IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 May 2022, accepted 10 June 2022, date of publication 22 June 2022, date of current version 5 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185235

Improving Process Discovery Algorithms Using Event Concatenation

MARYAM PISHGAR, (Graduate Student Member, IEEE), MARTHA RAZO, AND HOUSHANG DARABI, (Senior Member, IEEE)

Mechanical and Industrial Engineering Department, University of Illinois at Chicago, Chicago, IL 60607, USA

Corresponding author: Houshang Darabi (hdarabi@uic.edu)

This work was supported in part by the Centers for Disease Control and Prevention under Grant T42OH008672.

ABSTRACT Process mining is the discipline of analyzing and improving processes which are known as an event log. The real-life event log contains noise, infrequent behaviors, and numerous concurrency, in effect the generated process model through process discovery algorithms will be inefficient and complex. Shortcomings in an event log result in current process discovery algorithms failing to pre-process data and describe real-life phenomena. Existing process mining algorithms are limited based on the algorithm's filtering, parameters, and pre-defined features. It is critical to use a high-quality event log to generate a robust process model. However, pre-processing of the event log is mostly cumbersome and is a challenging procedure. In this paper, we propose a novel pre-processing step aimed to obtain superior quality event log from a set of raw data, consequently a better performing process model. The proposed approach concatenates events which hold concurrent relations based on a probability algorithm, producing simpler and accurate process models. This proposed pre-processing step is based on the probability of the frequency of concurrent events. The performance of the pre-processing approach is evaluated on 18 real-life benchmark datasets that are publicly available. We show that the proposed pre-processing framework significantly reduces the complexity of the process model and improves the model's F-Measure.

INDEX TERMS Process mining, concurrent relations, concatenation, pre-processing methods, probability.

I. INTRODUCTION

The goal of the process mining techniques is to support organizations by discovering, monitoring, and improving their processes [1]. Process mining has different applications in a variety of fields such as health care [2]–[4], insurance [5], and in manufacturing industry [6]. Process mining consists of three steps: process discovery, conformance checking, and enhancement [7]. Process discovery transforms an event log to a process model that describes the behavior of the processes in the forms of Petri Nets (PN) [8], Business Process Modeling Notation (BPMN) [9], Event-driven Process Chain (BPCs) [10], and Casual Nets (CN) [11]. Alpha-Algorithm [12], Heuristics Miner [13], Fodina [14], Inductive Miner [15], and Split Miner (SM) [16] are some examples of the existing process discovery algorithms. Conformance checking illustrates the deviations of the event log from the process model. Finally, the enhancement step focuses on improving the process by implementing several modifications to the process model and the event log as well. Among these three steps, process discovery plays the most fundamental step that demonstrates how process instances are executed in real life. Most of the process discovery algorithms use all available data in an event log to produce a process model. However, noise, inappropriate infrequent behaviors, and concurrency are presented in the real-life event log. The expression garbage-in garbage-out (Figure 1) underlies the fact that the poor quality of an event log would lead to poor quality of the corresponding generated process model. Therefore, the quality of the event log information is of the utmost importance when it comes to the generation of a process model [17].

The objective of the current work is to propose a pre-processing step aiming to determine a simpler and more accurate process model. This approach concatenates events which hold concurrent relations based on a probability algorithm which is explained comprehensively in the methodology section. The proposed model is tested on 18 real-life benchmark datasets which resulted in the complexity reduction of

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

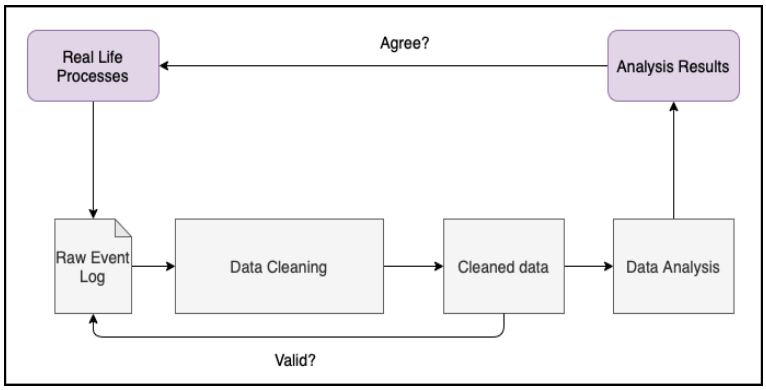


FIGURE 1. Garbage In - Garbage Out: the importance of pre-processing step.

the process model and improvement of the F-Measure metric compared to the results of using the raw event log statistically. We also compared the results of our method to those of the 4 best recent pre-processing approaches, and determined if the results are statistically significant or not.

The structure of this paper is as follows. Section II summarizes the literature review. The preliminaries are provided in Section III. We focus on the proposed approach in Section IV and Section V evaluates the approach by testing it against a raw event log. We then conclude the paper and discuss future work in section VI.

II. RELATED WORKS

Most process discovery algorithms use all behavior of the event log to generate a process model [7], [11], [18]–[20]. Although recent process discovery algorithms have been shown to remove some of the noise and infrequent behaviors in an event log [14], [15], the resultant process model is not clear in its execution semantics. Even though pre-processing of the event log is a cumbersome process, and ad hoc in its task, several methods have been developed which prioritize the motive to directly pre-process the raw event log.

A general rule for filtering activities is to filter out infrequent activities from the main event log. An example of an efficient tool that supports activity filtering is the plugin Filter Log using Simple Heuristics in the ProM process mining toolkit [21]. Another process discovery tool named Inductive Visual Miner has been discovered by Leemans *et al.* [14]. Inductive Visual Miner, filters activities by using a slider mechanism.

Another technique in performing the pre-processing tasks is to distinguish outlier traces from an event log and to filter them out. An example is provided by Ghionna *et al.* [22]. This technique initially determines frequent patterns from an event log and employs a Markov Cluster processing (MCL) on the traces of the event log. In this method, outlier tracers are considered as the ones that are unassigned to a cluster and are subsequently filtered out from the event log.

Lu *et al.* proposed another novel method, which uses event mapping that distinguishes outlier events from the events that are part of the mainstream behavior of a process. Between two executions of the process, pairing events, which were mapped to each other, are considered similar and the unmapped pairs are considered as dissimilar in behavior. Dissimilar behaviors are considered as outlier behaviors and will be filtered out from the event log to generate a more accurate process model [23].

A supervised manual approach has been proposed by Cheng and Kumar, which filters noisy traces from an event log. The authors used the Process Rule-Induction Modular (PRISM) algorithm to train marked sub-logs, which are labeled with noisy and clean traces. Unmarked sub-logs are predicated as noisy or cleaned traces. The noisy traces are then subsequently removed from the entire log [24].

Another related work is to build a prefix automaton of the event log, which was recently proposed by Conforti *et al.* [25]. The method filters out the outlier events from the event log by using an Integer Linear Programming (ILP) solver. A prefix automation event log that is minimal in terms of the number of arcs is built. Infrequent arcs and subsequent events belonging to these arcs are finally removed from the event log.

Another technique has been developed by Fani Sani *et al.* In this work, sequential pattern mining techniques are used to differentiate between outlier events and mainstream behavior events [26].

More recently, Suriadi *et al.* have suggested another pattern-based approach to provide a document that contains typical problems any event log may encounter and provides solutions to these problems. The authors confirm that these document patterns can serve as a repository of knowledge for analysis that is conducted in a semi-automated manner [27].

Tax *et al.* showed that filtering out chaotic activities from an event log helps to discover more accurate process models. Chaotic activities are defined to be those that can happen extemporaneously at any point in the process execution. Direct and indirect entropy-based activity filtering is used in separating chaotic activities from an event log [28].

Fernandez *et al.* proposed a pre-processing step to achieve execution workflows of raw event logs by using the

TABLE 1. The summary of the literature review.

Author	Year	Findings
[21]	2005	Filtering activities using Simple Heuristics in Heuristic Miner algorithm
[14]	2013	Filtering activities using a slider mechanism in Inductive Visual Miner algorithm
[22]	2008	Detecting and filtering the outlier traces from an event log by using the MCL on the traces of the event log
[23]	2005	Detecting outlier events from an event log through event mapping
[24]	2015	Detecting and filtering out noisy traces from an event log by using the PRISM rule-induction algorithm
[25]	2017	Filtering out the outlier events from the event log by using an ILP solver
[26]	2018	Detecting outlier events by using sequential pattern mining techniques
[27]	2017	Identifying typical problems that event log may encounter through pattern-based approach
[28]	2019	Filtering out chaotic activities from an event log
[29]	2020	Filtering noisy traces by using Google Cluster traces method
[30]	2020	Filtering out the specific traces using instance selection method
[31]	2020	Detecting and eliminating timestamp-related issues in the event log through their proposed model

Google Cluster traces method. The process model is then characterised and analysed by measuring the complexity metric [29].

Fani Sani *et al.* proposed an instance selection method to select specific variants or traces in the event logs to be fed to the process discovery algorithms. They introduced two types of selections which are variant-based, and trace-based sampling. In variant-based sampling, each variant in a sampled event log represents all the traces with the behavior in the original event log, so the frequency of each sample is 1. On the other hand, in trace-based sampling, the frequency of each unique sample varies and it is not 1 [30].

Fischer *et al.* presented an approach to detect and remove timestamp-related issues (timestamp imperfections) in an event log. They defined 15 metrics related to timestamp quality across two axes: four levels of abstraction (event, activity, trace, log) and four quality dimensions (accuracy, completeness, consistency, uniqueness) [31].

The summary of the mentioned methods is shown in Table 1.

The above pre-processing methods are useful to generate a more accurate and simpler model when an event log exhibits fewer concurrency between events. However, in cases where the event log contains many concurrent relations between events, these techniques would result in removing most of the relevant traces. Since most of the real-life processes contain concurrency between events and loops, it motivates us to discover a new pre-processing method that leads to significant results for any type of event log not just the event log with fewer concurrency.

To achieve this, we first find the probability of the frequency of pairs of events with concurrent relation in the event log. We then begin concatenating these events based on the descending order of the probability sums. In the case that two pairs of events have the same probability sums,a reposition function is performed which is explained in details in the methodology section. In the end, self-loops are removed from the event log. The resultant event log is fed to the SM algorithm which is a process discovery algorithm to generates an efficient process model. The efficiency of the process model is evaluated by measuring common metrics such as F-Measure, and complexity on 18 real-life benchmark datasets. We demonstrated statistical improvements in the efficiency of the process model by comparing our preprocessing approach result to that of using raw event log. Moreover, the results of our method were statistically compared to those of the 4 best recent pre-processing approaches.

III. PRELIMINARIES

In this section, we introduce process mining notations which are uniformly required throughout this paper. The preliminaries include brief definitions of the event log, directly-follows frequency, self-loops, concurrency relations, PN, F-Measure, complexity, and SM algorithm followed by an introduction to process mining and notations that ease the understanding of our pre-processing approach.

