working on automatically applying data cleaning techniques to enhance the model's performance. A recent study attempted to filter out traces containing outlier/noisy behavior in the event log [2], [3]. While both approaches show improvements in the process mining model, particularly in process detection, there is a risk of disregarding traces that exhibit external behavior. This can distort the normal distribution of behavioral processes and potentially lead to negative or inaccurate process mining results.

The process mining model takes event logs as input, which capture the smallest unit of data whenever an activity occurs in a process. Traces are sequences of events. Our research focuses on enhancement as one of the main types of process mining. In this study, we specifically utilize enhancement to improve the overall performance of the process mining model, with a focus on the control-flow perspective. As the term 'enhance' implies, it aims to improve the quality and understandability of the process model. There are two main types of enhancement: Repair and Extension. In our research, we utilize a repair technique to enhance the model's performance and propose a method for repairing event logs that contain outlier/noisy behavior.

The key contributions of our study are as follows.

i. In our research work, we first identify the outlier/noisy behavior in the event log and then repair them to attain more accurate results of the process mining model.
ii. In our study, outlier/noisy behavior is detected using a probabilistic method according to the process context, i.e., a small part of a sequence of activities that happen before and after the outlier/noisy behavior.
iii. After their detection, the corresponding behavior is exchanged with other fragments that are more likely to occur within the context where the outlier/noisy behavior occurs.
iv. We use a RapidMiner-based ProM extension [4], called RapidMiner [5]. The accuracy of our proposed method lies in identifying the outlier/noisy behavior in the event log and then repairing them to improve the overall quality and understandability of the process mining model.
v. Furthermore, we demonstrate that our proposed method yields more accurate results than the filtering technique in the process mining domain.

Our paper is organized as follows: related work is discussed in section II. Literature Review, Preliminaries, and the proposed technique are presented in section III. In section IV, we evaluate the details and discuss the data source and data processing. In conclusion, section V summarizes our paper.

## II. RELATED STUDY

Many process mining algorithms address outlier and noisy behavior [6], [7]. However, these methods often focus on specific types of noise, such as incompleteness, and may not be suitable for general event log cleaning. Our study proposes a comprehensive repair method that considers various types of noisy behaviors.

In ProM [8], there are basic filtering plug-ins that depend on activity frequencies and user input. The discovery of external temporary data is considered in many studies, such as [9], where research on various techniques for detecting outlier and noisy behavior in sequential data is discussed. These studies also discuss related methods that are specifically designed for the process mining domain. Researchers [10] and [11] suggest filtering methods that use additional information like training event data or a reference process model. However, it is not always possible to have a set of training events or a reference process model. Recently, many researchers in the field of in-process mining have proposed common objectives for filtering strategies.

In [3], an Anomaly Free Automaton (AFA) is built that considers the entire event log and gives limited value to non-filtering behavior with respect to AFA. In [4], a research technique is proposed that detects outliers based on conditional probabilities and sub-sequence potential activities. In [12], a configurable online filtering method is proposed that identifies outlier behavior in streaming event logs using conditional probability. All of the methods mentioned above focus on eliminating outlier and noisy behavior. However, repairing the outlier and noisy behavior in the event log is more valuable than simply removing it.

In some studies [13] and [14], researchers propose repair process models based on event logs. Similarly, in [15], a researcher uses a process model to repair an event log. In our study, we aim to design a repair technique, so we assume that there is no process model available for the implementation of our proposed algorithm.

## III. PRELIMINARIES

### A. NOTATIONS

Let $x$ be a set, and $m$ a multi-set over $x$ defined as a function $m : x \rightarrow n_{\geq 0}$ which count multiplicity of each element in $x$. Therefore, we describe a multi-set $m$ like $m = \{e_1^{k_1}, e_2^{k_2}, \ldots\ldots, e_n^{k_n}\}$, where $1 \leq j \leq n$ and having $m(e_j) = k_j$ with $k_j \in n_{\geq 0}$. If $k_j = 1$, then remove the superscript, and if for some $e \in x$ then $m(e) = 0$, and eliminate it from the multi-set notation. Whereas $m = [\emptyset]$ represents an empty multi-set, as $\forall_{e \in x}$, $m(e) = 0$. Hence by taking $\bar{m} = \{e \in x / m(e) > 0\}$, whereas $\bar{m} \subseteq x$. In the end, set of all possible multi-set of set $x$ are represented as m($x$).

By assuming $x^*$ as a set of all sequences of set $x$. Whereas a finite sequence $\delta$ with length $n$ over set $x$ is a function of $\delta = \{1, \ldots, n\} \rightarrow x$, alternatively describe as $\delta = \{x_1, x_2, \ldots\ldots, x_n\}$ where $x_j = \delta(j)$ for all $1 \leq j \leq n$. We can write the empty sequence as $\in$. Multiplication of sequences $\delta$ *and* $\delta'$ is written as $\delta.\delta'$. Having a function $hd : x^* \times n_{\geq 0} x^*$ representing the "head" of a sequence, as we have a sequence $\delta \epsilon x^*$ and $k \leq |\delta|$, then $hd(\delta, r) = (x_1, x_2, \ldots\ldots, x_k)$ with the sequence of first $r$ elements of $\delta$. At the point when $r = 0$ then we have $hd(\delta, 0) = \in$ which is symmetrical, whereas, $tl : x^* \times n_{\geq 0} x^*$ denotes the "tail" of a sequence and is represented as $tl(\delta, k) = (x_{n-k+1}, \ldots\ldots, x_n)$ with the sequence of last $r$ elements of $\delta$. Similarly, when we have $k = 0$ then $tl(\delta, 0) = \in$. Whereas Sequence $\delta'$ is a subsequence of sequence $\delta$, by denoting as $\delta \in \delta'$, *if and only if* $\exists_{\delta_1 \delta_2 \in x^*}$ $(\delta = \delta_1.\delta'.\delta_2)$. Therefore, by assuming $\delta, \delta' \in x^*$, we define the occurrence as $\delta' in \delta$ by $freq : x^* \times x^* \rightarrow n_{\geq 0}$ where $freq(\delta', \delta) = \left| \{1 \leq j \leq |\delta| / \delta'_1 = \delta_i, \ldots\ldots.\delta'_{|\delta'|} = \delta_{j+|\delta'|}\} \right|$. Taking an example event log presented in table 3 $freq((b), (a, b, b, c, d, e, f, h)) = 2$ *and* $freq((b, d), (a, b, d, c, e, g)) = 1$ so on.

### B. EVENT LOG

Many studies have shown that event logs are a basic requirement for process mining. According to Aalst [2], an event log has data that relate to a single process with some assumptions, such as: (i) cases exist in a process, (ii) every case has an event, i.e., each event is linked with the previous case, and (iii) events are in order within cases. In our study, let's assume that we have a set of activities represented as $\mathcal{A}$. As we know, event logs are the multi-set of sequence over $\mathcal{A}$, *i.e.*, $l \epsilon \mathrm{M}(\mathcal{A}^*)$. By considering each $\delta \epsilon \bar{l}$ we have a trace variant. Whereas $l(\delta)$ define as traces of $\delta$ which are existing inside the event log.

