Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 4741232, 10 pages
https://doi.org/10.1155/2022/4741232

Research Article

Repairing Event Logs to Enhance the Performance of a Process Mining Model

Shabnam Shahzadi,[1] Xianwen Fang,[1] Usman Shahzad ,[2,3] Ishfaq Ahmad,[2] and Troon Benedict [4]

[1]*Department of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China*
[2]*Department of Mathematics and Statistics, International Islamic University, Islamabad, Pakistan*
[3]*Department of Mathematics and Statistics, PMAS-Arid Agriculture University, Rawalpindi, Pakistan*
[4]*Department of Economics, Maasai Mara University, Narok, Kenya*

Correspondence should be addressed to Troon Benedict; troon@mmarau.ac.ke

Received 15 April 2022; Accepted 7 July 2022; Published 26 August 2022

Academic Editor: Tahir Mehmood

Organizations and companies are starving to improve their business processes to stay in competition. As we know that process mining is a young and emerging study that lasts among data mining and machine learning. The main goal of process mining is to obtain accurate information from the data; therefore, in recent years, it attracts the attention of many researchers, practitioners, and vendors. However, the purpose of enhancement is to extend or develop an existing process model by taking information from the actual process recorded in an event log. One type of enhancement of a process mining model is repair. It is common practice that due to logging errors in information systems or the presence of a special behavior process, they have the actual event logs with the noise. Hence, the event logs are traditionally thought to be defined as situation. Actually, when the logging is based on manual logging i.e., entering data in hospitals when patients are admitted for treatment while recording manually, events and timestamps are missing or recorded incorrectly. Our paper is based on theoretical and practical research work. The main purpose of our study is to use the knowledge gather from the process model, and give a technique to repair the missing events in a log. However, this technique gives us the analysis of incomplete logs. Our work is based on time and data perspectives. As our proposed approach allows us to repair the event log by using stochastic Petri net, alignment, and converting them into Bayesian analysis, which improves the performance of the process mining model. In the end, we evaluate our results by using the algorithms described in the alignment and generate synthetic/artificial data that are applied as a plug-in in a process mining framework ProM.

1. Introduction

As process mining is a new and emerging study that lasts among data mining and machine learning. It established a link between actual output data processes and the processes models. Process mining has three key components, which are, process discovery (learning process models from immature events), conformance checking (deviation monitoring by comparing model and log), and process enhancement (expand/upgrade existing process model), [1, 2]. Among them, the process detection aimed at obtaining processing data from the event log, they store execution data embedded in the information systems, in order to detect actual process models introduced primarily by Petri net [3], YAWL [4], or high-level Petri net [5], without prior knowledge [6]. Many algorithms are set to attain the goal successfully. However, the event logs are the main part of our study. In government, businesses, hospitals, and in other agencies, performance data embedded in information systems are usually stored in system or application logs, which later on can be converted into event logs. Appropriately, in business processes, the recorded event shows outstanding accuracy as it happens in an organization over time. Based on these acquisition algorithms, we developed a process model that, we expected.

There are lots of information systems that record detailed information regarding the supported process. Basically, they record the start and end of a process task and context data related to them. Event data collected from the logs are mostly done in business process management. After that, these logs are analyzed to get information about process performance. In some situations, information systems cannot force the process applicants to execute their work in accordance with strict procedures, as described in the process models. Instead, process participants are responsible for tracking their invisible activity sometimes in their system. Hence, according to [7], event logs may be incomplete/noisy. These data quality problems cause to affect the process mining model and mostly give us unsatisfactory results in the end.

Generally, analytics methods do not include data that is not in the system account-based process. Sometimes, things make it difficult for a few reasons, why data are not available. Another reason could be that the texts were forgotten, or missing, but actually, the work took place, or in some situations, work is not done. Generally, it is difficult to distinguish such situations, whereas the common hypothesis is that event logs have the reality that they only record those activities that actually take place.

Most commonly real-world data have incorrect or corrupt data records [8]. The most common problem, we face in recording an event log, with incorrect timestamps in a business process. As each and every event in the event log records the start or end of a processing time. Such timestamp type errors occurred in a situation when a set of events related to the same event log in a process having the same timestamp. However, these errors are caused by the cluttering of a logging system, e.g., a system does not record time but records only day, or delay during the logging process, i.e., an overcrowded logging system.

Analysis of well-researched data on such issues may give inaccurate or misleading information. For example, in the context of automated process acquisition the class of process mining techniques [9], aimed to extract a process model from an event log, in order to check the dependence among process functions, due to time stamp errors: as long as the functions ’a' and ’$b\prime$ have the same timestamp are considered to be parallel, i.e., performed by any system in the resulting process model, when in fact ’a' is the cause of ’$b\prime$, i.e., ’a' always precede ’$b\prime$ in a process model. Obviously, any information obtained from a system model, whose functions are based on inaccurate programming, will be misleading.

Model based on event data can be modified by using existing techniques [10]. In a case, if the data are recorded manually, it misleading the results and also gives less weight to the prior domain information. That is why, we apply the stochastic method to the modeling process behavior and introduce a new method of adjusting event logs based on a stochastically developed process model [9]. However, we discuss many methods to resolve the issue of incomplete event data and suggest a possible way to add missing entries to the event log based on the process provided. By using advanced Petri net with the knowledge of time and processing probabilities, we can model the process.

We use, generalized stochastic Petri nets (GSPNs) in various forms as described in [11]. Firstly, we take advantage of the path probabilities, to identify what events may be missing. Next, Bayesian networks [12] record both initial process estimates and actual observations used to calculate the most probable time stamp for each input. However, the repaired event logs are used for further analysis, e.g., in order to assist and find the responsible individuals that are used to determine the reason for entering, or to improve the mining methods by considering the complete event log. Remember that uncertainty of returned events should be in consideration at the end. As we know, this is the first task in which, we use statistical methods that are used to attain data, which is out of BPM process mining domain.