Definition 1 (Event Log): Let E be the universe of activities and $T \in E$ is a set of activities. An event log is a multi-set of sequences over T. A simple trace t is a sequence of activities, such as $t \in T$, where $t = \{e_i \in E | e_1, e_2, \ldots, e_i; 1 \leq i \leq n\}$. An event log is a multi-set of sequences over T, i.e., $L \in (T^*)$.

Moreover, each event has a label $\ell \in L$ and it refers to a task executed within a process, we retrieve the label of an event with the function $\lambda : E \rightarrow L$ to provide the label of the event, using, $\lambda(e) = e^{\ell}$ [16].

Definition 2 (Directly-Follows Frequency): Given any event log L and any two event labels ℓ, $\ell' \in L$, the directly-follows frequency of $\ell \rightarrow \ell'$ denotes the number of times ℓ' immediately appears after ℓ, in at least one trace in the given event log L. We denote this frequency with $|\ell \rightarrow \ell'|$ [16].

Definition 3 (Self-Loops): A self-loop exists in our event log L, if $|\ell \rightarrow \ell|$ is positive for some event $e \in E$ with $\lambda(e) = \ell$ [16].

Definition 4 (Concurrency Relation): Given any event logs, L and any two labels ℓ, $\ell' \epsilon L$, and any two events e_i, $e_j \in E$, are said to have concurrent relation, denoted by $(e_i||e_j)$, if and only if the following conditions exist: $|\ell \rightarrow \ell'| > 0$ and $|\ell' \rightarrow \ell| > 0$ where $\lambda(e) = \ell$ and $\lambda(e') = \ell'$ [16].

Definition 5 (Petri Net): A Petri net (PN) is a mathematical modeling technique, which is used to represent a process model in many different areas such as healthcare, and manufacturing. A PN can be formally described as

$$PN = (P, Ti, F, \pi),$$

P is a set of places, Ti is a set of transitions,

$$F \subseteq (P \times Ti) \cup (Ti \times P),$$

is a set of directed arcs, and π is a function that maps transitions to activities [32]. PNs are visualized as follows: circles represent places and transitions are signified with rectangles. Unidirectional arcs that are the set of F, connect transitions to the places and vice versa. The function,

$$\pi : Ti \rightarrow U \cup \{\perp\},$$

maps each transition to the activity $u \in U$, note that an activity can be non-observable, and these transitions are called hidden transitions and are shown by a black rectangle. PNs contain at least one initial marking and one final marking which represents the start and the end of a process execution respectively. A transition is called enabled if it fires. In order to fire a transition, all input places of a transition must hold at least one token. During firing a transition, the tokens move from all places which are inputs to the transition. As a result, the number of tokens for the mentioned places will be reduced by one. On the other hand, the tokens will be entered to the places which are the outputs of that transition, so the number of the token for those places increase by one. One of the important properties to maintain in a PN is the soundness of the PN [1].

Definition 6 (F-Measure): F-Measure is a common metric, which is used to measure the quality of the process discovery models. It creates a balance between fitness and precision values. In other words, F-Measure is a trade-off value between fitness and precision [33], and it is calculated based on the following formula:

$$FMeasure = \frac{2 \times Fitness \times Precision}{Fitness + Precision}$$

$$= \frac{TP}{TP + \frac{1}{2}(FP + FN)},$$

such that,

TP = Number of true positives

FP = Number of false positives

FN = Number of false negatives

Note that fitness defines how well a process model shows the behavior of the event log. If the value of fitness is 1, this means the model is able to reproduce all traces in the event log. In this paper, we use an alignment-based approach to measure fitness [34]. In the alignment-based method, the degree of the alignment between real traces in the log and the produced traces in the process model is measured.

Precision shows the capability of a model to produce the behaviors that are found only in the event log. The value of 1 shows that traces that are produced in the process model are available in the event log. In this work, the alignment-based method is used to measure precision in [35].

Definition 7 (Complexity): Complexity is another metric that is used to measure the quality of the process model. Complexity shows how much a process model is easy to understand. Several metrics are used to measure the complexity of a model [36]. In this work, Control-Flow Complexity (CFC) [37], size of the models [38], and structuredness [38] are used to measure the complexity of the models. CFC shows how many branching is prompted by the split gateways in a process model. The higher the amount of CFC value is, the more complex the model is. The size of a process model calculates the numbers of the nodes and arcs in the model. The lower the number of nodes and arcs in a process model, the lower the complexity. In the end, a process model is structured, If, for any split gateway in the process model, there is a corresponding join gateway. The more structured is a process model, the simpler the model is.

Definition 8 (Split Miner): SM is one of the process discovery algorithms that produce a sound PN from the event log, and it shows pointedly more improvement in producing models with less complexity and higher F- measure value than other process discovery algorithms. SM algorithm includes the 5 following steps. First, it generates a directly-follow graph and identifies short loops. Second, it discovers the existing concurrency relation between events. Then, it utilizes filtering, and in order to guarantee the soundness of this process, each node should be located on a path from the start node to the end node. Afterward, the algorithm derives choice and concurrency relations by adding split gateways. In the end, the join gateways are discovered. To accomplish this, SM also introduces 2 threshold values. The first one controls the filtering process, which is called frequency threshold, ϵ. The other one is η, which controls concurrency relations. Both of these parameters are percentile which numerical values are in the range of 0 to 1. The output of these steps is a BPMN, and it can be converted to a PN through using ProMs' BPMN Miner package [1]. This algorithm is publicly available as a java application [39].

Definition 9 (Process Mining): Process mining consists of 3 steps: process discovery, conformance checking, and enhancement of the process model [1]. An event log is the input of a process discovery algorithm and the output is the process model that can be in different forms such as PN, BPMN, EPCs, and CNs. We focus on BPMN, which is the output of the SM. Conformance checking evaluates whether the process model is an accurate representation of the event log. The most commonly used metrics of evaluations are F-Measure and complexity. Enhancement is used to provide improvement to the event log and its corresponding process model. By doing process discovery and conformance checking, we will uncover any problems, loopholes, and potential solutions, which can be subsequently implemented. Finally, we provide recommendations to improve a process model.

Given an event log, L containing containing n number of events and m number of traces, $t_j = \{e_1, e_2, \ldots, e_i\}$, such that $t_j \in L$, where $1 \leq j \leq m$ and $1 \leq i \leq n$.

We define $f(e_i)$ as follows:

$$f(e_i) = count(e_i) \tag{1}$$

where $count(e_i)$ is the frequency or a total number of times the event e_i is present in every trace t_j in the event log, L.

Let e_i and e_j be two events in the given event log, L. We define the frequency or number of times that e_j happens instantly after e_i in the event logs, L as follows:

$$f(e_i \rightarrow e_j) = \sum (e_i \rightarrow e_j) \tag{2}$$

Similarly, the frequency or number of times that e_i happens instantly after e_j in the event log, L is denoted as follows:

$$f(e_j \rightarrow e_i) = \sum (e_j \rightarrow e_i) \tag{3}$$

We denote the probability of the frequency that an event e_j occurs immediately after event e_i as,

$$P(e_i \rightarrow e_j) = \frac{f(e_i \rightarrow e_j)}{f(e_j)}. \tag{4}$$

Similarly, we define the probability of the frequency that an event e_i occurs immediately after event e_j as follows:

$$P(e_j \rightarrow e_i) = \frac{f(e_j \rightarrow e_i)}{f(e_i)}. \tag{5}$$

Let $P(e_i||e_j)$ be the probability sum of a concurrent set of events $e_i||e_j$ such that,

$$P(e_i||e_j) = P(e_i \rightarrow e_j) + P(e_j \rightarrow e_i) \tag{6}$$

and

$$P(e_k||e_l) = P(e_k \rightarrow e_l) + P(e_l \rightarrow e_k) \tag{7}$$

denote the probability sum of a second set of concurrent events, $e_k||e_l$.

For two concurrent events (e_i, e_j) and $(e_k, e_l) \in L$ such that $e_i||e_j$ and $e_k||e_l$, we define an index for a set of concurrent events $e_i||e_j$ as $(I^s e_i, I^{s+1} e_j)$, such that $I^s e_i \neq I^{s+1} e_j$, where $0 \leq s < \infty$. We define a second index for $e_i||e_j$ as $(I^q e_i, I^{q+1} e_j)$, such that $I^q e_i \neq I^{q+1} e_j$, where $0 \leq q < \infty$. Similarly, for a second set of concurrent events $e_k||e_l$, we define an index for $e_k||e_l$ as $(I^s e_k, I^{s+1} e_l)$, such that $I^s e_k \neq I^{s+1} e_l$. We define a second index for $e_k||e_l$ as $(I^q e_k, I^{q+1} e_l)$, such that $I^q e_k \neq I^{q+1} e_l$.

Let e_i and e_j be two concurrent events. We define $F^s_{e_i e_j}$ as the total number of instances that e_i and e_j are concurrent at the index $(I^s e_i, I^{s+1} e_j)$ and $F^q_{e_i e_j}$ as the total number of instances that e_i and e_j are concurrent at the index $(I^q e_i, I^{q+1} e_j)$ for each trace t_j in the event log, L.

Finally, p^* denotes the cut-off probability in a given concurrent relation.

IV. APPROACH

The proposed approach is a pre-processing algorithm that aims to remove concurrency and self-loops based on a probability algorithm to improve the quality of an event log for achieving an optimal process model. The method achieves a higher F-Measure compared to maneuvers using the raw event log. Moreover, our approach results in a simpler and more accurate model.

The approach consists of the following steps. Given an event log, L, containing the sequence of events $E = \{< e_i, e_j, \ldots..e_n >\}$, we find all $P(e_i||e_j)$ for all possible combinations of concurrent events, $(e_i||e_j)$ in the event log. We introduce a threshold value of p^* and only select the combinations whose probabilities pass the threshold value. Any given concurrent relation is selected if and only if both of its corresponding probabilities exceed the predefined threshold p^*. We then add the probabilities of the selected combinations and sort them in descending order. We perform a re-position step for the combinations with equal probability sums. After the re-position step, we concatenate the ordered combinations on the original event log. Finally, we remove all the self-loops that are presented on the event log. The intuition behind concatenating events with a concurrent relation is that in many real-life event log, concurrency accounts for a significant part of the behavior captured on the event logs. Therefore, by concatenating some of the concurrent events, the complexity of the model decreases significantly. The various steps of the proposed model are shown in Figure 2. The corresponding source code is publicly available on our GitHub repository.

A. PROBABILITY OF THE FREQUENCY OF THE EVENTS WITH A CONCURRENT RELATION

1) *Discovering concurrent events.*
In this stage, we discover all concurrent relations between all sets of 2 events by calculating all possible combinations of 2 events using $\binom{n}{2} = \frac{n!}{2!(n-2)!}$ and choosing only those events that are concurrent such that $(e_i \rightarrow e_j)$ and $(e_j \rightarrow e_i)$.
2) *Finding $P(e_i \rightarrow e_j)$ and $P(e_j \rightarrow e_i)$.*
We calculate the frequency, $f(e_i)$, for each event e_i, in each trace, $t_j \in L$.