***i. Petri Net:*** It is a mathematical model used to describe the distributed systems. Here in this model the entities called places are represented with states and events are represented by transitions. Places and transitions are connected by directed arcs, which indicate the flow of the tokens. It is basically used to analyze system behavior, deadlock etc.

***ii. Alpha Miner***: The identification of a process model is linked to event logs or observable data via the Alpha Miner (also known as the $\alpha$-algorithm, $\alpha$-miner). The data source for the alpha miner algorithm is event logs. It begins by converting the event logs into sequence, choice, parallel, and direct-follows relations. A petri net describing the process model is then constructed utilizing these connections. Put simply, it produces a visualizable, timestamped flow of business processes.

***iii. Inductive Miner:***Another popular process mining approach for identifying process models from event logs is the Inductive Miner. This method is based on the idea of splitting event logs into smaller sub-logs, also known as splits or cuts, and then identifying different cuts on the graph that is directly formed from the event logs. The versatility and scalability of the Inductive Miner is its key advantages. The approach of finding different divisions in the directly-follows graph and using the smaller components after division to indicate the activities' execution order is what makes Inductive Miners distinctive. By iteratively exploring the space of potential process models, the Inductive Miner algorithm may identify a variety of process architectures, from simple linear models to more intricate models with concurrency loops.

### C. PROPOSED PROCESS MINING MODEL: CONTROL FLOW REPAIRING EVENT LOGS

The methodology adopted for repairing the event logs shows that we first identify the outlier/noisy behavior, and then we repair the event log. Our study comprises two central control flow-oriented concepts: firstly, repairing the event log and then covering traces in the context surrounding the behavior of the sequence activities. Traces and context covering represent the surrounding behavior of some sequence activities.

### D. CONTEXT COVERING

Let $\delta, \delta' \epsilon x^*$ we define the context of $\delta'$ concerning $\delta$ as a function $con : x^* \times x^* \times n_{\geq 0} \rightarrow \mathcal{P}(x^* \times x^*)$ where $con(\delta', \delta, l, q) = \{(\delta_1, \delta_2) \in x^* \times x^* / \delta_1.\delta'.\delta_2 \epsilon \delta \bigwedge |\delta_1| = l \bigwedge |\delta_2| = q$ Furthermore, let $\delta'_1, \delta'_2 \epsilon x^*$, $cov : x^* \times x^* \times x^* :\rightarrow \mathcal{P}(x^*)$ are the function returns all subsequences to the traces happened in a given context, i.e., $cov(\delta, \delta'_1, \delta'_2) = \{\delta' \epsilon x^* \left| (\delta'_1, \delta'_2) \epsilon con(\delta', \delta \left| \delta'_1 \right|, \left| \delta'_2 \right|) \right|$.

Context covering distribute the given subsequences into their two neighboring subsequences with length $l$ and $q$ respectively, like $\delta'_1$ represent their left neighbor with length $l$ whereas, $\delta'_2$ represent right neighbor with length $q$. By taking an example, if $\delta = (a, b, d, e, f, g)$ we have $con((e), \delta, 1, 2) = ((b), (d, e))$. Remember that $l$ and $q$ may differ, with '0' value which shows a nearby subsequences $\epsilon$. Moreover, $cov(\delta, (a, b, d), (e, f, g) = \{c, h$ and

$cov(\delta(b), (d)) = \{(c)\}$. Hence our main purpose is to find out and replace the outlier/noisy behavior which depends on probability of occurrence for a sub-sequence in the specific context. In a case, if the proposed probability is lower than the limited value, then we take the behavior as outlier/noisy, to get their probabilities we describe the covering probability in the following section.

## IV. COVERING PROBABILITY

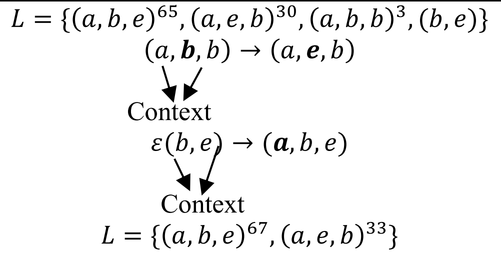
Let $\delta, \delta'_1, \delta'_2 \in x^*$, maximum sub-sequences with length $K$ and a multi-set of sequences $l \in \mathrm{M}(x^*)$. We define $c.p : x^* \times x^* \times x^* \times n_{\geq 0} \times \mathrm{M}(x^*) \rightarrow [0, 1]$ as an empirical conditional probability $\delta'$ covered by $\delta_1 and \delta_2$ in event log $l$, like:

$$c.p\left(\delta', \delta'_1, \delta'_2, K, l\right) = \begin{cases} \dfrac{\sum_{\delta=1} (l(\delta) frq(\delta'_1, \delta', \delta'_2, \delta)}{\sum_{\delta=1} (l(\delta) \sum_{|\delta''| \leq K} frq(\delta'_1, \delta'', \delta'_2, \delta)} \\ \quad if\ \exists \delta \epsilon l(cov(\delta', \delta'_1, \delta'_2) \neq \emptyset \\ 0 \quad otherwise \end{cases}$$

The numerator of the covering probability is the number of times the context $(\delta'_1, \delta'_2)$ covers various substrings inside the lower length that equals $K$ and the denominator is the number of times the context $(\delta'_1, \delta'_2)$, is encountered across the entire event log. It is evident that the value obtained are real numbers within the range [0, 1]. A subsequence $\delta'$ among the range context $(\delta'_1, \delta'_2)$ is more likely to occur if the value is higher. As a result, $c.p\left(\delta', \delta'_1, \delta'_2, K, 1, l\right) = 1$ means that if context $(\delta'_1, \delta'_2)$ exist, then sub-sequence $\delta'$ also occur frequently among them. According to the event log shown in table 1, $CP(\epsilon, (b), (c), 1, l) = \frac{7}{12}$ *and* $CP((b), (b), (c), 1, l) = \frac{1}{12}$.

**TABLE 1. Significant contexts and their probable subsequence for K=1.**

| Significant Context | Frequency | Probability Subsequences |
|---|---|---|
| $(\emptyset, (a))$ | 98 | $\emptyset$ |
| $(\emptyset, (b))$ | 70 | $(a)$ |
| $(\emptyset, (e))$ | 30 | $(a)$ |
| $((a), (b))$ | 101 | $(\emptyset, (e))$ |
| $((a), (e))$ | 95 | $(\emptyset, (b))$ |
| $((b), (e))$ | 67 | $\emptyset$ |
| $((b), \emptyset)$ | 103 | $(\emptyset, (e))$ |
| $((e), \emptyset)$ | 35 | $\emptyset$ |
| $((e), \emptyset)$ | 97 | $(\emptyset, (b))$ |


$L = \{(a,b,e)^{65}, (a,e,b)^{30}, (a,b,b)^{3}, (b,e)\}$
$(a,\mathbf{b},b) \rightarrow (a,\mathbf{e},b)$
Context
$\varepsilon(b,e) \rightarrow (\mathbf{a},b,e)$
Context
$L = \{(a,b,e)^{67}, (a,e,b)^{33}\}$

The goal of our study is, firstly, to consider the covering probability of sub-sequences within the noisy/outlier behavior in an event log. Whereas their significant context is a context that occurs significantly; therefore, the noisy/outlier sub-sequence is replaced with other sub-sequences having a higher covering probability between the given contexts.