Our whole paper is systematized as; Section 2, we have related work. Section 3 based on preliminaries, in Section 4 discusses how we organize repair event logs in a timed event log. Section 5 tells us about the layout structure and put the time in the repaired event log. The evaluation of our approach using artificial data is described in Section 6. We present the conclusion in Section 7.

2. Related Work

Recent log repair methods depend heavily on alignment among event logs and given process models/process specifications. Actually, it is a significant technique in conformance checking [13]. However, in order to know who a model represents in reality we use fitness [14], accuracy [15], generality, and simplicity. Conflict in planning indicates that something is wrong and indicates, where the deviation is and how bad it is. Reference [16] discusses four confusing algorithms for logging awareness of the logging system [17], aligning event logs with declarative models. Occasionally it is not only important to know the control flow but also, we need to know about the data and resources in account [18].

The existing conformance checking methods are may be used to guide the performance of a given process model to trace the event logs. If we have any movement in a log, but it does not work on the model, then in that situation we call it to log movement; on the other hand, if the model has a movement, but is not recorded on a log, then we call it model movement. After finding out the log or model movement, then there is a problem in the alignment and we can handle this problem by repairing log instead of the model [19–24].

As mentioned earlier, when rare problems are associated with alignment and when outliers appear in the event log, research is done on aligning logs based on alignment. The missing cases can be detected with the reference to process specifications and heuristics [23]. According to [24] repair the logs for nonexistent events by repairing the control flow and timestamps. Although [22] introduces not only the missing method but also the unwanted and displaced events. We can easily access to our main objective by repairing event log and complete reference model. Hence, [20], therefore, introduces an automated method of removing abnormal behavior from event logs by using automaton logs. In addition, the two traces can also be aligned [21].

3. Preliminaries

Here, we will give the formal description and concepts about the methods and techniques we use to describe how we repair and handle missing values in the process log.

3.1. Event Logs. The set activities $'a'$ and time domain $'t\ d'$ in an event log are as follows:

$$L_{a,t\ d} = (e, c, \alpha, \gamma, \beta, \succ), \tag{1}$$

where

e: defines as events of a finite set.

c: defines in the event log as a set of cases.

α: $e \longrightarrow a$ shows an activity related to each event as a function.

γ: $e \longrightarrow t\ d$ defines function relating each event with the time domain.

β: $e \longrightarrow c$ defines in a case relating to each event as a surjective function.

$\succ \subseteq e_1 \times e_2$ shows succession relation, in order to define total ordering in the event e. However, $we_2 > e_1$ used as a notation for $(e_2, e_1) \in \succ$. Hence, therefore we can say that the ordered set of an event link with one case as a "trace."

3.2. Petri Net. It was initially presented by C.A. Petri in 1962. Basically, it is a 5-tuple with initial marking defined as $\mathrm{PN} = (P, T, F, W, M_0)$:

(i) $P = \{p_1, p_2, \ldots\ldots, p_n\}$ defines as a finite set of places (circles)

(ii) $T = \{t_1, t_2, \ldots\ldots, t_n\}$ defines as a finite set of transitions (bars)

(iii) $F \subseteq (P \times T) \cup (T \times P)$ defines as a set of arcs that link P and T

(iv) W: $F \longrightarrow \{1, 2, 3, \ldots\ldots,\}$ defined as weighted function

M_0: $P \longrightarrow \{0, 1, 2, 3, \ldots\ldots,\}$ defines as a marking denoting the number of tokens (black dots) in a place P, whereas the initial marking is denoted by M_0

3.2.1. Stochastic Petri Net. In this, we use time and data perspectives because they define the flow of information between the tasks. As each transition in SPN is exponentially distributed with firing time. It became a workflow net in a situation when; (i) when we have one place to start, (ii) having one place to end the process, and (iii) every node on the trace from start to end.

3.2.2. Generalized Stochastic Petri net. It is defined as

$$\mathrm{GSPN}\colon\ (pn, t_t, t_i, \lambda), \tag{2}$$

where

(i) r: $(p, t, \mathrm{pre}, \mathrm{post}, \mathrm{inh}, m_0)$ based on places, transition, inhibitor arc, and initial marking

(ii) $t_t \subseteq t$ shows a set of time transitions where $t_T \neq \varphi$

(iii) $t_i \subseteq t$ shows a set of immediate transitions $(t_i \cap t_T = \varphi \text{ and } t = t_i \cap t_T)$

(iv) $\lambda = (\lambda_1, \ldots\ldots, \lambda_{|T|})$ with $\lambda_i \in r_+$ define the firing delay in a case when the transition is temporized or in a situation when the probability of firing a transition is immediate

In Figure 1, we show the GSPN, where transitions are either timed (rectangular box), or immediate (zero firing time, with black bar). However, we constantly give priority to the immediate transitions, over time. The probability mass function will be used to break the firing bond between immediate transitions in order to compete with firing. In GSPN marking is over in a situation when there is at least an immediate transition to make the marking visible. In GSPN we also introduce inhibitor arcs, which connect the place with the transition. At their terminating point, an arrow has a small empty circle that shows that it is an inhibitor arc. In a situation, where we have more tokens in the output of the inhibitor, it means, we have multiplicity in arcs, and we cannot fire the transitions.