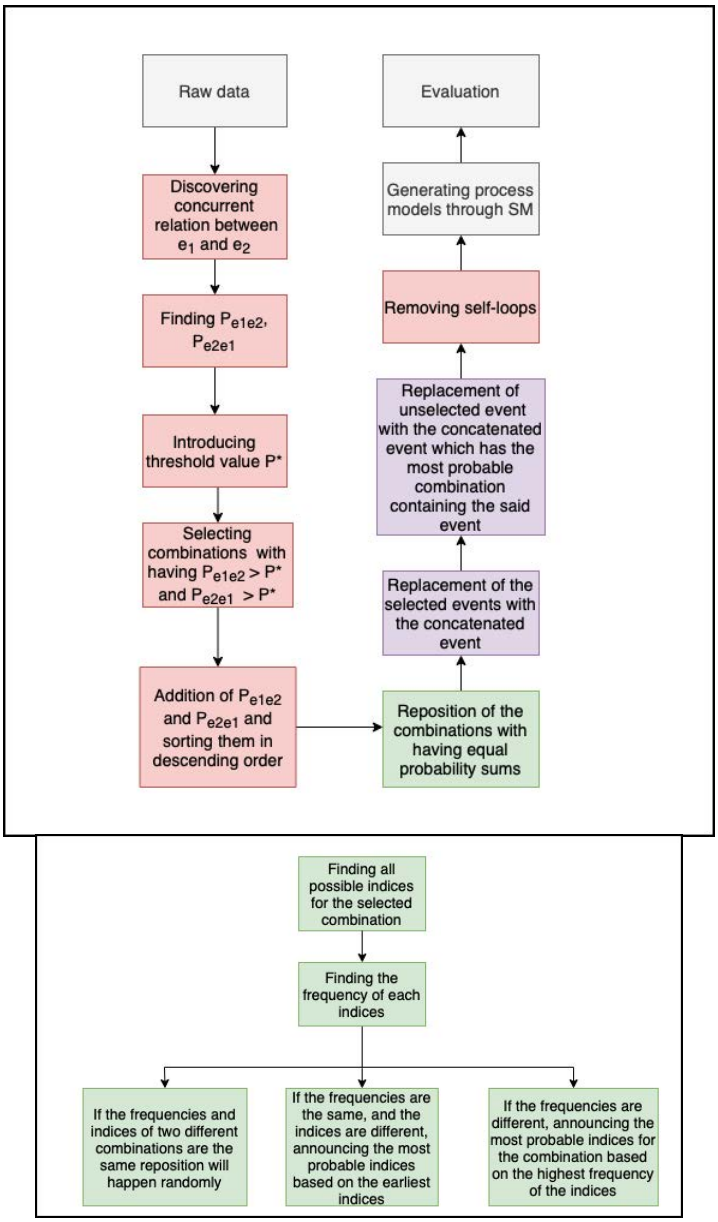


FIGURE 2. Proposed model: the top figure shows each step of the proposed model and the bottom one illustrates the details of the reposition step.

We then calculate $f(e_i \rightarrow e_j)$ and $f(e_j \rightarrow e_i)$ of only the concurrent events $e_i||e_j$, and $P(e_i \rightarrow e_j)$ and $P(e_j \rightarrow e_i)$ respectively.

3) *Introducing threshold value p^*.*
We introduce the threshold value p^* which denotes the cut-off probability in a given concurrent relation. The value of the threshold p^* is found via hyper-parameter optimization and selection of the highest F-Measure values.

4) *Selecting combinations using p^*.*
The concurrent combination of events will be chosen if and only if both $P(e_j \rightarrow e_i)$ and $P(e_i \rightarrow e_j)$ exceed the value of threshold p^* such that,

$$P(e_i \rightarrow e_j) > p^* \text{ and } P(e_j \rightarrow e_i) > p^*.$$

5) *Adding and sorting concurrent event probabilities.*
Lastly, we take the sum of $P(e_i \rightarrow e_j)$ and $P(e_j \rightarrow e_i)$ for all concurrent events that meet the condition above. For example, let $P(e_i||e_j)$ be the probability sum of a concurrent set of events $e_i||e_j$ such that,

$$P(e_i||e_j) = P(e_i \rightarrow e_j) + P(e_j \rightarrow e_i)$$

and

$$P(e_k||e_l) = P(e_k \rightarrow e_l) + P(e_l \rightarrow e_k)$$

denotes the probability sum of a second set of concurrent events, (e_k, e_l).

We then sort all the $P(e_j||e_i)$ for all concurrent events chosen in descending order the following way:

$$P(e_i||e_j) > P(e_k||e_l).$$

B. RE-POSITION OF THE COMBINATION WITH EQUAL PROBABILITY SUMS

This step uses the selected combinations of step A as an input. In this stage, those selected combinations with concurrent relation are sorted, in descending order, based on their probability sums. However, we may come across several cases with equal probability sums, for which case re-position is necessary to sort them for the concatenation step. Reordering the position of those combinations is done according to the following steps.

For $P(e_i||e_j) = P(e_k||e_l)$

For concurrent events (e_i, e_j) and $(e_k, e_l) \in L$ such that $P(e_i||e_j) = P(e_k||e_l)$.

1) For 2 concurrent events, $e_i||e_j$ we find all the indices, $(I^s e_i, I^{s+1} e_j)$ of (e_i, e_j), for every trace, $t_j \in L$.
2) For $e_i||e_j$, we calculate $F^s_{e_i e_j}$ and $F^q_{e_i e_j}$ as the total number of instances that e_i and e_j are concurrent at the index $(I^s e_i, I^{s+1} e_j)$ and $(I^q e_i, I^{q+1} e_j)$ respectively.
3) If $F^q_{e_i e_j} > F^s_{e_i e_j}$, then we announce indices $(I^q e_i, I^{q+1} e_j)$ as the most probable or likely index for concurrent events $e_i||e_j$.
4) We repeat steps $1-3$ for all for all the concurrent event combinations with equal probability sums.

For $P(e_i e_j) = P(e_k e_l)$ and $F^s_{e_i e_j} = F^q_{e_i e_j}$

We take 2 concurrent events (e_i, e_j) and $(e_k, e_l) \in L$ such that $P(e_i||e_j) = P(e_k||e_l)$ and $F^s_{e_i e_j} = F^q_{e_i e_j}$.

1) Let's take 2 indices for the concurrent events $e_i||e_j$, $(I^s e_i, I^{s+1} e_j)$ and $(I^q e_i, I^{q+1} e_j)$ such that $F^s_{e_i e_j} = F^q_{e_i e_j}$.
2) We find that $(I^s e_i, I^{s+1} e_j)$ happens before $(I^q e_i, I^{q+1} e_j)$ for $e_i||e_j$.
3) We announce, $(I^s e_i, I^{s+1} e_j)$ as the earlier indices between $F^s_{e_i e_j} = F^q_{e_i e_j}$
4) We repeat steps $1-3$ for all the concurrent event combinations with equal probability sums and equal frequencies for the indices.

$P(e_j||e_i) = P(e_k||e_l)$ and $F^s_{e_i e_j} = F^q_{e_i e_j}$ and $(I^s e_i, I^{s+1} e_j) = (I^q e_i, I^{q+1} e_j)$

In the rare scenario where 2 combinations may have the same probability sum and the same most likely indices in an event log, such that $P(e_j||e_i) = P(e_k||e_l)$, $F^s_{e_i e_j} = F^q_{e_i e_j}$ and $(I^s e_i, I^{s+1} e_j) = (I^q e_i, I^{q+1} e_j)$, we choose the order of combinations for the concatenation randomly.

C. CONCATENATION AND REMOVING SELF-LOOPS

1) *Replacement with the concatenated event.* Thus far, the order of the combinations for concatenation is finalized in the previous step. Our algorithm selects the ordered combinations one by one for the concatenation. In order to concatenate the 2 events in a combination, suppose that we have a combination (e_i, e_j), we remove the event, e_i replace e_j with the concatenated event, $e_i \oplus e_j$. Note that the name of the concatenated event is derived based on the combination of 2 event names which are going to be concatenated. In this case, the name of the concatenated event will be $e_i \oplus e_j$. This step will be repeated until all selected combinations will be concatenated in an event log.
2) *Replacement with most probable combination.* After concatenating all combinations in the event log based on descending order, there will be events that are yet not concatenated. We examine each event individually that is not concatenated and see if the event, $e_i \in e_i \oplus e_j$. If that is the case, we remove e_i and replace e_i with $e_i \oplus e_j$. If $e_i \nexists e_i \oplus e_j$, then the final trace will remain unchanged.
3) *Removing Self-Loops.* Finally, we remove all the self-loops, such that $e_i \rightarrow e_i$.

Let $E = \{< e_i, e_j, \ldots..e_n >\}$ be the sequence of events in a given trace t_j and $T = t_1, t_2, \ldots, t_n$ be the set of all traces in the event log, L. Our algorithm is as follows:

Algorithm 1 Stochastic Concatenation Algorithm

Input: Raw event log, L;
Output: Output event log, L'';
Initialisation :
Using $0 < p^* < 1$;
$C \leftarrow [\ ], D \leftarrow \{\}, S \leftarrow \{\}, R \leftarrow \{\}, G \leftarrow \{\}$;
LOOP Process
for $e_i||e_j \epsilon L, i < j$ **do**
  if $P(e_i \rightarrow e_j) > p^*$ and $P(e_j \rightarrow e_i) > p^*$ **then**
    Add $e_i||e_j$ to C;
  end if
end for
for $e_i||e_j \epsilon C$ **do**
  $D(e_i||e_j) \leftarrow P(e_i||e_j) + P(e_j||e_i)$. Sort D based on descending order of the probability sums. Add all combinations with equal probability sums to R.
end for
for $e_i||e_j \epsilon R$ **do**
  Find all indices, $(I^s e_i, I^{s+1} e_j)$;
end for
for $(I^s e_i, I^{s+1} e_j)$ **do**
  $S\{I^s e_i, I^{s+1} e_j\} \leftarrow$ Find all $F^s_{e_i e_j}$, the total number of instances that e_i and e_j are concurrent at the index $(I^s e_i, I^{s+1} e_j)$;
end for
for $(I^s e_i, I^{s+1} e_j)$ and $(I^q e_i, I^{q+1} e_j) \epsilon S$ **do**
  if $F^s_{e_i e_j} > F^q_{e_i e_j}$ **then**
    $G \leftarrow$ the most probable indices for $e_i||e_j$ =the indices of $F^s_{e_i e_j} \epsilon S$
  else
    if $F^s_{e_i e_j} = F^q_{e_i e_j}$ **then**

$G \leftarrow$ the most probable indices for $e_i||e_j$ = the earlier indices between $F^s_{e_ie_j}$ and $F^q_{e_ie_j} \epsilon S$;
end if
end if
end for
for $e_i||e_j \epsilon G$ **do**
if $(I^s e_i, I^{s+1} e_j) = (I^q e_i, I^{q+1} e_j)$ **then**
$G \leftarrow$ The order of combination for concatenation will be chosen randomly;
end if
end for
Replace the keys of D which has equal probability sums with the corresponding ordered keys of G;
for $e_i||e_j \epsilon T$ **do**
if $e_i||e_j \epsilon D$ **then**
L' $\leftarrow$ Drop e_i and replace e_j with the concatenated event which is $e_i \oplus e_j$;
end if
end for
for $e_i \epsilon$ **do**
L'
if $e_i \epsilon D$ **then**
L'' $\leftarrow$ Replace e_i with the concatenated event, which has the most probable combination containing the e_i in D ;
end if
end for
for $e_i \rightarrow e_i \epsilon$ **do**
L''
L'' $\leftarrow$ Remove first e_i ;
end for
Return L''

V. EVALUATION

In this section, the proposed model is evaluated. After modifying the event log, we used the SM algorithm to generate the process models. The SM algorithm takes an event log in XES or MXML formats, and the thresholds of η and ϵ as inputs, and produces a BPMN process model as an output. For evaluation, a set of publicly available logs were used. In the following parts, we provide a summary of benchmark datasets. Then we discuss the results.