In the context of our example, we take a trace $(a, b, b, e, f, g)$, by considering $((b), (e))$ as a frequent context, and then replace the subsequence $(b)$ with $\in$, that is more often to occur among the given context, after repairing we get the trace $(a, b, d, e, f, g)$, which have no outlier present in it. In another example, considered trace $\delta = (a, b, d, e, f, h)$ with the significant context $((a), (b))$, by considering the entire event log, we assume $\epsilon$ occurs among them rather than $(e)$. After their replacement, we get the repaired trace as $(a, b, d, e, f, h)$. Likewise, for $(a, c, d, e)$ by assuming context $(c, d)$ with their covering probabilities, we replace $\in$ it with $(e)$ and get $(a, b, c, d, e)$ without any outlier.

The user determines which condition is significant by setting the corresponding limit value. The context frequency with the least number of traces in an event log and their limited value is taken as a significant context. The user can also specify the maximum length of covered subsequence $(K)$ and contexts subsequence $(e_l)$ respectively. Remember that the $(c_l)$ defines two values like the maximal length for $\delta'_1, \delta'_2$.

As our proposed study is based on the repairing method of an event log '$l$', context frequency with the limit value $t_c$, context sub-sequences length $c_L(l, q)$ and their upper bound with length $n$ of covered sub-sequence $K$. Initially, all the traces are scanned to calculate their covering probabilities for contexts and possible sub-sequences. After that, we calculate their context frequency and covering probability (according to $t_c$) for all traces and sub-sequence present in it (with length $\leq K$). In a case, if context frequency is significant and their covering probability is lower than we interchange each sub-sequence with the best one based on the context frequency. Otherwise, in a situation, if we have insignificant context then it is impossible to use them for repairing noisy behavior.

Table 1 provides a basic visual demonstration of the operation of the suggested technique. To repair an event log with 100 traces, we consider $K = 1$ as the maximum subsequence length and context lengths equal to 1. Initially, important contexts and their likely subsequences are identified by scanning the event log. We identify a noisy/infrequent behavior if the related context is significant, and the subsequence is not likely for that context. For example, in the significant context $(\langle a \rangle, \langle b \rangle)$, the occurrence of $\langle a \rangle$ is unlikely.

Repairing Event Logs 9, identifying anomalous behavior, our goal is to substitute unlikely subsequences with more likely ones. As we are looking for a subsequence that is as close as possible in length to the outlier subsequence for substitution. The subsequence with the highest probability among all the candidate subsequences is the one we are interested in. We first look for a subsequence with the same length, for instance, if the outlier subsequence has length 2. We try for a subsequence $\delta''$ with length 1 or 3 if there isn't a meaningful subsequence with length 2 for that context. Next, we select the candidate subsequence with the highest probability in that situation. Table 1 shows that there are two subsequences that can be substituted with $\langle b\rangle$: i.e. $\varepsilon$ and $\langle c\rangle$. We select $\langle c\rangle$ because its length is comparable to the outlier subsequence, and the trace is modified to $\langle a, c, b\rangle$. It is more likely that a occurs for the other outlier trace in this case, among the context $(\varepsilon, \langle b\rangle)$. It becomes $\langle a, b, c\rangle$. when $\langle a\rangle$ is substituted. The initial point of the previous context's right subsequence will serve as the beginning point of the subsequent scanning subsequence following each replacement to prevent infinitive loops. For instance, we won't verify subsequence $\langle a\rangle$ again after by replacing $\varepsilon$ *with* $\langle a\rangle$ in $\langle a, b, c\rangle$ and considering $\langle b\rangle$ to be the next subsequence.

After detecting the noisy behavior, we try to replace the impossible sequence with the most likely ones. For substitution, we are looking for a sequence that is as close as possible to the next outlier our interest is towards the highest probability. To avoid endless loops, after every change, we first scanned the sequence with the right sequence of the previous context.

### A. EVALUATION OF THE PROCESS MINING MODEL: REPAIRING NOISY/OUTLIER BEHAVIOR

Here, we discuss the important steps with their concise algorithmic explanation of our control flow repairing event logs by using context and covering probability in our study we focus on the behavioral context to lower the computational complexity with their sub-sequence maximum length that is '$K$' *and* '$l$'. By defining the behavioral context, we have.

$$\mathrm{B}_L^{K,l} = \{(\delta_K, \delta_l)\,\epsilon\,\mathrm{B}_L\,|1 \leq |\,\delta_l| \leq l \wedge 1 \leq |\delta_K| \leq K$$

By keeping in mind that, do not take the context length of sub-sequence '0'

Furthermore, we restrict the maximum length of the sub-sequence to the limited value $p_l$. After that, their relative behavioral context frequency becomes as.

$$f_{B_L}^{K,l}(\delta_r, \delta_l) = \frac{\sum_{\delta\epsilon\bar{L}}\left(L(\delta) \times \sum_{\delta'\in A^*,\delta'\leq p_l}|\delta_{\delta_K,\delta',\delta_l}|\right)}{|L|}$$

whereas $\delta_K$, $\delta_l$, *and*$B_L$ are equals to $\mathrm{B}_L^{r,l}$. Similarly, empirical conditional covering probability is defined as.

$$c.p\left(\delta', \delta_1', \delta_2', K, l\right) = \begin{cases} \dfrac{\sum_{\delta=1}(l\,(\delta)\,frq(\delta_1', \delta', \delta_2', \delta)}{\sum_{\delta=1}(l\,(\delta)\sum_{|\delta''|\leq K} frq(\delta_1', \delta'', \delta_2', \delta)} \\ \quad if\ \exists\delta\epsilon l(cov(\delta', \delta_1', \delta_2') \neq \emptyset \\ 0 \quad otherwise \end{cases}$$

For every trace, we begin from a sub-sequence length 0. Our main focus is to initiate from the longest one that is equal to '1'. As we know, if context $\left(\delta_{l_n}, \delta_{l_{n-1}}, \ldots\ldots., \delta_{l_1}\right), \left(\delta_{k_1}, \ldots\ldots., \delta_{k_n}\right)$ is frequent, then the $\left(\delta_{l_n-1}, \ldots\ldots., \delta_{l_n}\right), \left(\delta_{r_1}, \ldots\ldots., \delta_{r_{n-1}}\right)$ is also frequent. Our interest is in the longer context because they are more interesting. On-Line 17th of algorithm 1, if the context is insignificant w.r.t $t_c$ and the empirical conditional covering probability of sub-sequence $\delta'$ is not at their highest-level w.r.t $t_{c.p}$, so, replace them with suitable sub-sequences. Remember that, with every replacement, their index is high which makes shower the end of the algorithm.