3.2.3. Generally Distributed Transition Stochastic Petri Net GDT_SPN. It is based on seven tuples $\{p, \tau, \wp, \mathrm{W}, f, m_0, d\}$, whereas basic Petri net has $\{p, t, f, m_0\}$.

(i) t: represent transition sets that are $t_i \cup t_T$ composed of timed and immediate transitions

(ii) $\wp$: $t \Rightarrow \mathcal{N}_0^+$ priorities to the transition, when $\forall_t \in t_i$: $\wp(t) \geq 1$ and $\forall_t \in t_i$: $\wp(t) = 0$

(iii) W: t_i�$\mathbb{R}^+$ show the weighted probability of the immediate transition

(iv) d: $t_T \Rightarrow d$ represents the arbitrary probability distribution to the timed transition, by showing the duration of the equivalent activities

As we describe above the SPN mode, it allows us to give arbitrary duration distribution of time in a timely manner. As in our study, we take the normally distributed duration. Note that we avoid the normal distribution because it defines the negative domain. So, we assume that most of their weighted probability lies in the positive domain, which shows us that the error presented in the correction of negative duration is not taken into consideration.

As we show an example of SPN model in Figure 2, where immediate transitions are represented with bars and timed transitions with the rectangular box. In Figure 2, the immediate transitions are represented with their weights, for example, where the process will reverse 0.25 times, leaving a loop 0.75 times. We left out the essentials and explained the 1st priority for every immediate transition. However, the timed variables are labeled from $'a'$ to $'h\prime$ and their lengths are usually distributed with their parameters. Remember that the model is chosen to be free and sound, containing loops, choices, and parallelism.

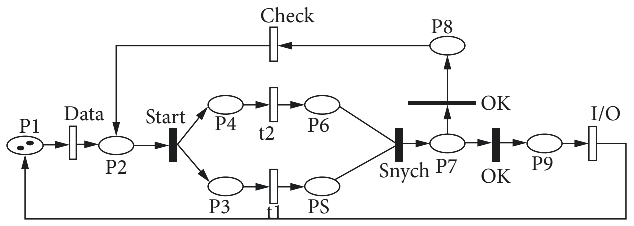

Figure 1: Generalized stochastic Petri net.

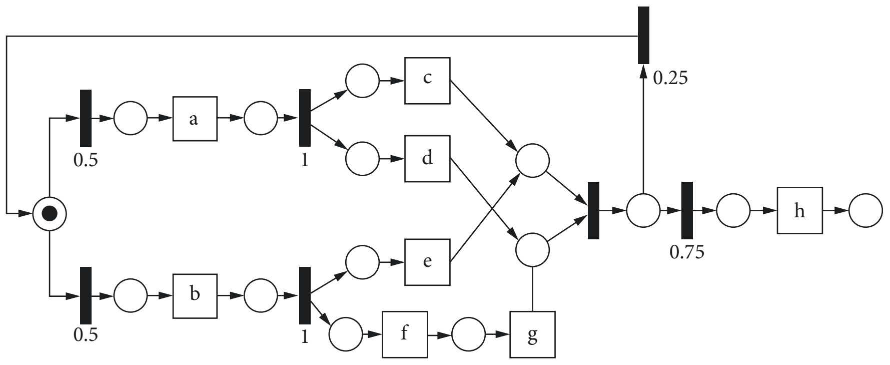

Figure 2: Generally distributed transition stochastic Petri net (GDT_SPN).

In our study, we use race semantics that allows memory describe in [25]. It means concurrently enabled transitions race are enabled right firing, whereas transitions will only be rest in a case if they are disabled for firing. Hence, the same transitions, that remain enabled after the fire does not lose their continuity.

3.2.4. Timed Transition and Immediate Transition. An empty rectangle in Figure 3 represents the timed transition and a solid rectangle is shown as an immediate transition. In SPN model the immediate transitions are fired as soon as they are enabled (no waiting time), whereas others are fired at a rate depending on transition.

3.2.5. Bayesian Network. Let $x_1, \ldots, x_n$ represents a set of random variables. It is a directed acyclic graph (n, f), whereas n shown as a set of nodes, $n_1, \ldots\ldots, n_k$ assigned to a random variable $x_1, \ldots, x_n$ and $f \subseteq n \times n$ represented as a set of directed edges. Hence, $(n_i, n_j) \in f$ be an edge from parent to child node n_i, n_j. Edges reflect the conditional dependencies among the corresponding random variable x_i, x_j. So, each random variable is independent of its predecessors, by giving value to its parent node.

Let suppose a set of parent nodes π_i represented as x_i. As we know, Bayesian network is well-defined by the probability distribution of the node 'n_i' in 'n' like $p(x_i/\pi_i)$, and their conditional dependency relation encoded. However, their joint probability distribution of the overall Bayesian network is defined as $p(x_1, \ldots\ldots, x_n) = \prod_{i=1}^{n} p(x_i/\pi_i)$.

As we know, the SPN model also captures the probability information regarding the duration of each task in a process model. Hence, therefore, we use Bayesian Network [12, 26] to know the interdependency among the random time taken from the SPN model framework. In Figure 4, we represent an example of BN that show us the relationship of a small part of the process model in Figure 2. Arcs among the function 'b' and 'e' , and among 'b', f' and 'g' are consecutive. Notice that there is no direct dependency among the function 'f' and 'e' , as they are parallel, hence, therefore, we suppose that the length of their function is independent. Generally, Bayesian Network is a straightforward acyclic graph that takes dependency among the random variables on a probabilistic model [26]. Arcs from the parent node to the child's show that the distribution of the child's probability depends on their parent values.

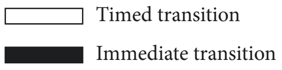

Figure 3: Timed and immediate transitions.