A. DATASETS

4TU Center of Research Data as of August 2021 provided a collection of several real-life event logs [40]. The event logs listed in Table 2 were used to evaluate the proposed approach: All logs of annual Business Process Intelligence Challenge (BPIC), Road Traffic Fine Management Process (RTFMP), SEPSIS Cases log from a hospital each of which denotes the pathway through the hospital, and Hospital event log which contains information related to billing.These logs record executions of processes in a variety of fields such as healthcare, finance, and government affairs. A pre-processing step [41] was applied to remove infrequent behaviors from the BPIC15, BPIC14, and BPIC17 logs because of the complexity of the logs. Before removing infrequent behaviors, the process discovery algorithm generates a model with F-Measure close to zero due to the complexity of the event logs. The statistics of the logs are shown in Table 2.

B. RESULTS

To conduct the experiments, we used RapidProM [42] which extends RapidMiner with process mining analysis capabilities. This platform helps us to use a workflow for the experiments. The algorithm takes a threshold value of p^* to choose the combinations of concurrent events for the concatenation step.

To find the optimal of p^*, we ran hyper-parameter optimization with the steps of 0.1 on 18 benchmark datasets. The optimal p^* was chosen based on the highest value of the F-Measure. The experimental results showed that if the optimal value for p^* is chosen as 0.7, it yielded the highest value of F-Measure in 13 out of 18 datasets. However, the rest of those 5 had no to negligible differences in F-Measure before and after concatenation. Consequently, the optimal value for p^* was chosen to be 0.7. Figure 3 illustrates the variation in F-Measure by applying hyper-parameter optimization with the step of 0.1 on 18 benchmark datasets. Based on this figure, it can be observed that the F-Measure value peaks at the threshold of 0.7 on almost all datasets.

Also, to choose the optimal values of SM thresholds, we used 16 different settings, i.e., the threshold η in [0.1, 0.2, 0.3, 0.4] and ϵ in [0.1, 0.2, 0.3, 0.4]. Note that the SM default threshold values of η and ϵ are 0.4 and 0.1 receptively. The optimal value of the parameters were chosen based on the highest value of the F-Measure. We found that the optimal value of η and ϵ thresholds are 0.4 and 0.4 respectively.

For the first experiments, we used the optimal values of the thresholds for the SM algorithm, and the raw event logs to discover process models. Then, we applied our methodology to the raw event logs and used the optimal value of the thresholds for our algorithm and the SM algorithm to discover the process models from the concatenated logs. Finally, the quality of the process models was measured based on the common metrics of fitness, precision, and F-Measure as the proxies of accuracy, size, CFC, and structuredness as the proxies of complexity. The results of this evaluation on 18 benchmark datasets are summarized in Tables 3 and 4. In the cases in which improvements were observed in any of the evaluation metrics after applying concatenation, the Wilcoxon Test was run to calculate the P-value, and find out if the improvements are statistically significant or not. The level of significance was selected to be 0.05.

The measuring tools to calculate fitness and precision, work on the PN. However, the output of SM process discovery is a BPMN model. Therefore, we needed to convert the resulting BPMN models to the PN, which is performed by using ProMs' BPMN Miner package [43]. The complexity metrics were computed on BPMN models. All the tests were

TABLE 2. Statistics of the event logs.

Event logs	Original Logs				Pre-Processed Logs			
	Total Traces	Distinct Traces	Total Events	Distinct Events	Total Traces	Distinct Traces	Distinct Events	%of Remaining Traces
BPIC12	13087	4366	262200	36	9291	1191	32	71
$BPIC13_{cp}$	1487	183	6660	7	1263	38	4	85
$BPIC13_{inc}$	7554	1511	65533	13	6043	111	12	80
$BPIC14_{f}$	41353	14948	369485	9	32668	9454	7	79
$BPIC15_{1f}$	902	295	21656	70	884	66	70	98
$BPIC15_{2f}$	681	420	24678	82	667	417	80	98
$BPIC15_{3f}$	1369	826	43786	62	1204	810	48	88
$BPIC15_{4f}$	860	451	29403	65	817	431	55	95
$BPIC17_{5f}$	975	446	30030	74	966	432	60	99
$BPIC17_{f}$	21861	8767	7141198	41	19674	1722	12	90
RTFMP	150370	231	561470	11	103755	171	10	69
SEPSIS	1050	846	15214	16	766	425	12	73
BPIC 18-control	43808	1231	59	7	33707	41	4	77
BPIC 18-inspection	5485	6732	3190	15	4321	2197	9	79
BPIC 18 Reference	43802	5463	515	6	42983	427	4	98
BPIC 19	251734	11973	9463	44	241111	10000	37	95
BPIC 20 ID	1050	846	15214	16	766	425	12	73
Hospital	100000	1020	11132	18	878	89870	12	89

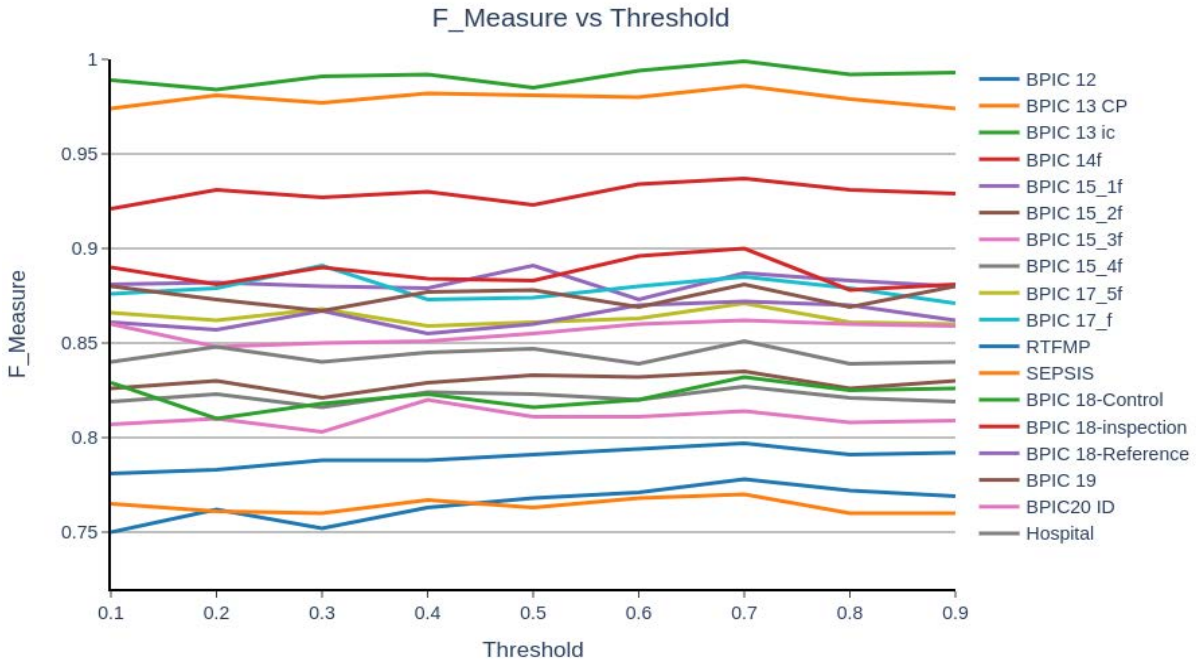


FIGURE 3. Hyper-parameter optimization with the step of 0.1 on 18 benchmark datasets and the respected F-Measures.

performed on a computer running Windows 10 with an Intel i7-6700 CPU and 16 GB RAM.

Table 3 shows the experimental results for the best obtained F-Measure values, corresponding fitness, and precision values before and after pre-processing. Our results show that the F-Measure values were significantly improved by concatenating the events, which have a concurrent relation in the real-life event logs. 14 of 18 benchmark datasets have clearly shown major improvement in F-Measure values, which is depicted in Figure 4c. Also, Figure 4a and Figure 4b illustrate the fitness and precision respectively for the corresponding F-Measure.

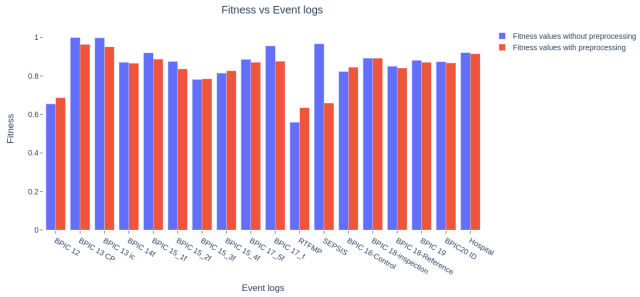
TABLE 3. Results of the proposed model in terms of fitness, precision, and F-Measure on the event logs, and the Wilcoxon test results on the improved cases to statistically identify the level of significance.