We cannot take '$K$' *and* '$l$' equal to 0. To find out the noisy/outlier behavior at the beginning and ending of the trace, therefore, in every trace for an event log, we add an artificially generating event.

**Algorithm 1** Repairing Noisy/Outlier Behavior in an Event Log

```
1.  Produce REPAIR (L,p_l, r, l,t_{c.p}, t_c)
2.  L' ← [] // empty multi-set
3.  For each (δ ∈ L) do
4.  Adding artificially generating beginning and terminating activities δ
5.  for(i ← 0 top_l)do //sub-sequence
6.  for (j ← lto 1)do // left context
7.  for (k ← rto 1)do // right context
    end for
8.  ind← 0
9.  (i + j + k + ind ≤ |δ|) then //context+sub-sequence part of tracesδ
10. δ_l ← (δ_ind, ........, δ_ind+j)
11. δ' ← δ_ind+j+i, ........, δ_ind+j+i)
12. δ_r ← (δ_ind+j+i+1, ........, δ_ind+j+i+k)
13. if f_{B_L}^{r,l} (δ_r, δ_l) ≥ t_c ∧ γ_L^{p_l}(δ', δ_r, δ_l) ≥ t_{c.p} then
14. Replace(δ', (δ_r, δ_l))
    end if
15. δ'' ← replacement acc. strategy
16. Replaceδ' andδ'' inδ
17. ind← ind + |δ''
18. ind← ind+1
19. Repaideventlog← add
20. Return
```

In algorithm 1 we have the pseudo-code for our proposed repair method. Event log like '$L$' start with the input of our proposed method, sub-sequence with maximum length ($p_l \in z_{\geq 0}$), the maximum length of the right and left sub-sequence context ($n \in K, n \in l$), least limited values length for the relative behavioral context ($t_c \in z_{>0}$), empirical conditional covering probability ($0 \leq t_{c.p} \leq 1$), and $L'$ returns the repaired event log.

Real event logs mostly contain noisy/outlier behavior. Such behavior can result in inaccurate process mining algorithms, which can reduce the accuracy of our proposed process mining model. Enhancement aims to improve the quality, value, desirability, and attractiveness of a process mining model. Therefore, repairing event logs from noisy behavior is essential. Repairing an event log in a process mining model is a main challenge in process discovery [7].

Let's consider an example of a hospital, where the event logs contain noisy behavior, defining sequences of executed business process activities, usually within the context of cases, for example, a patient admitted to a hospital or case ID.

Performance in that case context is referred to as an event, and the sequence of specific cases refers to traces. It is possible to define the same sequence of activities for multiple traces, but for that, each event should be unique, whereas the traces themselves have different events.

Hence, the receptionist identifies the patient, and after their identification creates a new record, the clinician admits the patient and takes vitals, by giving treatment to the patient. After the physician evaluates the patient and gives no lab/procedure, diagnosis of the disease, and give treatment plan at the end accountant gives the patient billing and discharge slip. Let's suppose $(a, c, b, d, e, f, g, h)$ which are the shorthand activity names. In our study, formally we define an event log as a multi-set of sequences for the activities. In tables: 2 and [tab4]3, we take the example of event logs with noisy behavior and their resources.

**TABLE 2.** List of activities and resources.

| Letter | Activity | Resources |
|---|---|---|
| a | create New Patient Record | Receptionist |
| b | Admit | Clinician |
| c | Take Vitals | Clinician |
| d | Treatment | Clinician |
| e | No lab/Procedure | Physician |
| f | Diagnosis | Physician |
| g | Treatment Plan | Physician |
| h | Billing and Discharge | Accountant |

**TABLE 3.** Trace in the event log.

| Case ID | Trace | Frequency |
|---|---|---|
| 1 | (a, b, c, d, e, f, h) | 2 |
| 2 | (a, b, d, e, f, c, h) | 1 |
| 3 | (a, c, d, e) | 1 |
| 4 | (a, b, d, e, f, g) | 1 |
| 5 | (a, b, d, e, f, h) | 1 |
| 6 | (a, b, d, e) | 1 |
| 7 | (a, c, d, e) | 1 |
| 8 | (a, b, b, c, d, e, f, g) | 1 |
| 9 | (a, b, e, f, h) | 1 |
| 10 | (a, c, d, b, e, f, a, h) | 1 |

For example, consider an event log that relates an event to case ID value 1. Hence, the receptionist identifies the patient, and after identification, a new record is created. The clinician admits the patient and takes vitals, giving treatment to the patient. After the physician evaluates the patient and provides no lab/procedure, a diagnosis of the disease is made, and a treatment plan is provided. Finally, the accountant provides the patient billing and a discharge slip. Let us suppose (a, c, b, d, e, f, g, h) are shorthand activity names.

In our study, formally, an event log is defined as a multi-set of sequences for the activities. In tables 1 and 2, examples of event logs with noisy behavior and their resources are provided.

Here, 74 events are considered, linked with 11 traces. Each trace occurs once except the first one, which occurs twice. As shown in Table 3, the first three traces have no noisy/outlier behavior. However, in the 4th and 5th traces, activities 'g' and 'h' are missing. The remaining seven traces, including the first three, exhibit several types of noisy/outlier behavior. In process discovery, the alpha miner [9] is more sensitive to noisy/outlier behavior, resulting in an inaccurate process mining model. On the other hand, the inductive miner [10] implements a built-in filtering technique to oversee such behaviors.

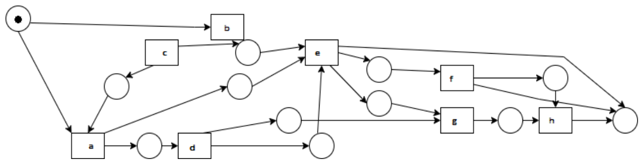


**FIGURE 1.** Alpha miner with noisy/outlier behavior.

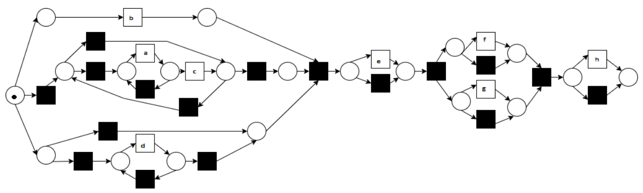


**FIGURE 2.** Inductive miner with noisy/outlier behavior.

In figure 1, apply Alpha Miner with noisy /outlier behavior and in Figure 2, we apply the inductive miner with noisy behavior. The black box represents the immediate transition, and the white box represents the transition after the immediate transition. Circles represent the places, with a dot inside representing a token.