3.3. Cost-Based Alignment. By taking an example as in Table 1, which contains two traces 't_1' and 't_2' to check whether the traces are same as the model, we need to align them both. By using the procedure proposed by Ajmone Marsan et al. [25], which leads to the movements of a sequence that replay traces in the model. This movement is a synchronous movement, a model movement, or a log movement. The official definition of the alignment method remains outside our research work, but we discuss some important points here. In alignment, the model and log were replayed together to find out the best event design for the activities in the model. Therefore, the corresponding synchronous movement reflects events in the log by allowing right place for the model, and in this case, both model and a log move together. In a situation when model activities or

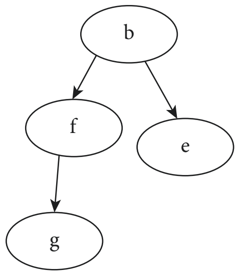

Figure 4: Bayesian network.

Table 1: Log.

t_1	$(a,$	$b,$	$c,$	$d,$	$e,$	$f,$	$g,$	$h)$
t_2	$(e,$	$g,$	$h)$					

events in a log appear to be inconsistent, then the model and the log should move asynchronously. Hence, the model movement represents the activities in the model, which does not have an event in the log in the current location, and in the same way, the log movement of an event in a log does not have any activity, that is, currently enabled model situation during recurrence. Therefore, it is possible to allocate the cost separately for each activity in different types of movement.

In Table 2, the perfect alignment among the log and model is shown with trace 't_1', which shows that the sequence can be reproduced in a sequence of the corresponding movements. As in trace 't_1' and the model in Figure 2 indicates that two cases 'b', and 'f' are not in the sequence, which may be due to documentation error. Since activity 'f' corresponds to 'e', there are two 't_2' candidate alignments, as shown in Table 3. The '⊥'symbol describes the action used to indicate non-synchronous movement, that is, modeling behavior with inconsistent recordings. In the above example, we need two model steps that align trace 't_2' to the model. However, Table 4, indicates the alignment between log and model for trace 't_2'.

In short, alignment method proposed [14, 27], that used to know the appropriate cost-optimal methods among the traces, log, and the model through their structure and sequence of the events taken into the log without having timestamps or probabilities. Hence, in our research work, we develop an alignment approach based on probabilities.

4. Repairing Event Logs

Our research work, suggests us the probabilistic way of returning from nonevent to event logs. We give in our study two ways of repairing the event log; (i) retrieving the most likely events with their corresponding time and (ii) Monte Carlo simulation, which is selected to adopt the random events, depending on their posterior probability. However, in multi-imputation method, the simulation mode is more useful component because it gives reasons to handle verity of process steps that involved nonexistence events. But here in our research work, we focus on repairing mode and discuss the variation in random events

Table 2: Alignment for trace t_1.

Log	$(a$	C	d	B	e	f	g	$h)$
Model	$(e$	C	d	B	e	f	g	$h)$

Table 3: Candidate alignment.

Log	$\perp$	e	$\perp$	g	h
Model	B	e	f	g	h

Table 4: Alignment for trace t_2.

Log	$\perp$	$\perp$	e	G	h
Model	B	f	e	G	h

where required. The problem we face here to solve is first, identify the part of the model, that is, not present in the trace (which), and then measure the estimated time of the activities present in that part of the model (when). In a theoretical perspective, it is important to compare the probabilities of all possible paths in the model that are related to the trace. All the techniques/methods may allow different distribution of the events present in a sequence with the traces. For example, in a sequence $t_2 = (e, g, h)$ of the log and model shown in Figure 2, two small ways are assumed that the model is given the alignment in Table 3, but maybe their probabilities are different. It is possible, that every loop repetition actually happened, but we may forget to consider each repetition. The path (b, e, f, g, a, c, d) is an option to repair the t_2 trace. Similarly, the 2nd multiplication move in the model with the path $(b, e, f, g, b, f, e, g, h)$, whereby taking the path in a different event like 'e' and 'g' in the t_2 of the model present. In short, we can say that there are lots of immeasurable traces in a model having the loop.

To compare the probabilities of these methods, we 1st calculate the distributing probabilities of the activities in a path and then compare it with the model methods that are relatable. After that, we focus on which activity will be performed, that defines the timing of the most visible event. To make it simpler and easier, we propose a method that divides the problem into two parts as shown in Figure 5; (i) repair the structure and (ii) insertion of tie in a process. Remember that after doing this, we accept the limitations that nonexistent events can only have if the selected path is indicating at least one in the log.

5. Logs Repairing Realization

In this part, we describe the fulfillment of the methods, we discuss above. There are certain steps required to fulfill the requirements:

(1) SPN model, which is considered a supported model, should be sound [27], and optional; [28], that need not to be edited. That type of model takes the largest part of the process model without having unnecessary limitations.

(2) The SPN model is set as a standard model, which reflects on the structural behavior and time duration.

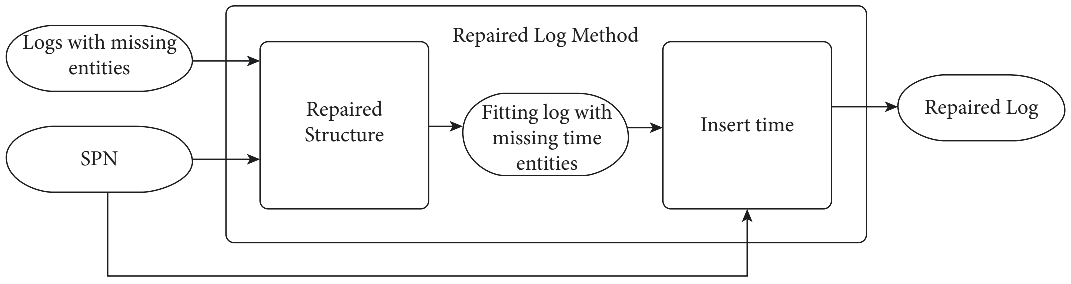
Logs with missing entities
SPN
Repaired Log Method
Repaired Structure
Fitting log with missing time entities
Insert time
Repaired Log

Figure 5: Repair control flow and timestamp.