Event logs	Fitness before Concatenation	Fitness after Concatenation	P-value	Precision before Concatenation	Precision after Concatenation	P-value	F-Measure before Concatenation	F-Measure after Concatenation	P-value
BPIC12	0.655	0.687	0.041	0.723	0.673	NA	0.687	0.778	0.042
$BPIC13_{cp}$	0.999	0.964	NA	0.931	0.974	0.039	0.964	0.986	0.041
$BPIC13_{inc}$	0.998	0.951	NA	0.908	0.998	0.038	0.951	0.999	0.036
$BPIC14_{f}$	0.871	0.866	NA	0.861	0.962	0.036	0.866	0.937	0.023
$BPIC15_{1f}$	0.920	0.887	NA	0.857	0.857	NA	0.887	0.887	NA
$BPIC15_{2f}$	0.875	0.836	NA	0.800	0.804	0.054	0.836	0.835	NA
$BPIC15_{3f}$	0.782	0.785	0.053	0.789	0.748	NA	0.785	0.814	0.040
$BPIC15_{4f}$	0.815	0.827	0.053	0.839	0.839	NA	0.827	0.827	NA
$BPIC17_{5f}$	0.886	0.871	NA	0.857	0.857	NA	0.871	0.871	NA
$BPIC17_{f}$	0.956	0.876	NA	0.809	0.839	0.042	0.876	0.885	0.060
RTFMP	0.559	0.635	0.023	0.736	0.807	0.039	0.635	0.797	0.027
SEPSIS	0.976	0.659	NA	0.500	0.679	0.032	0.659	0.770	0.024
BPIC 18-control	0.823	0.845	0.041	0.779	0.810	0.047	0.810	0.832	0.042
BPIC 18-inspection	0.892	0.892	NA	0.849	0.901	0.039	0.873	0.900	0.043
BPIC 18Reference	0.850	0.841	NA	0.856	0.910	0.035	0.853	0.872	0.029
BPIC 19	0.881	0.871	NA	0.812	0.889	0.028	0.851	0.881	0.041
BPIC 20 ID	0.874	0.867	0.039	66	60	0.045	260	245	0.046
Hospital	0.921	0.915	NA	0.732	0.781	0.045	0.826	0.851	0.043

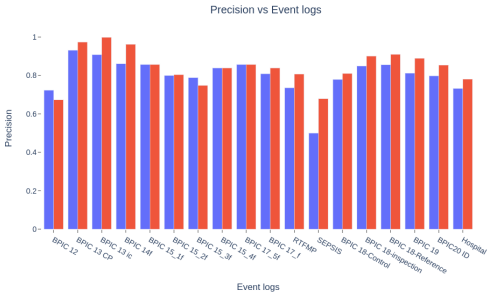
TABLE 4. Results of the proposed model in terms of CFC, size, structuredness, and the Wilcoxon test results on the improved cases to statistically identify the level of significance.

Event logs	CFC before Concatenation	CFC after Concatenation	P-value	Size before Concatenation	Size after Concatenation	P-value	Structurdness before Concatenation	Structurdness after Concatenation	P-value
BPIC12	144	66	0.031	72	38	0.025	2230	2120	0.024
$BPIC13_{cp}$	16	10	0.020	12	8	0.028	68	44	0.043
$BPIC13_{inc}$	19	5	0.030	14	12	0.059	80	20	0.039
$BPIC14_{f}$	33	18	0.029	21	12	0.042	304	80	0.025
$BPIC15_{1f}$	88	88	NA	72	72	NA	1320	1320	NA
$BPIC15_{2f}$	146	142	0.051	105	103	0.052	2544	3100	NA
$BPIC15_{3f}$	141	140	0.050	109	94	0.042	2578	1854	0.035
$BPIC15_{4f}$	104	101	0.050	77	75	0.052	1374	1338	0.050
$BPIC17_{5f}$	111	111	NA	84	84	NA	240	240	NA
$BPIC17_{f}$	35	23	0.034	25	18	0.043	100	74	0.042
RTFMP	50	48	0.050	32	30	0.049	5412	5410	0.053
SEPSIS	90	52	0.026	43	24	0.051	5436	4423	0.036
BPIC 18-control	145	138	0.040	61	60	0.050	982	901	0.043
BPIC 18-inspection	134	122	0.035	46	40	0.041	878	874	0.056
BPIC 18Reference	165	154	0.042	53	43	0.037	67	59	0.045
BPIC 19	155	130	0.046	46	39	0.042	404	344	0.039
BPIC 20 ID	166	131	0.039	66	60	0.045	260	245	0.046
Hospital	176	101	0.029	67	59	0.026	409	376	0.041

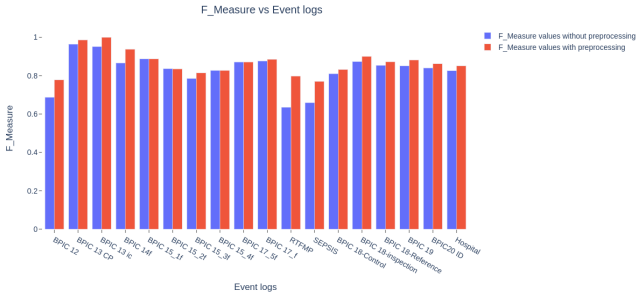
Table 4 shows the results of the comparison of the discovered process models with respect to their complexity both before and after the pre-processing. The complexity of the models has been evaluated based on 3 different metrics; CFC, size, and structuredness, which we have mentioned earlier in the preliminaries section. All of these metrics are inversely related to the understandability of the process models. Based on our results, 16 out of 18 benchmark datasets clearly showed that the process models that are generated through SM after the pre-processing step are simpler in terms of CFC. Besides, the overall size of the models is also found to be greatly reduced as shown in 17 out of 18 benchmark datasets. Moreover, the structuredness of the models has improved in 14 out of 18 benchmark datasets. These results point out



(a) Fitness values

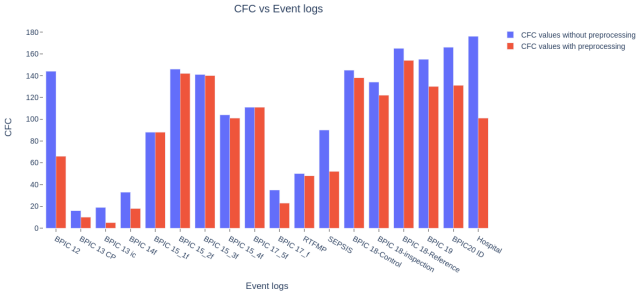


(b) Precision values

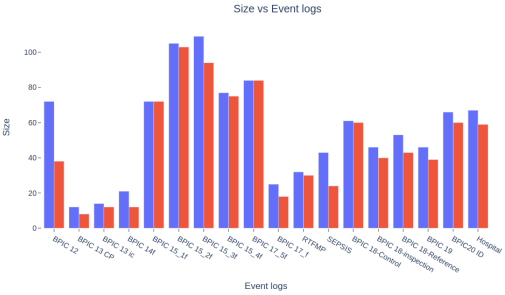


(c) F-Measure values

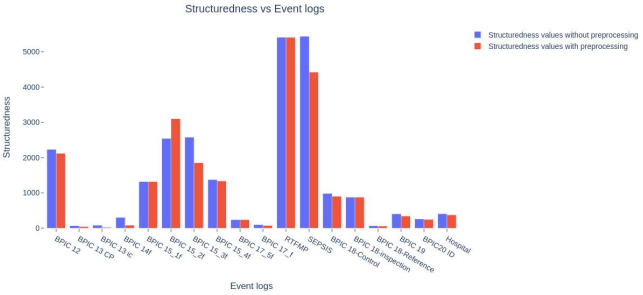
FIGURE 4. **Fitness, Precision and F-Measure values of discovered process model before and after pre-processing on the event logs.**



(a) CFC values



(b) Size values



(c) Structuredness values

FIGURE 5. **CFC, size values, and structuredness values of the discovered process model before and after pre-processing on the event logs.**

that our models are much more feasible to understand and effective in performance with less complexity as shown in Figures 5a,5b, and 5c.

When comparing P-values, 14 out of 18 datasets significantly improved for F-Measure and CFC; 12 out of 18 datasets were observed to have significant improvement

TABLE 5. Results of running time for the pre-processing step, process discovery time after applying pre-processing, total discovery time before and after applying pre-processing step.

Event logs	Pre-Processing Time (S)	Process Discovery Time with Pre-Processing	Total Discovery Time (S) with Pre-Processing	Total Discovery Time (S) without Pre-Processing
BPIC12	0.05	0.10	0.15	0.22
$BPIC13_{cp}$	0.02	0.009	0.029	0.01
$BPIC13_{inc}$	0.03	0.024	0.054	0.03
$BPIC14_{f}$	0.03	0.15	0.18	0.20
$BPIC15_{1f}$	0.009	0.12	0.129	0.15
$BPIC15_{2f}$	0.009	0.092	0.101	0.10
$BPIC15_{3f}$	0.02	0.06	0.08	0.09
$BPIC15_{4f}$	0.01	0.04	0.05	0.06
$BPIC17_{5f}$	0.007	0.08	0.087	0.09
$BPIC17_{f}$	0.02	0.8	0.82	1.00
RTFMP	0.08	0.10	0.18	0.20
SEPSIS	0.08	0.01	0.09	0.02
BPIC 18-control	0.04	0.20	0.24	0.25
BPIC 18-inspection	0.02	0.15	0.17	0.26
BPIC 18Reference	0.04	0.14	0.18	0.25
BPIC 19	0.03	0.16	0.19	0.26
BPIC 20 ID	0.02	0.05	0.07	0.09
Hospital	0.07	0.10	0.17	0.20

for size; and 13 out of 18 datasets were found to be critically improved in terms of structuredness. Hence, our proposed approach has shown statistical improvements on different evaluation metrics using 18 benchmark datasets.

Based on Table 2, the improvements in F-Measure and complexity resulted from decreasing/removing the infrequent traces which cause minimal scarification in fitness, on the other hand, it increases much in the precision values. In some event logs, our approach removed as much as 30% of the infrequent traces from the original event logs in order to reach the best value of the F-Measure. The percentage of the remaining traces in the event logs after pre-processing is shown in Table 2.

Another reason that our approach leads to the improvements of the F-Measure values and the complexity of the process models is that concatenating some concurrent relations, removes some of the unnecessary behaviors from the event logs. Table 2 shows that the numbers of distinct events after pre-processing steps are decreased which indicates that unnecessary behaviors are removed and caused significant improvements in F-Measure values and the complexity of the process models.

RTMF dataset contains high frequent behavior, whereas the SIPSIS dataset contains many distinct traces that happen only once. Also, both these datasets contain many concurrency and loops compared to other datasets. The algorithm was able to successfully eliminate concurrency of data and remove self-loops by effectively concatenating a large set of events that have concurrency relations. As a result, most infrequent behaviors, and infrequent traces in the event logs were removed. Therefore, the algorithm was able to facilitate the process discovery algorithm to generate a process model that has distinctly higher F-Measure values and lower complexity. To elucidate further, most of the BPIC15 datasets contain only fewer numbers of concurrency relations between events compared to other datasets. Even when the concurrency relation exists, the probability value of the first event follows the second event that exceeds the $p*$ threshold value of our algorithm. However, the probability of the second event follows the first event does not pass our p^* threshold value. As a result, our algorithm did not choose these combinations for concatenation. In fact, our algorithm selects combinations, which are having strong concurrent relations for concatenation. As a result, the F-Measure values for BPIC15 datasets either have improved slightly or have not changed compared to the results of using just the raw data. In terms of complexity, even though our algorithm made minor changes in these event logs, still the complexity of the process models have improved in most of the BPIC15 datasets.