In this case, we first repair the event log and then apply the alpha miner and inductive miner. The black box shows the immediate transitions, and the white box shows the transitions after that. We attain an accurate and understandable process mining model, as shown below.

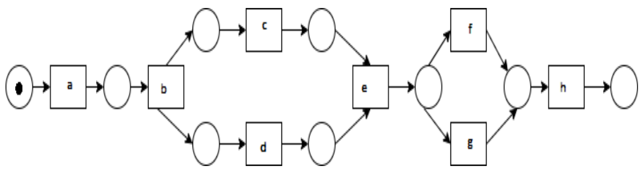


**FIGURE 3.** Alpha miner after repairing event log.

In Figure 3, after repairing the event log, we apply the alpha miner. Here, circles represent the places with a token inside, and the box represents the transition.

In Figure 4, we apply the inductive miner after repairing the event log. The black box represents the immediate transition, and the white box represents the flow of cases in the process. Hence, we realize that instead of removing the event log with

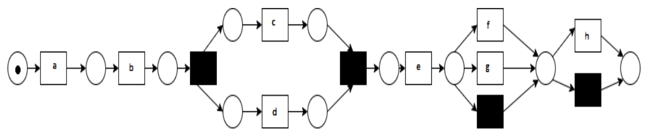


FIGURE 4. Inductive miner after repairing event log.

noisy/outlier behavior from the dataset, we repair them to attain accurate results. The black box shows the immediate transitions, and the white box shows the timed transitions. Because event logs have significance in the process mining algorithm, rather than removing them, we choose to repair them. This enhances the performance and quality of the process mining model.

### B. SIMULATION SETUP

#### 1) DATA SOURCES

To validate our proposed method, we conducted experiments using the ProM framework on artificially generated datasets. We created 10 distinct business process models and generated 100 log traces for each model. We introduced various types of noise to assess the robustness of our repair technique.

The user begins by customizing a series of parameters to generate a process model. These parameters determine the size of the process model and the probability of different structures appearing in the model (e.g., sequence, choice, parallel, loop). These parameters ensure the creation of an authentic business process model that accurately reflects real-world scenarios.

For our experiment, we randomly generated 10 different business process models. Once we obtained a process model, we used the matching plug-in to generate 100 log traces for each model. It is important to note that, at this stage, the event log precisely conforms to the process model. We used these conformance event logs to calculate the denominator of the ‘covering probability.’ The workflow of the artificial data generation algorithm is illustrated in Figure 5.

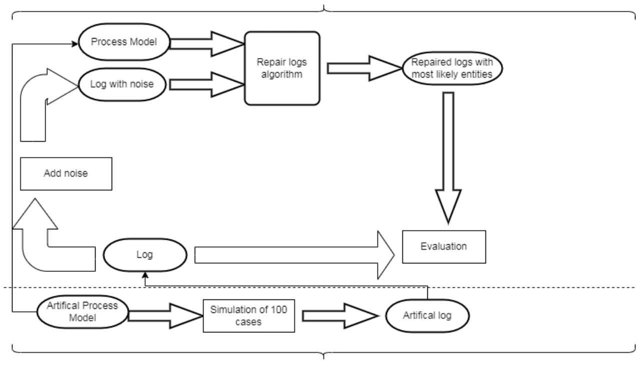


FIGURE 5. Artificial data generation algorithm.

## V. RESULT AND DISCUSION

We conducted experiments using real and artificial event data to assess the proposed method’s efficacy. We focused only on context subsequences with a length of **1**, simplifying our process. In the ProM-framework, we used the Repair Log plug-in (***RL***) as the repair method. This plug-in takes an event log as input and produces a repaired event log as output. Additionally, we selected the context ***CLs*** with left and right sequence lengths, determined the maximum subsequence length (***K***), and set the threshold (***TC***). We also used the Repair Log plug-in in RapidProM, implementing our proposed method across multiple event logs with varying thresholds and process mining algorithms with different configurations. RapidProM is a RapidMiner module that integrates various process mining algorithms with scientific workflows.

### A. EVALUATION MEASURES

Here, we implement the algorithms used in our research work by using PM4py and ProM Framework which is evaluated by the following perspectives. The Fitness, Precision, Accuracy and F-Measure are used to compare the performance of each classifier. Because of the data set’s asymmetry, overall accuracy may be misleading.

i. ***Fitness::*** Assume that $d$ is the total number of distinct traces found in the combined log. For every log $j(1 \le j \le d)$, the number of process instances that make up the current trace is denoted by $n_j$; the number of missing tokens is indicated by $m_j$; the number of tokens that remain is indicated by $r_j$; the number of consumed tokens is indicated by $c_j$; and the number of tokens produced during log replay of the current trace is indicated by $p_j$. The following is the definition of the token-based fitness metric:

$$fitness = \frac{1}{2}\left(1 - \frac{\sum_{j=1}^{d} n_j m_j}{\sum_{j=1}^{k} n_j c_j}\right) + \frac{1}{2}\left(1 - \frac{\sum_{j=1}^{d} n_j r_j}{\sum_{i=1}^{k} n_j p_j}\right)$$

whereas, for all $j$, $m_j \le c_j$ and $r_j \le p_j$, therefore $0 \le fitness \le 1$.

ii. ***Precision***: Let $d$ represent how many distinct traces there are in the combined log. Note that invisible tasks may enable successive labeled tasks, but they are not counted themselves. For each log trace $j(1 \le j \le d)$, $n_j$ is the number of process instances integrated into the current trace, and $y_j$ is the mean number of enabled transitions during log replay of the current trace. Additionally, the collection of visible tasks in the Petri net model is denoted by $t_v$. Here is how the precision metric is defined:

$$precision = \frac{\sum_{j=1}^{d} n_i \left(|t_v| - y_i\right)}{\left(|t_v| - y_i\right) . \sum_{j=1}^{d} n_i}$$

iii. ***Accuracy:*** It’s the proportion of accurately predicted samples relative to the total samples within the dataset is called accuracy. It can be calculated as:

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

We used fitness and precision to assess the discovered process models. Fitness calculates the percentage of event log behavior that a process model also describes. Precision, on the other hand, estimates how much of the behavior that a model predicts is recorded in an event log. Low precision shows that a process model, compared to an event log, is more significant. The tradeoff between these measures should be noted. Sometimes, excluding a small portion of behavior results in an insignificant reduction in fitness, though precision increases significantly. Consequently, the F-Measures metric was employed, which combines fitness and precision, to assess the identified process models.

$$\frac{2 \times \boldsymbol{Precision} \times \boldsymbol{Fitness}}{\boldsymbol{Precision} + \boldsymbol{Fitness}}$$

Note that fitness often plays a significant role in many applications. As a result, the concept of conditional precision is additionally employed, wherein only the precision values of process models with fitness values lower than 0.95 are considered. The Matrix Filter also performs well on event logs with noise, which is another positive aspect.