(3) Activity duration is independent having a wide range of probabilities that contain a positive domain.

(4) Time stamp recorded in an event log should be correct.

(5) Each trace with having at least one event, and all these events contain a timestamp.

(6) Duration of case activity does not depend on circumstances.

(7) We suppose that the data are MAR and the probability of missing events and log does not depend on the timestamp of the missing events. The algorithm of our repaired event log is shown in Figure 6.

5.1. Structure Repairing. We start repairing the structure, for each trace. It became a trivial matter, and once we find out the path of some model that fit the visibility of the trace in a better way. Cost-based fitness alignment [15], discussed in Section 3, is used here and it is based on

(1) Events are parallel when the model run in a straight line

(2) When events are missing in a trace, it means model move alone

(3) There are certain events present that do not fit the model in a recorded location in a situation when log moves alone

After setting the sync cost and model move to ’0′ , then the log cost goes to the highest value, i.e., 1000. The algorithm aligns all paths for each model in a way that each event in a sequence was designed for the corresponding activity. That will also work well for the acyclic model too. In cyclic model, we have an infinite model, and we give them minimum cost per movement to the model, in the next step, we compare them with their resulting alignment values. As mentioned in Figure 6, next we select the cost-minimal alignment that we want to repair. Hence the algorithm replays the path taking place from the model and repeats their probabilities to decide along the path. By doing this way, we consider specific information on the alignment option to improve it [15]. Also, we consider that, on the other hand, parts with some missing activities are less likely than others. In this way, we can also determine the parameters of the missing data, and their level of deficit. We allow the domain expert to specify by selecting the missing events. They decide how to measure the probabilities against each other, by giving preference to the high-risk paths that determine the severity of the repaid change, or to paths with fewer incidents that do not need to be followed. This step-by-step guide us to understand the best cost and allows us to control how many modeling modes can be displayed in log of any event, i.e., by considering loop of an SPN model with ’n′ function in it. After setting the probability of deficit down, e.g., setting the probability that ’0.1′ is missing (’10%′ of the event chance is lost), continuous multiplication of the loop will be less likely, and their probability will be multiplied by a factor of ’$0.1\,n'$ this feature may be missing in all events. Very large values, such as ’0′, or ’1′ should be avoided because the probability of any alignment involving both the symmetrical and parallel movement would be ’0′.

We may choose to align the candidates by using different ways that depend on the intended use. We select the alignment with the highest probability in most/least likely scenario. But we can also select randomly depending on the probability to follow the most compelling technique. As once we finalized the structure, after repairing the trace, we move on and insert time on nonexistent events, i.e., target model movement.

5.2. Insertion of Time in the Event Log. In order to include time details, it is not enough just to consider SPN model, but we need to add information regarding each trace, that records timestamps. As we discuss in Section 3, we have a solution for inserting time in the event log, by explaining the BN. So, we convert the SPN model into BN. First, we identify the way of SPN model by giving a path, which eliminates the model selection after removing the branches from a system that are not taken into consideration. After revealing the net from the 1st marking from the selected path, note that a loop with a special type of option will be removed from the model by any given trace. By taking the following trace $t_3 = (a, d, c, a, c, d, h)$, suppose, we select the following alignment which is shown in Table 5.

As in the unfolded model given in Figure 7, black part represents the path from the model, and as it is unfolded, therefore, the gray part is removed. Unfolded model still has similarities, but because of its acyclic nature, we directly convert it into Bayesian network with the same structure, and their random variables show the time transition. Because of their frequent repetition of a loop, the operation is possible several times, therefore, we divide them by adding

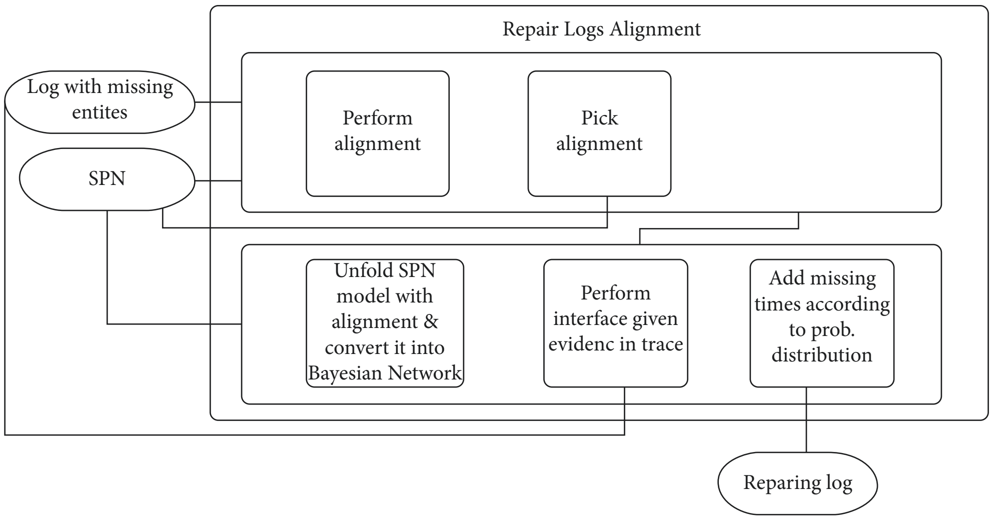
Repair Logs Alignment
Log with missing entites
SPN
Perform alignment
Pick alignment
Unfold SPN model with alignment & convert it into Bayesian Network
Perform interface given evidenc in trace
Add missing times according to prob. distribution
Reparing log

Figure 6: Repaired log algorithm.