Table 5 demonstrates the total discovery running time for which the proposed model was run, that is a summation of running time for our pre-processing step, and the running time for process discovery step after applying our approach. Also, it shows the total discovery running time without applying any pre-processing step. Comparing total discovery running time with applying the proposed approach and without applying the pre-processing step on 18 benchmark datasets, 14 of those were shown to run faster when the pre-processing step was applied. As a result, even though, our proposed approach contains a hyper-parameter optimization step, the total discovery time was improved. This could be due to removing some concurrency and self-loops through our algorithm from the logs, consequently, producing more efficient event logs.

TABLE 6. The evaluation metrics' results on 18 benchmark datasets after applying the proposed model and the 4 most recent approaches in the literature as the pre-processing steps, and the Wilcoxon test results on the improved cases to statistically identify the level of significance.

Approaches	F-Measure	P-value for F-Measure	CFC	P-value for CFC	Size	P-value for Size	Structuredness	P-value for Structured-ness
				BPIC12				
[28]	0.761	0.048	76	0.043	46	0.029	2340	0.045
[30]	0.781	NA	75	0.042	41	0.039	1113.1	NA
[29]	0.741	0.041	78	0.038	40	0.038	3001	0.036
[31]	0.751	0.042	79	0.038	45	0.031	2450	0.047
Proposed	0.778	NA	66	NA	38	NA	2120	NA
				$BPIC13_{CP}$				
[28]	0.882	0.043	24	0.045	25	0.034	54	0.042
[30]	0.871	0.042	20	0.046	23	0.041	48.5	0.049
[29]	0.852	0.039	30	0.039	28	0.032	49	0.049
[31]	0.878	0.043	36	0.034	25	0.034	74	0.032
Proposed	0.986	NA	10	NA	8	NA	44	NA
				$BPIC13_{ic}$				
[28]	0.874	0.034	32	0.033	33	0.038	39	0.041
[30]	0.891	0.036	12	0.046	45	0.032	25	0.031
[29]	0.895	0.039	14	0.043	32	0.041	43	0.022
[31]	0.894	0.037	10	0.047	23	0.042	32	0.032
Proposed	0.999	NA	5	NA	12	NA	20	NA
				$BPIC14_{f}$				
[28]	0.834	0.023	30	0.032	14	0.046	99	0.034
[30]	0.893	0.039	25	0.039	15	0.045	91	0.039
[29]	0.912	0.046	23	0.042	34	0.032	89	0.047
[31]	0.911	0.045	20	0.045	22	0.039	85	0.048
Proposed	0.937	NA	18	NA	12	NA	80	NA
				$BPIC15_{1f}$				
[28]	0.873	0.045	97	0.036	81	0.032	1450	0.031
[30]	0.856	0.043	95	0.038	79	0.038	1430	0.031
[29]	0.879	0.045	90	0.041	74	0.048	1500	0.029
[31]	0.882	0.049	100	0.035	78	0.037	1400	0.032
Proposed	0.887	NA	88	NA	72	NA	1320	NA
				$BPIC15_{2f}$				
[28]	0.811	0.046	153	0.048	189	0.032	3442	0.038
[30]	0.812	0.045	171	0.041	154	0.039	3382	0.039
[29]	0.801	0.041	184	0.039	120	0.041	3328	0.042
[31]	0.791	0.039	158	0.042	111	0.043	3281	0.045
Proposed	0.835	NA	142	NA	103	NA	3100	NA
				$BPIC15_{3f}$				
[28]	0.781	0.029	189	0.032	99	0.042	2100	0.042
[30]	0.784	0.039	181	0.039	198	0.036	1988	0.048
[29]	0.788	0.041	172	0.043	200	0.035	2300	0.038
[31]	0.801	0.048	162	0.049	103	0.039	2001	0.039
Proposed	0.814	NA	140	NA	94	NA	1854	NA
				$BPIC15_{4f}$				
[28]	0.797	0.036	190	0.029	85	0.029	1400	0.035
[30]	0.802	0.046	121	0.046	87	0.039	1470	0.027
[29]	0.791	0.035	151	0.034	86	0.038	1366	0.046
[31]	0.801	0.045	120	0.047	78	0.031	1400	0.035
Proposed	0.827	NA	101	NA	75	NA	1338	NA
				$BPIC17_{5f}$				
[28]	0.851	0.045	119	0.045	100	0.029	299	0.039
[30]	0.842	0.043	140	0.039	98	0.039	260	0.043
[29]	0.843	0.044	152	0.034	90	0.038	271	0.042
[31]	0.832	0.029	120	0.041	88	0.031	256	0.046
Proposed	0.871	NA	111	NA	84	NA	240	NA

TABLE 6. ***(Continued.)*** **The evaluation metrics' results on 18 benchmark datasets after applying the proposed model and the 4 most recent approaches in the literature as the pre-processing steps, and the Wilcoxon test results on the improved cases to statistically identify the level of significance.**

				BPIC17$_f$				
[28]	0.852	0.045	29	0.039	19	0.029	100	0.035
[30]	0.842	0.039	30	0.034	24	0.039	78	0.041
[29]	0.843	0.043	33	0.031	28	0.038	77	0.041
[31]	0.862	0.047	28	0.043	20	0.031	89	0.039
Proposed	0.885	NA	23	NA	18	NA	74	NA
				RTFMP				
[28]	0.792	0.042	78	0.041	56	0.029	6000	0.039
[30]	0.781	0.038	99	0.032	74	0.039	6792	0.031
[29]	0.778	0.034	90	0.038	76	0.038	5911	0.046
[31]	0.729	0.025	63	0.044	88	0.031	6700	0.035
Proposed	0.797	NA	48	NA	30	NA	5410	NA
				SEPSIS				
[28]	0.733	0.027	92	0.036	37	0.029	4444	0.049
[30]	0.752	0.036	82	0.037	48	0.039	5531	0.042
[29]	0.738	0.032	77	0.038	74	0.042	7351	0.037
[31]	0.761	0.042	73	0.044	88	0.031	6137	0.038
Proposed	0.770	NA	52	NA	24	NA	4423	NA
				BPIC 18-control				
[28]	0.809	0.031	198	0.033	73	0.039	988	0.025
[30]	0.805	0.028	193	0.035	74	0.038	967	0.027
[29]	0.812	0.034	145	0.044	78	0.034	943	0.032
[31]	0.824	0.037	154	0.031	88	0.031	928	0.033
Proposed	0.832	NA	138	NA	60	NA	903	NA
				BPIC 18-inspection				
[28]	0.852	0.039	184	0.039	58	0.045	905	0.035
[30]	0.845	0.036	183	0.041	81	0.034	901	0.041
[29]	0.877	0.045	138	0.043	63	0.038	909	0.032
[31]	0.887	0.047	190	0.037	76	0.037	900	0.043
Proposed	0.900	NA	122	NA	40	NA	874	NA
				BPIC 18-Reference				
[28]	0.0837	0.038	195	0.027	76	0.041	73	0.046
[30]	0.861	0.043	194	0.032	98	0.034	99	0.030
[29]	0.842	0.042	171	0.035	88	0.038	88	0.039
[31]	0.821	0.036	167	0.044	77	0.039	88	0.039
Proposed	0.872	NA	154	NA	43	NA	49	NA
				BPIC 19				
[28]	0.868	0.044	77	0.037	58	0.045	444	0.037
[30]	0.832	0.039	142	0.045	73	0.038	378	0.048
[29]	0.865	0.043	149	0.041	99	0.035	413	0.042
[31]	0.872	0.047	141	0.049	42	0.046	387	0.046
Proposed	0.881	NA	130	NA	39	NA	344	NA
				BPIC 20-ID				
[28]	0.823	0.029	183	0.035	100	0.038	432	0.033
[30]	0.842	0.045	193	0.033	99	0.041	423	0.037
[29]	0.831	0.036	163	0.044	69	0.045	328	0.038
[31]	0.851	0.046	176	0.039	87	0.042	300	0.047
Proposed	0.862	NA	131	NA	60	NA	245	NA
				Hospital				
[28]	0.822	0.031	194	0.035	100	0.034	733	0.032
[30]	0.842	0.042	163	0.040	88	0.039	421	0.039
[29]	0.821	0.028	129	0.042	76	0.044	403	0.048
[31]	0.832	0.038	172	0.036	99	0.036	405	0.046
Proposed	0.851	NA	101	NA	59	NA	376	NA

TABLE 7. Percentage of the distinct concurrent events on 18 benchmark datasets and its relationship with the evaluation metrics improvements by using different methodologies.

Approaches	Percentage of Distinct Concurrent Events	F-Measure Improvement	Improvement	CFC Improvement	Improvement	Size Improvement	Improvement	Structuredness Improvement	Improvement
				BPIC12					
[28]	11.3	7.4	Yes	68	Yes	26	Yes	-110	No
[30]	11.3	9.4	Yes	69	Yes	31	Yes	1116.9	Yes
[29]	11.3	5.4	Yes	66	Yes	32	Yes	-771	No
[31]	11.3	6.4	Yes	65	Yes	27	Yes	-220	No
Proposed	11.3	9.1	Yes	78	Yes	34	Yes	110	Yes
				$BPIC13_{CP}$					
[28]	34.7	-0.82	No	-8	No	-13	No	14	Yes
[30]	34.7	-9.3	No	-4	No	-11	No	19.5	Yes
[29]	34.7	-11.2	No	-14	No	-16	No	19	Yes
[31]	34.7	-8.6	No	-20	No	-13	No	-6	No
Proposed	34.7	2.2	Yes	6	Yes	4	Yes	24	Yes
				$BPIC13_{IC}$					
[28]	25.8	-7.7	No	-13	No	-19	No	41	Yes
[30]	25.8	-6	No	7	Yes	-31	No	55	Yes
[29]	25.8	-5.6	No	5	Yes	-13	No	37	Yes
[31]	25.8	-5.7	No	9	Yes	-9	No	48	Yes
Proposed	25.8	4.8	Yes	14	Yes	2	Yes	60	Yes
				$BPIC14_{f}$					
[28]	5.4	-3.2	No	3	Yes	7	Yes	205	Yes
[30]	5.4	2.7	Yes	8	Yes	6	Yes	213	Yes
[29]	5.4	4.6	Yes	10	Yes	-13	No	215	Yes
[31]	5.4	4.5	Yes	13	Yes	-1	No	219	Yes
Proposed	5.4	7.1	Yes	15	Yes	9	Yes	224	Yes
				$BPIC15_{1f}$					
[28]	25.1	-2.5	No	-7	No	-84	No	-898	No
[30]	25.1	-2.4	No	-25	No	-49	No	-838	No
[29]	25.1	-3.5	No	-38	No	-15	No	-784	No
[31]	25.1	-4.5	No	-12	No	-6	No	-737	No
Proposed	25.1	-0.1	No	4	Yes	2	Yes	-556	No
				$BPIC15_{3f}$					
[28]	12.6	-0.4	No	-48	No	10	Yes	478	Yes
[30]	12.6	-0.1	No	-40	No	-89	No	590	Yes
[29]	12.6	0.3	Yes	-31	No	-91	No	278	Yes
[31]	12.6	1.6	Yes	-21	No	6	yes	577	Yes
Proposed	12.6	2.9	Yes	1	Yes	15	Yes	724	Yes
				$BPIC15_{4f}$					
[28]	22.6	-3	No	-86	No	-8	No	-26	No
[30]	22.6	-2.5	No	-17	No	-10	No	-96	No
[29]	22.6	-3.6	No	-47	No	-9	No	8	Yes
[31]	22.6	-2.6	No	-16	No	-1	No	-26	No
Proposed	22.6	0	No	3	Yes	2	Yes	36	Yes
				$BPIC17_{5f}$					
[28]	32.4	-2	No	-8	No	-16	No	-79	No
[30]	32.4	-2.9	No	-29	No	-14	No	-20	No
[29]	32.4	-2.8	No	-41	No	-6	No	-31	No
[31]	32.4	-3.9	No	-9	No	-4	No	-160	No
Proposed	32.4	0	No	0	No	0	No	0	No
				BPIC 17_{f}					
[28]	13.2	-2.4	No	6	Yes	6	Yes	0	No
[30]	13.2	-3.4	No	5	Yes	1	yes	22	Yes
[29]	13.2	-3.3	No	2	Yes	-3	No	23	Yes
[31]	13.2	-1.4	No	7	Yes	5	Yes	11	Yes
Proposed	13.2	0.9	Yes	12	Yes	7	Yes	26	Yes