**TABLE 4. Real event logs use in our study.**

| Event Log | Activities Count | Traces Count | Event Count | Variants Count |
|---|---|---|---|---|
| offer_BPIC_2017 | 13 | 13087 | 561470 | 17 |
| 2013_BPIC | 8 | 9658 | 60849 | 2263 |
| application_BPIC_2012 | 11 | 42995 | 65533 | 16 |
| Workflow_BPIC_2012 | 7 | 5015 | 72413 | 2278 |
| Offer_BPIC_2012 | 6 | 7554 | 31244 | 231 |
| raod_fines | 10 | 1000000 | 452359 | 168 |
| hospital_biling | 15 | 150370 | 193849 | 846 |
| sepsis_case | 18 | 10035 | 150525 | 1 |
| credit | 16 | 1050 | 15224 | 1020 |

In our experiment, we aimed to evaluate the performance of our proposed approach on real event logs. Table 4 provides the necessary details of the event logs used in our experiments. We also introduced varying amounts of noise into these event logs, which involved randomly adding, removing, and switching activities within the traces. For example, '$Road - Fitness - 0.5$' was created by adding 5% of each category of noise mentioned earlier. It is important to note that modified, filtered, or repaired event logs were used to discover process models for all tests. Real event logs without noise were used to assess conformity and evaluate the quality of the resulting process model.

Four different methods were employed to identify the best process models for these event logs. The first method used was $N\&IMi$, where N represents an event log with modifications, and $IMi$ [7] was applied to 51 different types of noise filtering thresholds ranging from $0 to 1$. The second method employed was called $M\&IM$, where $M$ represents the event log repaired with the filtered matrix. The inductive miner was then implemented on the event log that had already been repaired using the filtered matrix method. On event logs that were already repaired using our proposed method, the fundamental Inductive Miner was also applied, referred to as $R\&IM$. Finally, the repaired event logs ($R\&IMi$) were subjected to the inductive miner and four different types of noise filtering thresholds (0.1 *to* 0.4).

Figures 6 and 7 illustrate the results obtained from applying these proposed methods to the event logs and their corresponding F-Measure. This data indicates that the Inductive Miner ($N\&IMi$) with noisy behavior does not yield a suitable process model for the sixty-five event logs. On the other hand, the inductive miner finds a process model with a suitable F-measure that does not exhibit noisy behavior. It can be observed that $R\&IM$ produces better results than $M\&IM$ for most of the event logs. The output of $M\&IM$ at the beginning of the experiment is relatively good for the hospital billing event log due to the presence of numerous variants. However, only 1% of these variants account for 94% of the traces, making this type of event log ideal for filtering. To achieve the best result, the inductive miner or $M\&IM$ can be applied to the cleaned event logs. Additionally, $M\&IM$ and the Inductive Miner typically sacrifice a significant amount of fitness to produce the best possible process model according to the $F - Measure$. Figures 8 and 9 present the results. The aforementioned results demonstrate that the $R\&IM$ methods, in combination with the inductive miner, yield the best process model with high precision without considering fitness. However, as observed, the Inductive Miner is unable to find a process model with high fitness and precision simultaneously for event logs with substantial noisy behavior.

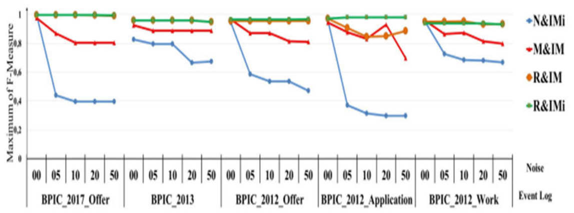


**FIGURE 6. BPIC event logs F- measure after implementing different techniques.**

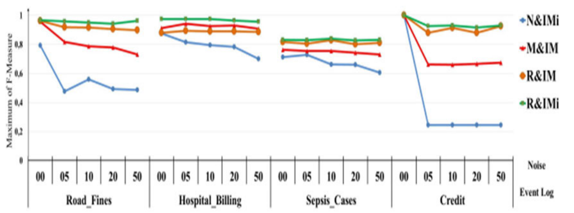


**FIGURE 7. Real event logs F-measures after implementing different techniques.**

In figures 10, we have best discovered process models on BPIC using RL, MF and NF methods in 10(a), 10(b), and 10(c) with their fitness, precision and F-measure results. Whereas, figure 12 and 11 shows synthetic event log F-measures and conditional precision after applying different methods on it. Finally, it should be noted that achieving the best results often involves removing a significant amount of