Table 5: Alignment.

Log	$(a$	d	c	$\perp$	c	d	$h)$
Model	$(a$	d	c	a	c	d	$h)$

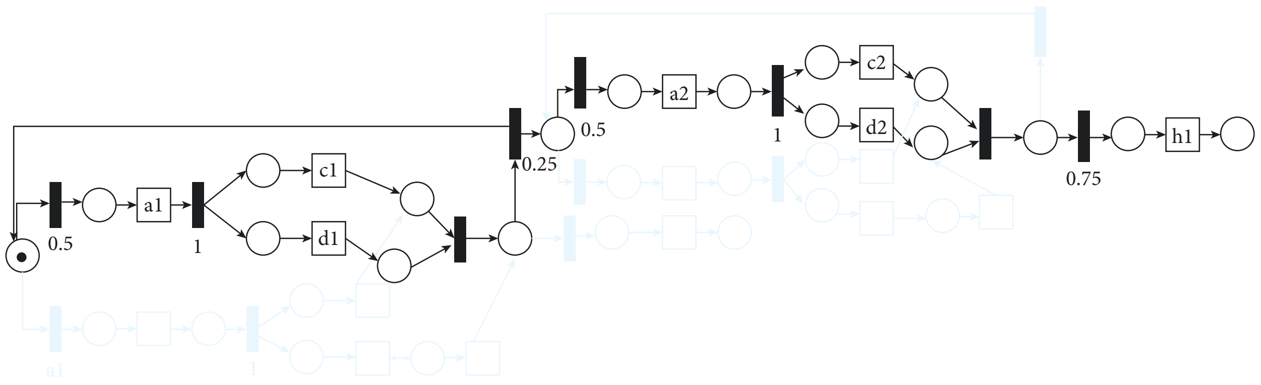

Figure 7: Unfolded Model (a, d, c, a, c, d, h) from Figure 2.

an indicator of their transitions for example, a_1 & a_2 corresponding to the 1st and 2nd occurrence of transition 'a' . However, exposure is made by crossing the path around the model and determining it by directing and keeping traces of transition events.

With the same structure, we convert the unfolded model into Bayesian network. As in Bayesian network, most of the immediate transitions are not required, because it is time-consuming and their selection needs to be made in the unfolded process, whereas only those immediate transitions, which are joined with the parallel branches will be saved.

6. Evaluation

By using ProM, we can test the quality of the algorithm described in Figure 8. The problem we face actually, whether we take the event that took place is not recorded. That is why we need to control the experiment, having a real comparison of our repaired results, we start with the trace that goes to fit model. We do this by choosing the right ones in the real world, or by simulating them in artificial cases. In the 2nd step, we can 1st find the input from the algorithm, to remove the percentage of random events in the related sequence. After that, we transfer the event log by inserting the missing log into the repaired algorithm and in the model, accordingly what to repair.

Output of the repaired algorithm can be tested against the actual trace to now, how can we handle the missing event logs. There are two quality scales that we can use for the modified log. Cost-based alignment described in [15], compares the model and the logs that how well they fit. However, by using the existing calculation techniques we compare two traces with two logs that convert each real trace into a sequential Petri net model that measures their validity with the modified traces. Eligibility is based on the quality of the structure, that is, the fair value of the test, even if it repairs the appropriate events accordingly. In order to estimate the quality of the repaired timestamps, we compare

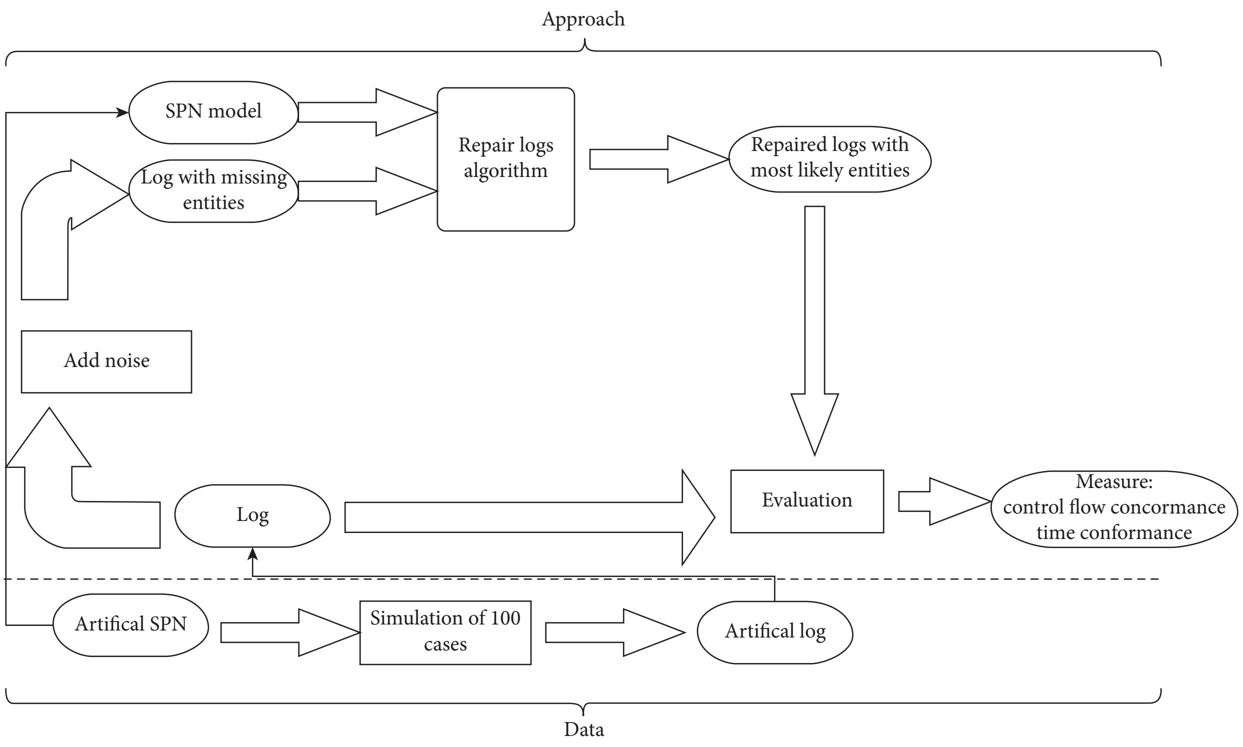
Approach
SPN model
Log with missing entities
Repair logs algorithm
Repaired logs with most likely entities
Add noise
Log
Evaluation
Measure: control flow concormance time conformance
Artifical SPN
Simulation of 100 cases
Artifical log
Data