TABLE 7. ***(Continued.)*** **Percentage of the distinct concurrent events on 18 benchmark datasets and its relationship with the evaluation metrics improvements by using different methodologies.**

RTFMP									
[28]	37.3	15.7	Yes	-28	No	-24	No	-588	No
[30]	37.3	14.6	Yes	-49	No	-42	No	-1380	No
[29]	37.3	14.3	yes	-40	No	-44	No	-499	No
[31]	37.3	9.4	Yes	-13	No	-56	No	-1288	No
Proposed	37.3	16.2	Yes	2	Yes	2	Yes	2	Yes
SEPSIS									
[28]	23.1	7.4	Yes	-2	No	6	Yes	992	Yes
[30]	23.1	9.3	Yes	8	Yes	-5	No	-95	No
[29]	23.1	7.9	Yes	13	Yes	-31	Yes	-1915	No
[31]	23.1	10.2	Yes	17	Yes	-45	No	-7.1	No
Proposed	23.1	11.1	Yes	38	Yes	19	Yes	1013	Yes
BPIC 18-control									
[28]	26.4	-0.1	No	-53	No	12	No	-6	No
[30]	26.4	-0.5	No	-48	No	-13	No	15	Yes
[29]	26.4	0.2	Yes	0	No	-17	No	39	Yes
[31]	26.4	1.4	Yes	-9	No	27	Yes	44	Yes
Proposed	26.4	2.2	Yes	7	Yes	1	Yes	79	Yes
BPIC 18- inspection									
[28]	31.7	-2.1	No	-50	No	-12	No	-27	No
[30]	31.7	-2.8	No	-49	No	-35	No	-23	No
[29]	31.7	0.4	Yes	-4	No	-17	No	-31	No
[31]	31.7	1.4	Yes	-56	No	-30	No	-22	No
Proposed	31.7	2. 7	Yes	12	Yes	6	Yes	4	Yes
BPIC 18- Reference									
[28]	42.5	-1.6	No	-30	No	-23	No	-6	No
[30]	42.5	0.8	Yes	-29	No	-45	No	-32	No
[29]	42.5	-1.1	No	-6	No	-35	No	-21	No
[31]	42.5	-3.2	No	-2	No	-24	No	-31	No
Proposed	42.5	1.9	Yes	11	Yes	10	Yes	8	Yes
BPIC 19									
[28]	12.8	1.7	Yes	-22	No	-12	No	-40	No
[30]	12.8	-1.9	No	13	Yes	-27	No	26	Yes
[29]	12.8	1.4	Yes	6	Yes	-53	No	-9	No
[31]	12.8	2.1	yes	14	Yes	4	Yes	17	Yes
Proposed	12.8	3	Yes	25	Yes	7	Yes	60	Yes
BPIC 20- ID									
[28]	33.5	-1.7	No	-17	No	-34	No	-172	No
[30]	33.5	0.2	Yes	-27	No	-33	No	-163	No
[29]	33.5	-0.9	No	3	Yes	-3	No	-68	No
[31]	33.5	1.1	Yes	-10	No	-21	No	-40	No
Proposed	33.5	2.2	Yes	35	Yes	6	Yes	15	Yes
Hospital									
[28]	13.7	-1.4	No	-18	No	-33	No	-324	No
[30]	13.7	1.6	Yes	13	Yes	-21	No	-12	No
[29]	13.7	-0.5	No	47	Yes	-9	No	6	Yes
[31]	13.7	0.6	Yes	4	Yes	-32	No	4	Yes
Proposed	13.7	2.5	Yes	75	Yes	8	No	33	Yes

Moreover, we ran experiments to compare the results of our methodology with the best 4 recent pre-processing steps in the existing literature. To do so, we used the threshold of 0.7 for our proposed approach, and the thresholds of 0.4 and 0.4 for η and ϵ respectively for the SM algorithm. For the existing approaches, first, we applied their algorithms on

the logs, and the resultant logs were then fed to the SM algorithm with the thresholds of 0.4 and 0.4 for η and ϵ respectively to discover the process model. Similar to what was mentioned above, when we compare the proposed model with the mentioned existing approaches result, in the cases in which improvements were perceived in any of the evaluation metrics, the Wilcoxon Test was run to measure the level of significance. Comparing P-values in F-Measure, 17 out of 18 datasets were shown to be significantly improved as compared to other approaches. Additionally, in terms of CFC, size, and structuredness, the p-value in all datasets had critical improvements as shown in Table 6. Hence, the results indicate that our proposed approach statistically outperformed the best 4 recent pre-processing steps.

Table 7 indicates that the results of the proposed approach as compared with the results of the best 4 recent approaches in the literature, improved not only on the datasets which hold fewer concurrency between events, but also on those that contain more concurrency. Table 7 is constructed to find out if there is any relation between the percentage of existing distinct concurrent events on each dataset, and the improvements of each evaluation metrics on proposed model and the current four best existing methods in the literature. As it can be observed, the existing pre-processing methods are useful to generate a more accurate and simpler model when an event log exhibits fewer concurrency between events. However, in cases where the event log contains many concurrent relations between events, these techniques would not improve the results. Since most of the real-life processes contain concurrency between events and loops, we believe that our approach can be more useful on the real life datasets as compared to other approaches. For instance, for the RTFMP and the BPIC-cp datasets in which the percentage of the concurrent events were higher, other methods had no to negligible improvements, but for the datasets like BPIC14-f and BPIC12 that had the lower percentage of concurrency, other methods had improvements. However, our method showed improvements in almost all evaluation metrics regardless of the percentage of concurrency in the event logs.

VI. CONCLUSION

Process mining provides insights into the real processes happening in real life. Most process discovery algorithms are designated to model event logs, which are assumed they are clean. However, real-life processes are mostly noisy, contain infrequent behaviors, many concurrency, and loops. As a result, the generated process models are mostly inaccurate and largely complex. However, adding a pre-processing step before the process discovery leads to essentially removing these issues to visualize the data more clearly and drastically improving the performance of the process discovery algorithms. Here, we proposed a pre-processing technique to increase the performance of the process discovery algorithms by recommending a probability-based concatenation of events, which hold concurrent relations.

For evaluation of the process models, first 18 real-life benchmark datasets were used and fed to the SM algorithm since it is the utmost discovery algorithm in terms of F-Measure and complexity for the process discovery to discover the process models. On the other hand, our pre-processing step was applied on the same event logs, and the resultant event logs were used SM algorithm to discover the process models. Next, we measured the F-Measure and complexity of the models in both cases. On the other hand, the results of our proposed model were compared with the best 4 recent pre-processing steps in the existing literature. In the cases in which improvements were observed in any of the evaluation metrics, the Wilcoxon Test was run to measure the level of significance. The results indicate that the pre-processing step increases the F-Measure values and decreases the complexity of the models statistically by concatenating some of the unnecessary concurrent events and removing self-loops.

In future work, we aim to incorporate other perspectives such as what pre-processing approaches can be developed in order to streamline and improve our results by incorporating different prevailing pre-processing strategies. Finding the optimal values of thresholds and speeding up the process is challenging, and time-consuming. Consequently, we have a plan to find techniques to automatically adjust the optimal values of the parameters than the hyper-parameter optimization we currently favor. Also, we concatenated the events which hold concurrent relations, so we aim to test the effects of concatenating other events which hold other relations such as causality and conflict relations.

VII. LIST OF SYMBOLS

E	set of events
T	set of traces
L	event logs
L	mutli-set of labels
ℓ	label of each event
λ	function that retrieves the label of each event
p	set of places in a PN
T_i	set of transition in a PN
F	set of directed arcs in a PN
π	a function, which maps a transition to either a single observable event or to the non-observable event
$\perp$	non- observable event
U	set of activities
p^*	threshold value for concatenation algorithm
η	SM threshold which controls concurrency relations
ϵ	SM threshold which controls filtering process
$f(e_i)$	the frequency of e_i
$f_{(e_i e_j)}$	frequency of e_j which occurs immediately after event e_i
$F^s_{e_i e_j}$	s^{th} frequency of the indices for the combination (e_i, e_j)
$I^s e_i$	s^{th} index of event e_i in an event log
$(e_i \oplus e_j)$	concatenated event for (e_i, e_j)

$f(e_i \rightarrow e_j)$
: the frequency that e_j happens instantly after e_i

$f(e_j \rightarrow e_i)$
: the frequency that e_i happens instantly after e_j

$P(e_i \rightarrow e_j)$
: the probability that e_j happens instantly after e_i

$P(e_j \rightarrow e_i)$
: the probability that e_i happens instantly after e_j

$P(e_i||e_j)$
: probability of the frequency that event e_j which occurs immediately after event e_i

$P(e_j||e_i)$
: probability of the frequency that event e_i which occurs immediately after event e_j

ACKNOWLEDGMENT

Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the Centers for Disease Control and Prevention or the Department of Health and Human Services.