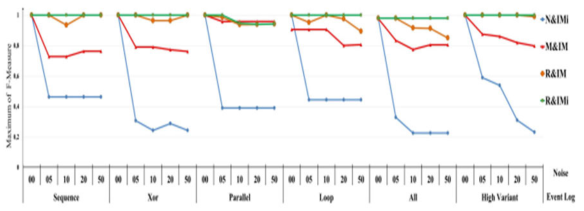


**FIGURE 8.** Conditional precision BPIC event after implementing various methods.

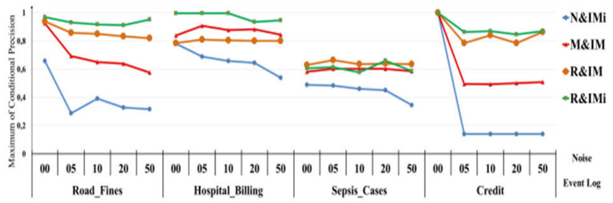


**FIGURE 9.** Conditional precision for real event logs after implementing different techniques.

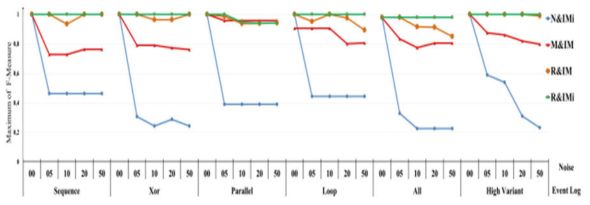


**FIGURE 11.** F-measure for synthetic event logs after implementing methods.

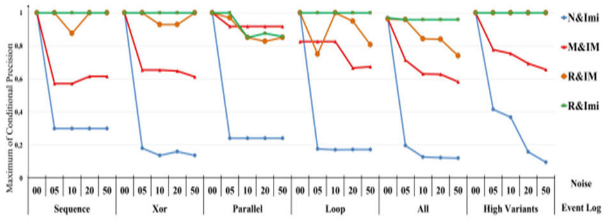


**FIGURE 12.** Synthetic event logs conditional precision after applying different methods.

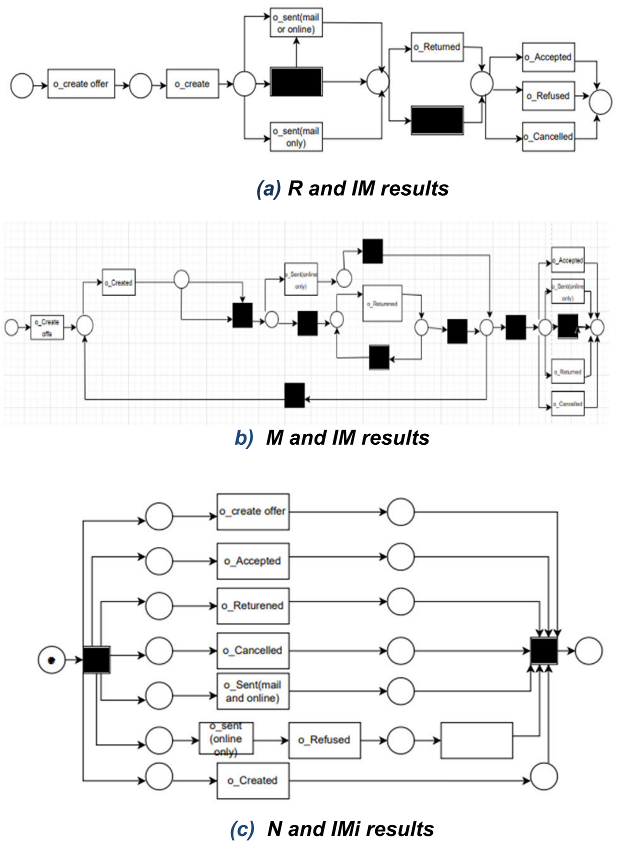


**FIGURE 10.** The best discovered process models on BPIC using RL, MF and NF methods in 10(a), (b)and (c) with their fitness, precision and f-measure results.

event log behavior using filtering techniques. A list of trace percentages is still present in each event log for the top M&IM model. To attain the best process model, sometimes we need to use 5% of the traces. Therefore, a substantial amount of behavior needs to be deleted from the event log. In the repair method, all traces are still present in an event log, although they can be altered. It is important to note that in grid search on various parameters, all methods displayed the best outcomes. Adjusting these thresholds, like with other innovative process mining-specific data cleansing techniques, poses a challenge for users.

**TABLE 5.** R and IM results.

| | |
|---|---|
| Fitness | O.999 |
| Precision | 0.994 |
| F-Measure | 0.996 |

**TABLE 6.** M and IM results.

| | |
|---|---|
| Fitness | O.989 |
| Precision | 0.776 |
| F-Measure | 0.876 |

**TABLE 7.** N and IMI results.

| | |
|---|---|
| Fitness | 1.000 |
| Precision | 0.282 |
| F-Measure | 0.444 |

After obtaining the business process model and event logs, we can simulate non-conformance in the event log from the real world by extracting 10% of the event log from the full conformance event log and adding noise to it is shown in figure 13. In our study's experiments, types of noise addition included missing, dislocated, and redundant logs. The proportion of noise addition increased from 30% to 40% and then 50% which is shown in figure [DCLfiglabel14]14 to 17, resulting in 500 log traces with noise. Subsequently, based on the method proposed in our paper, the event log after noise addition was repaired, and the conformance log before noise

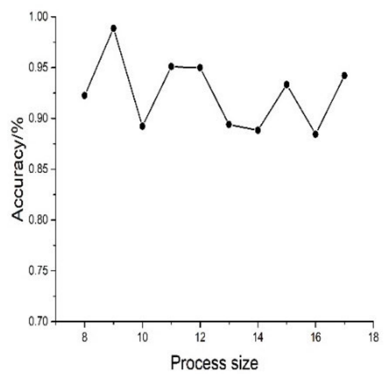


**FIGURE 13.** Accuracy at 10% noise.

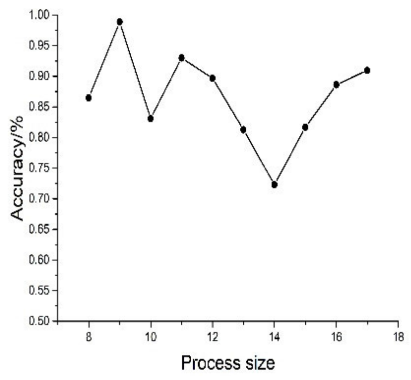


**FIGURE 14.** Accuracy at 20% noise.

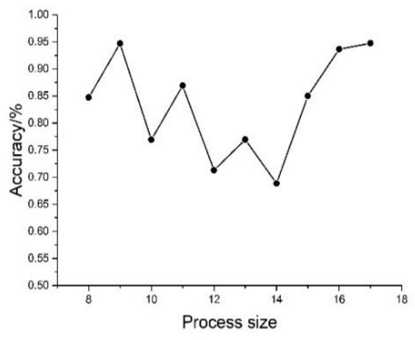


**FIGURE 15.** Accuracy at 30% noise.

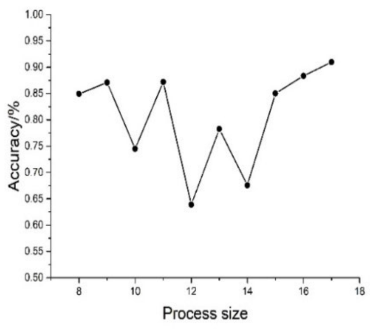


**FIGURE 16.** Accuracy at 40% noise.

addition was used as the standard to measure the repair result. In other words, if two event logs were the same, the repair accuracy was considered 100%. When the two event, logs were not the same, the 'difflib' module in Python was utilized

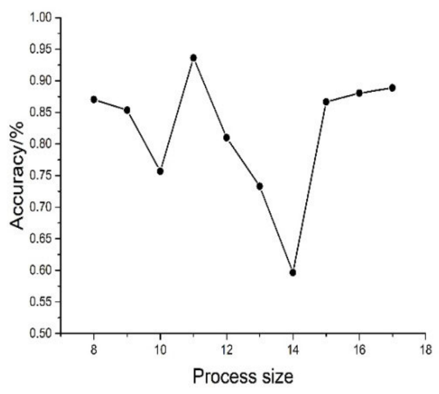


**FIGURE 17.** Accuracy at 50% noise.

to automatically calculate the similarity between them. The classes and methods provided by this module were used to compare different sequences. It could also compare files and generate different results. Finally, these similarities were used as accuracy measures for the repair results.

## VI. CONCLUSION

Process mining models are designed to operate with clean event logs. However, event logs often contain outliers and noisy behavior, which can lead to inaccurate process mining outcomes. By addressing this issue and correcting the noisy behavior in the event log, we can improve the performance of the process mining model. In our study, we investigate a process mining model that handles noisy behavior in event logs. This is done by decomposing the event logs into sub-logs and using the covering probability to eliminate the noisy behavior in each sub-log. Once repaired, the sub-logs are reintegrated into the original event log at their appropriate positions. Additionally, we propose a probabilistic method that relies on the frequency of activity occurrences in specific situations. This approach allows us to eliminate noisy and abnormal behavior from the event log, providing a comprehensive view of the process.

To evaluate the effectiveness of our proposed repair technique, we generate an artificial event log with simulated noisy behavior using the ProM framework. Through this application, we create a test set to demonstrate that our method can identify and repair various types of noisy and outlier behavior in event logs. In conclusion, our proposed method shows that repairing event logs can greatly enhance the performance of process mining models. Future research could explore the application of this method to different types of processes and further refine the probabilistic detection algorithm to handle more complex noisy behaviors.