Figure 8: Evaluation of repaired quality from event log.

the actual event time with the repaired one. That can be useful only in a sense when we make the right selection of events, which is included. MAE is useful for the total error for featured events. Basically, that is a definition of complete differences between the repaired and actual event time.

6.1. Generating Repaired Algorithm for Artificial Model. We first tested the algorithm for repair according to the performance model presented in Section 3 in Figure 2.

Figures 9 and 10 show the quality of the resulting measure we obtain after repairing the simulated traces. Tests were performed with a 100-match tracing log. However, each dot represents a repaired result of a log with varying percentages of randomly deleted events. In Figure 9, we have the values of alignment performance. Squares on the solid line indicate, the amount of corresponding movement. The other two lines are model number (circular dot) and number of log movement (gray triangular), which are required to match the traces.

The model structure given in Figure 2, has a choice between two branches combined with three high activities and four low, hence, we can return the process to the appropriate activity having a low noise level (approximately 30%). But that order cannot be confirmed because of their similarity in the model. However, change in the order of two events in a repaired sequence is because of the constant movement of one event, as the movement of the log and moving model into another place. At low-level noise note that the movement of a log and the model are the same. And at the high-level noise, there is a high probability that a single loop event is not present in Figure 2.

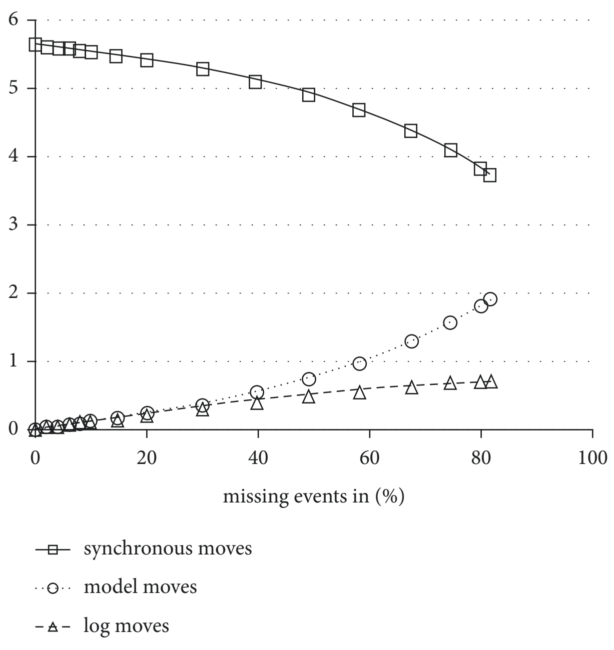

Figure 9: Actual model with 100 traces.

On the right side of Figure 9, we show the total error specified in the model with the related time unit. Graph 9 and 10; show us that the ratio between the actual and repaired event time increase with increasing noise value. From the abovementioned results, we summarize that the algorithm with repaired log is useful with low noise.

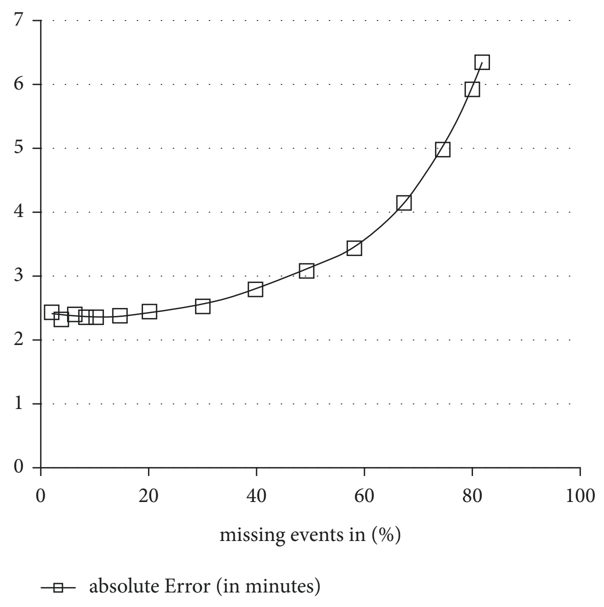

Figure 10: Repairing 100 traces from model in Figure 2.