REFERENCES

[1] W. van der Aalst, "Data science in action," in *Process Mining: Data Science in Action*, W. van der Aalst, Ed., Berlin, Germany: Springer, 2016, pp. 3–23.

[2] J. Theis and H. Darabi, "Adversarial system variant approximation to quantify process model generalization," *IEEE Access*, vol. 8, pp. 194410–194427, 2020, doi: 10.1109/ACCESS.2020.3033450.

[3] M. Pishgar, M. Razo, J. Theis, and H. Darabi, "Process mining model to predict mortality in paralytic ileus patients," in *Proc. Int. Conf. Cyber-Phys. Social Intell. (ICCSI)*, Dec. 2021, pp. 1–6.

[4] J. T. M. Pishgar, M. D. Rio, A. Ardati, H. Anahideh, and H. Darabi, "Prediction of unplanned 30-day readmission for ICU patients with heart failure," *BMC Med. Inform. Decis. Making*, vol. 22, no. 1, pp. 1–12, 2022, doi: 10.1186/s12911-022-01857-y.

[5] H. AlQaheri and M. Panda, "An education process mining framework: Unveiling meaningful information for understanding students' learning behavior and improving teaching quality," *Information*, vol. 13, no. 1, p. 29, Jan. 2022, doi: 10.3390/info13010029.

[6] S. Suriadi, M. T. Wynn, C. Ouyang, A. H. M. ter Hofstede, and N. J. van Dijk, "Understanding process behaviours in a large insurance company in Australia: A case study," in *Advanced Information Systems Engineering*, C. Salinesi, M. C. Norrie, Ó. Pastor, Eds. Berlin, Germany: Springer, 2013, pp. 449–464.

[7] J. Theis, I. Mokhtarian, and H. Darabi, "Process mining of programmable logic controllers: Input/output event logs," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 216–221, doi: 10.1109/COASE.2019.8842900.

[8] W. M. P. van der Aalst, "Petri nets," in *Encyclopedia of Database Systems*, L. Liu and M. T. ÖZsu, Eds. Boston, MA, USA: Springer, 2009, pp. 2103–2108.

[9] W. M. P. van der Aalst, "Business process modeling notation," in *Encyclopedia Database Systems*, L. Liu and M. T. ÖZsu, Eds. Boston, MA, USA: Springer, 2009, pp. 293–294.

[10] J. Mendling, "Event-driven process chains (EPC)," in *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, J. Mendling, Ed. Berlin, Germany: Springer, 2008, pp. 17–57.

[11] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Causal nets: A modeling language tailored towards process discovery," in *CONCUR 2011 Concurrency Theory*, J.-P. Katoen and B. König, Eds. Berlin, Germany: Springer, 2011, pp. 28–42.

[12] B. Mikolajczak and J.-L. Chen, "Workflow mining alpha algorithm—A complexity study," in *Intelligent Information Processing and Web Mining*, M. A. Klopotek, S. T. Wierzchon, and K. Trojanowski, Eds. Berlin, Germany: Springer, 2005, pp. 451–455.

[13] A. J. M. M. Weijters and J. T. S. Ribeiro, "Flexible heuristics miner (FHM)," in *Proc. IEEE Symp. Comput. Intell. Data Mining (CIDM)*, Apr. 2011, pp. 310–317, doi: 10.1109/CIDM.2011.5949453.

[14] S. K. L. M. vanden Broucke and J. De Weerdt, "Fodina: A robust and flexible heuristic process discovery technique," *Decis. Support Syst.*, vol. 100, pp. 109–118, Aug. 2017, doi: 10.1016/j.dss.2017.04.005.

[15] S. J. Leemans, D. Fahland, and W. M. Van Der Aalst, "Discovering block-structured process models from event logs—A constructive approach," in *Proc. Int. Conf. Appl. Theory Petri Nets Concurrency*. Cham, Switzerland: Springer, 2013, pp. 311–329.

[16] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, and A. Polyvyanyy, "Split miner: Automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, no. 2, pp. 251–284, 2019, doi: 10.1007/s10115-018-1214-x.

[17] R. P. J. C. Bose and W. M. P. V. D. Aalst, "Analysis of patient treatment procedures: The BPI Challenge case study," BPM Center, Germany, BPM Rep., 2011, vol. 1118.

[18] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: A two-step approach to balance between underfitting and overfitting," *Softw. Syst. Model.*, vol. 9, no. 1, p. 87, 2008, doi: 10.1007/s10270-008-0106-z.

[19] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," in *Business Process Management*, G. Alonso, P. Dadam, and M. Rosemann, Eds. Berlin, Germany: Springer, pp. 328–343.

[20] A. Pinto, R. A. Ribeiro, and I. L. Nunes, "Fuzzy approach for reducing subjectivity in estimating occupational accident severity," *Accident Anal. Prevention*, vol. 45, pp. 90–281, Mar. 2012, doi: 10.1016/j.aap.2011.07.015.

[21] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The ProM framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005*, G. Ciardo and P. Darondeau, Eds. Berlin, Germany: Springer, 2005, pp. 444–454.

[22] L. Ghionna, G. Greco, A. Guzzo, and L. Pontieri, "Outlier detection techniques for process mining applications," in *Foundations of Intelligent Systems*, A. An, S. Matwin, Z. W. Ras, and D. Slezak, Eds. Berlin, Germany: Springer, 2008, pp. 150–159.

[23] X. Lu, D. Fahland, and W. M. P. van der Aalst, "Conformance checking based on partially ordered event data," in *Proc. Bus. Process Manage. Workshops*, F. Fournier J. Mendling, Eds. Cham, Switzerland: Springer, 2015, pp. 75–88.

[24] 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, doi: 10.1016/j.dss.2015.08.003.

[25] R. Conforti, M. L. Rosa, and A. H. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Sep. 2017, doi: 10.1109/TKDE.2016.2614680.

[26] M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst, "Improving process discovery results by filtering outliers using conditional behavioural probabilities," in *Proc. Bus. Process Manage. Workshops*, E. Teniente and M. Weidlich, Eds. Cham, Switzerland: Springer, 2018, pp. 216–229.

[27] S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn, "Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs," *Inf. Syst.*, vol. 64, pp. 132–150, Mar. 2017, doi: 10.1016/j.is.2016.07.011.

[28] N. Tax, N. Sidorova, and W. M. P. van der Aalst, "Discovering more precise process models from event logs by filtering out chaotic activities," *J. Intell. Inf. Syst.*, vol. 52, no. 1, pp. 107–139, Feb. 2019, doi: 10.1007/s10844-018-0507-6.

[29] D. A. Fischer, K. Goel, R. Andrews, C. G. J. van Dun, M. T. Wynn, and M. Röglinger, "Enhancing event log quality: Detecting and quantifying timestamp imperfections," in *Proc. Int. Conf. Bus. Process Manage.*, vol. 392. Seville, Spain: Springer, Sep. 2020, pp. 309–326.

[30] M. Sani, S. van Zelst, and A. Van Der, "Improving the performance of process discovery algorithms by instance selection," *Comput. Sci. Inf. Syst.*, vol. 17, no. 3, pp. 927–958, 2020.

[31] D. A. Fischer, K. Goel, R. Andrews, C. van Dun, M. Wynn, and M. Röglinger, "Enhancing event log quality: Detecting and quantifying timestamp imperfections," in *Proc. 18th Int. Conf. Bus. Process Manage. (BPM)*, vol. 392. Seville, Spain: Springer, Sep. 2020, pp. 309–326.

[32] J. De Weerdt, M. D. Backer, J. Vanthienen, and B. Baesens, "A robust F-measure for evaluating discovered process models," in *Proc. IEEE Symp. Comput. Intell. Data Mining (CIDM)*, Apr. 2011, pp. 148–155, doi: 10.1109/CIDM.2011.5949428.
[33] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Inf. Syst.*, vol. 37, no. 7, pp. 654–676, 2012, doi: 10.1016/j.is.2012.02.004.
[34] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. IEEE 15th Int. Enterprise Distrib. Object Comput. Conf.*, Aug. 2011, pp. 55–64, doi: 10.1109/EDOC.2011.12.
[35] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Measuring precision of modeled behavior," *Inf. Syst. e-Bus. Manage.*, vol. 13, no. 1, pp. 37–67, Feb. 2015, doi: 10.1007/s10257-014-0234-7.
[36] J. Mendling, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Cham, Switzerland: Springer, 2008.
[37] J. Cardoso, "Business process control-flow complexity: Metric, evaluation, and validation," *Int. J. Web Services Res.*, vol. 5, no. 2, pp. 49–76, Apr. 2008.
[38] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7PMG)," *Inf. Softw. Technol.*, vol. 52, pp. 127–136, Feb. 2010, doi: 10.1016/j.infsof.2009.08.004.
[39] R. Confort. *BPMN Miner: Automated Discovery of BPMN Process Models with Hierarchical Structure*. Accessed: Aug. 1, 2022. [Online]. Available: https://github.com/raffaeleconforti/ResearchCode
[40] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Inf. Syst.*, vol. 37, no. 7, pp. 654–676, Nov. 2012, doi: 10.1016/j.is.2012.02.004.
[41] R. Conforti, M. L. Rosa, and A. H. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Sep. 2017, doi: 10.1109/TKDE.2016.2614680.
[42] W. M. van der Aalst, A. Bolt, and S. J. van Zelst, "RapidProM: Mine your processes and not just your data," 2018, *arXiv:1703.03740*.
[43] R. Conforti, M. Dumas, L. García-Bañuelos, and M. L. Rosa, "Beyond tasks and gateways: Discovering BPMN models with subprocesses, boundary events and activity markers," in *Business Process Management*, S. Sadiq, P. Soffer, and H. Völzer, Eds. Cham, Switzerland: Springer, 2014, pp. 101–117.

MARYAM PISHGAR (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in industrial engineering and management system from the University of Illinois at Chicago (UIC), in 2020, where she is currently pursuing the Ph.D. degree in industrial engineering and operation research with the Department of Mechanical and Industrial Engineering. Her current research interests include process mining, process discovery, and its application in health care, machine learning, deep learning, and application of artificial intelligence in occupational safety and health.

MARTHA RAZO received the B.S. and M.S. degrees in applied mathematics from the Illinois Institute of Technology, in 2017. She is currently pursuing the Ph.D. degree in industrial engineering and operation research with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago (UIC). Her current research interests include process mining, deep learning, inventory analysis, sales forecasting, and big data.

HOUSHANG DARABI (Senior Member, IEEE) received the Ph.D. degree in industrial and systems engineering from Rutgers University, New Brunswick, NJ, USA, in 2000. He is currently a Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago (UIC). His research has been supported by several agencies, such as the National Science Foundation, the National Institute of Standard and Technology, and the National Institute of Occupational Safety and Health. His current research interests include the application of data mining, process mining, and time series classification in design and analysis of healthcare, safety, and education systems.