## DISCLOSUER STATEMENT

No potential conflict of interest was reported by the authors.

## REFERENCES

[1] W. M. P. van der Aalst, "Using process mining to bridge the gap between BI and BPM," *Computer*, vol. 44, no. 12, pp. 77–80, Dec. 2011.

[2] W. M. P. van der Aalst, *Process Mining—Data Science in Action*. Berlin, Germany: Springer, 2016.

[3] R. Conforti, M. L. Rosa, and A. H. M. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Feb. 2017.

[4] M. F. Sani, S. J. Zelst, and W. M. van der Aalst, "Improving process discovery results by filtering outliers using conditional behavioural probabilities," in *Proc. Bus. Process Manag. Workshops*, 2017.

[5] W. van der Aalst, B. F. van Dongen, C. W. Gunther, A. Rozinat, E. Verbeek, and T. Weijters, "ProM: The process mining toolkit," in *Proc. BPM*, 2009, vol. 489, no. 31, pp. 1–4.

[6] W. M. P. van der Aalst, A. Bolt, and S. J. van Zelst, "RapidProM: Mine your processes and not just your data," 2017, *arXiv:1703.03740*.

[7] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Proc. Bus. Process Management Workshops*. Cham, Switzerland: Springer, 2014, pp. 66–78.

[8] C. W. Gunther and W. M. P. van der Aalst, "Fuzzy mining—Adaptive process simplification based on multi-perspective metrics," in *Business Process Management* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2007, pp. 328–343.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 823–839, May 2012.

[10] J. Wang, S. Song, X. Lin, X. Zhu, and J. Pei, "Cleaning structured event logs: A graph repair approach," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 30–41.

[11] H.-J. Cheng and A. Kumar, "Process mining on noisy logs—Can log sanitization help to improve performance?" *Decis. Support Syst.*, vol. 79, pp. 138–149, Nov. 2015.

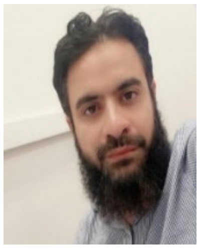
[12] S. J. van Zelst, M. F. Sani, A. Ostovar, R. Conforti, and M. La Rosa, "Filtering spurious events from event streams of business processes," in *Proc. CAISE*, 2018.

[13] D. Fahland and W. M. P. van der Aalst, "Model repair—Aligning process models to reality," *Inf. Syst.*, vol. 47, pp. 220–243, Jan. 2015.

[14] A. Armas-Cervantes, N. van Beest, M. La Rosa, M. Dumas, and S. Raboczi, "Incremental and interactive business process model repair in apromore," in *Proc. BPM Demos*. Boca Raton, FL, USA: CRC Press, 2017.

[15] A. Rogge-Solti, R. S. Mans, W. M. P. van der Aalst, and M. Weske, "Improving documentation by repairing event logs," in *The Practice of Enterprise Modeling* (Lecture Notes in Business Information Processing), vol. 165, J. Grabis, M. Kirikova, J. Zdravkovic, and J. Stirna, Eds. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-41641-5_10.

**SHABNAM SHAHZADI** received the master's degree in statistics from Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan. She is currently pursuing the Ph.D. degree with the School of Mathematics and Big Data, Anhui University of Science and Technology, China. She was a Lecturer with Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, and the National University of Modern Languages (NUML), Islamabad, Pakistan. She has authored and coauthored research articles in various academic journals. Her research interests include stochastic Petri nets, deep learning, machine learning, data management, advanced statistics, complex designs, probability theory, and statistical inference. She was a Manage Editor of *Journal of Rawalpindi Medical College* (JRMC).



**WALID EMAM** received the B.S. degree in special mathematics and the M.S. and Ph.D. degrees in mathematical statistics from the Faculty of Science, Al Azhar University, Egypt, in 2007, 2015, and 2018, respectively. His research interests include econometrics, multivariate analysis, data mining, regression analysis, survival analysis, public health, biostatistics, probability distributions, statistical inference, environmental statistics, and economic statistics.

**USMAN SHAHZAD** received the M.Sc. degree in statistics from International Islamic University, Islamabad, Pakistan, the M.Phil. degree in statistics from Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan, and the Ph.D. degree in statistics from International Islamic University, Islamabad. He was a Lecturer with Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi. He has published more than 60 research articles in research journals. His research interests include survey sampling, extreme value theory, stochastic process, probability, data mining, and non-parametric statistics. He served as an Associate Editor for the *Heliyon* journal and *Mathematics* section.

**SOOFIA IFTIKHAR** was born in Pakistan. She received the master's degree from the Department of Statistics, University of Peshawar, the M.Phil. degree from the Department of Statistics, Shaheed Benazir Bhutto Women University Peshawar (SBBWUP), and the Ph.D. degree in statistics from the University of Peshawar, specializing in the field of sampling. Since August 2006, she has been an Assistant Professor with the Department of Statistics, SBBWUP. She has authored and coauthored research articles in various academic journals. Her research interests include statistical estimation and survey sampling.

**ISHFAQ AHMAD** was born in Pakistan, in 1981. He received the B.Sc. degree in statistics, physics, and mathematics from Bahauddin Zakariya University, Multan, Pakistan, in 1999, the M.Sc. and M.Phil. degrees in statistics from Quaid-i-Azam University, Islamabad, Pakistan, in 2003 and 2005, respectively, and the Ph.D. degree in probability and mathematical statistics from the Institute of Applied Mathematics, University of Chinese Academy of Sciences (UCAS), Beijing, China, in 2010. He is currently an Associate Professor of statistics with the Department of Mathematics and Statistics, Faculty of Basic and Applied Sciences, International Islamic University, Islamabad, Pakistan. Before this, he was an Assistant Professor with King Khalid University, Saudi Arabia. His main research interests include extreme value theory, statistical inference, survey sampling, optimization, bayesian analysis, and functional analysis. He has published more than 65 articles in journals of international repute, such as the *International Journal of Climatology* and *Scientific Reports*.

**GAURAV SHARMA** received the M.Tech. degree from the Guru Jambheshwar University of Science and Technology, Hisra, and the Ph.D. degree from Punjabi University Patiala, India. He is currently a Professor with the Department of Computer Science and Engineering, Seth Jai Parkash Mukand Lal Institute of Engineering and Technology, Haryana, India. He has 20 years of teaching and academic experience. He has published approximately 44 research papers/chapters in various SCI/ESCI/SCOPUS/WoS journals, book chapters, and conferences. He has guided 12 M.Tech. candidates. His research interests include cloud computing, fog computing, and WSN. He organized various national-level seminars/conferences and faculty development programs. He also conducted/organized various technical events/workshops for diploma-level students at various institutes. He is a CSI member and reviewed various research articles from reputed publication houses such as Springer/Elsevier/IEEE.