7. Conclusion

In short, our proposed method of repairing event logs in a process mining model is available for further analysis. By this method, we divide our problem into two parts; (i) structural repairing and (ii) time insertion. However, structural repairing is performed by extension of [13], which depends on the probability path. However, time insertion is attained by taking Bayesian Network as a reference that represents the formation of each trace in a model. Algorithms can handle a large class of process models. As our first test shows that when noise is less, we can repair the structure and time. In our study, there are some limitations, which, we take into consideration; (i) we use a heuristic process to separate structure and time for repairs, in order to reduce the intricacy of the problem, as event timestamp also impacts on the probability path, (ii) normal distribution, although with good calculation limited the duration of a model activity, because of having a negative background, (iii) the assumption of interdependence between the duration of work and between traces are very strong, because resources play a vital role in the process, and (iv) assume that SPN model have the truth, and deviation from a log is due to documented error, rather than process. These limitations are only possible in a standard process when we have some deviation in a model.

In conclusion, we use knowledge gathered from the process model, by giving a technique to repair the missing event log in a trace. This technique gives us an incomplete log analysis. By using the stochastic Petri net model and alignment we repair the event log and convert them into Bayesian analysis. Hence, we evaluate our results by using the algorithms described in the alignment and generate synthetic/artificial data that is applied as a plug-in in a process mining framework ProM.

Data Availability

The data are included within the study for finding the results.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] W. Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, p. 352, Springer, Berlin, Germany, 2011.

[2] A. Polyvyanyy, W. M. P. V. D. Aalst, A. H. M. T. Hofstede, and M. T. Wynn, "Impact- driven process model repair," *ACM Transactions on Software Engineering and Methodology*, vol. 25, pp. 1–60, 2017.

[3] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, 1989.

[4] W. V. D. Aalst and A. Ter Hofstede, "YAWL: yet another workflow language," *Information Systems*, vol. 30, no. 4, pp. 245–275, 2005.

[5] K. Jensen, *High-Level Petri Nets: Applications and Theory of Petri Nets*, Informatik-Fachberichte, Springer, Berlin, Germany, 1991.

[6] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[7] F. Daniel, K. Barkaoui, and S. Dustdar, "IEEE Task Force on Process Mining," in *LNBIP 99*, pp. 169–194, Springer, 2012.

[8] G. Ciardo, R. German, and C. Lindemann, "A characterization of the stochastic process underlying a stochastic petri net," *IEEE Transactions on Software Engineering*, vol. 20, no. 7, pp. 506–515, 1994.

[9] A. Rogge-Solti, W. V. D. Aalst, and M. Weske, "Discovering stochastic petri nets with arbitrary delay distributions from event logs," in *Proceedings of the 9th International Workshop on Business Process Intelligence 2013*, Springer, Berlin, Germany, August 2013.

[10] D. Fahland and W. V. D. Aalst, "Repairing process models to reflect reality," in *Business Process Management BPM*, vol. 7481, pp. 229–245, Springer, Berlin, Heidelberg, 2012.

[11] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, no. 2, pp. 93–122, 1984.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, Massachusetts, USA, 1988.

[13] A. Rozinat and W. V. D Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, pp. 64–95, 2008.

[14] W. V. D. Aalst, A. Adriansyah, and B. Van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining and Knowledge Discovery*, vol. 2, pp. 182–192, 2012.

[15] A. Adriansyah, J. Munoz-Gama, and J. Carmona, *Alignment Based Precision Checking*, Springer, Berlin, Germany, 2013.

[16] F. Bezerra and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems," *Information Systems*, vol. 38, pp. 33–44, 2013.

[17] M. Leoni, F. Maggi, and W. Aalst, *Aligning Event Logs and Declarative Process Models for Conformance Checking*, Springer, Berlin, Germany, 2012a.

[18] M. Leoni, W. Aalst, and B. Dongen, *Data- and Resource-Aware Conformance Checking of Business Processes*, Springer, Berlin, Germany, 2012b.

[19] D. Fahland and W. M. van der Aalst, "Model repair-aligning process models to reality," *Information Systems*, vol. 47, pp. 220–243, 2015.

[20] R. Conforti, M. L. Rosa, and A. H. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 300–314, 2017.

[21] R. Jagadeesh Chandra Bose and W. M. Van der Aalst, "Process diagnostics using trace alignment: opportunities, issues, and challenges," *Information Systems*, vol. 37, pp. 117–141, 2012.

[22] W. Song, X. Xia, H. A. Jacobsen, P. Zhang, and H. Hu, "Efficient alignment between event logs and process models," *IEEE Transactions on Services Computing*, vol. 10, pp. 136–149, 2017.

[23] W. Song, X. Xia, and H. Jacobsen, "Heuristic recovery of missing events in process logs," in *Proceedings of the 2015 IEEE International. Conference. Web Services*, New York, NY, USA, June 2015.

[24] M. Leoni and W. Aalst, "Aligning event logs and process models for multi-perspective conformance checking: an approach based on integer linear programming," in *Proceedings of the 11th International. Conference. Business Manage*, June 2013.

[25] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani, "The effect of execution policies on the semantics and analysis of stochastic petri nets," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 832–846, 1989.

[26] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, p. 1270, MIT Press, Cambridge, Massachusetts, USA, 2009.

[27] A. Adriansyah, B. V. Dongen, and W. V. D. Aalst, "Conformance checking using cost-based fitness analysis," in *Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference EDOC 2011*, pp. 55–64, IEEE, Helsinki, Finland, August 2011.

[28] W. V D. Aalst, "Verification of workflow nets," in *Proceedings of the 18th International Conference on Application and Theory of Petri Nets, ICATPN'97*, pp. 407–426, Springer, Berlin, Heidelberg, January 1997.