Improving Process Discovery Results by Filtering Out Outliers from Event Logs with Hidden Markov Models

Zhenyu Zhang[1,2], Ryan Hildebrant[1], Fatemeh Asgarinejad[1], Nalini Venkatasubramanian[2], Shangping Ren[1]
[1]Department of Computer Science, San Diego State University, San Diego, CA 92182, USA
[2]Department of Computer Science, University of California, Irvine, CA 92697, USA
zzhang4430@sdsu.edu, rhildebrant@sdsu.edu, fasgarinejad2881@sdsu.edu, nalini@ics.uci.edu, sren@sdsu.edu

***Abstract*—Process Mining is a technique for extracting process models from event logs. Event logs contain abundant explicit information related to events, such as the timestamp and the actions that trigger the event. Much of the existing process mining research has focused on discovering the process models behind these event logs. However, Process Mining relies on the assumption that these event logs contain accurate representations of an ideal set of processes. These ideal sets of processes imply that the information contained within the log represents what is really happening in a given environment. However, many of these event logs might contain noisy, infrequent, missing, or false process information that are generally classified as outliers. Extending beyond process discovery, there are many research efforts towards cleaning the event logs to deal with these outliers. In this paper, we present an approach that uses hidden Markov models to filter out outliers from event logs prior to applying any process discovery algorithms. Our proposed filtering approach can detect outlier behavior, and consequently, help process discovery algorithms return models that better reflect the real processes within an organization. Furthermore, we show that this filtering method outperforms two commonly used filtering approaches, namely the *Matrix Filter* approach and the *Anomaly Free Automation* approach for both artificial event logs and real-life event logs.**

I. Introduction

Over the last decade, process mining has emerged as a new research field that uses available data, such as event logs, to understand how processes are being executed in real life. Given an event log, process mining aims to extract process knowledge (e.g., process models) and provide valuable insights to help better understand, monitor, and improve the current processes [1]. Process mining has been applied successfully in many application domains, such as the banking sector, the insurance industry, e-government communications, and medical field.

The starting point for process mining is an event log. An event log contains information about a working process as it takes place. Each event in the log reflects an activity that can be represented as a well-defined step in the process and is related to a particular trace of a process instance. An event log may also contain other information related to the event such as the executor of the event, the metadata associated to an event, and other related data attributes. Using these attributes of an event log, we can represent a process model that is useful to a stakeholder. However, in practice, process mining can have many challenges. One of the most challenging problems in process mining is concerned with the derivation of accurate process models from event logs.

Numerous discovery algorithms including the Alpha algorithm [2], the heuristic algorithm [3] and region-based approaches [4], [5] claim to address this problem thorough utilizing different trade-offs between the degree to which they accurately capture the behavior recorded in an event log and the complexity of the derived process model [6].

Many process mining algorithms are based on the assumption that an event log accurately represents information about a working process as it takes place. Unfortunately, many real-life process event logs often contain *noise* and *infrequent behavior*. *Noise* refers to inserting erroneous activities in the log, not logging some activities that have occurred, or reporting some activities with an out-of-order time sequence [7], [8].

The presence of noise can result from data entry problems, faulty data collection instruments, data transmission, streaming problems, or other technical limitations. However, *infrequent behaviour* refers to a behaviour that only occurs in exceptional case within the process. Without having business or domain knowledge, distinguishing between noise and infrequent behaviour is a challenging task [9]. Therefore, we consider this as a separate research question and do not cover such cases in this paper. As such, we consider both noise and infrequent behaviour to be *outliers*.

The presence of outliers will lead to infrequent paths within the derived model. This causes the process model to become cluttered and results in a model that is simply not an accurate representation of the actual behavior. In order to limit these adverse effects, event logs are typically subjected to a pre-processing phase where they are manually cleaned from outliers [6]. However, this is a challenging and time-consuming task, with no guarantee on the effectiveness of the resulting model, especially in the context of large event logs exhibiting complex process behavior [10].

The inability to effectively detect and filter out outliers adversely affects the quality of the discovered model. In particular, the discovered model's precision can be greatly affected. A frequently used performance metric is precision that shows the degree to which a model allows for unobserved behavior in the log. In some cases, low levels of outliers will have a detrimental effect on the quality of the models that are produced by various discovery algorithms. The Heuristics Miner [3], Fodina [11], and the Inductive Miner [12] algorithms claim to have noise-tolerant capabilities. However, they result in low-quality models in the presence of low-levels of outliers. For instance, the Heuristics Miner can have a 49% drop in precision when the amount of outliers corresponds to just 2% of the total log size [13].

In this paper, we present an approach that uses Hidden Markov Models to filter out outliers from event logs prior to applying any process discovery algorithms. Below we provide an outline of the approach and an overview of the four steps required. First, we obtain the mainstream process model by applying an existing process discovery algorithm on process mainstream behaviors across the entire event log. For a given event log, the process mainstream behaviors imply either a set of frequently occurring event traces or a set of event traces that cover all of the frequently occurring activities. Next, we replay the original event log using the mainstream process model and obtain a mainstream sublog. The mainstream sublog shows us where the event traces fit within the mainstream process model. For the third step, we use the mainstream process model and the mainstream sublog to construct a hidden Markov model. Based on the hidden Markov model , we calculate the probability of the occurrence of each event trace, which is recorded in the original event log but not in the mainstream sublog, and compare it with a user-defined threshold. We define the event trace as an outlier if the probability of the occurrence of an event trace derived from the hidden Markov model is lower than the threshold. Finally, we remove the outliers from the original event logs to improve process discovery results.

The approach has been implemented on top of the *pm4py* Framework [14], and is evaluated by using the combination of two commonly used filtering approaches: the *Matrix Filter* approach [15] and the *Anomaly Free Automation* approach [13]. We measure the effectiveness of the proposed approach when using artificial and real-life event logs. The results of our experiments show that our approach adequately identifies and removes outliers, leading to an increase in the process discovery results. We measure these quality metrics in terms of fitness, precision, and F1-score. Additionally, we compare our process discovery results under different model complexity levels against the commonly used filtering approaches in terms of behavior recall, behavior precision, structural recall, and structural precision. The results of these experiments show that our approach outperforms two commonly used filtering approaches, namely the *Matrix Filter* approach and the *Anomaly Free Automation* approach.

The paper is organized as follows: Section II introduces definitions used in the rest of the paper. Next, we develop an approach to identify mainstream behaviors of the event log in Section III. Section IV presents the proposed approach to construct the hidden Markov model to filter out outliers from event logs. We evaluate the proposed outlier filtering technique in terms of four criteria mentioned above using artificial and real-life even logs and discuss our findings in Section V. Section VI discusses the related work and provides an overview of process mining algorithms with a focus on their noise tolerance capabilities. Lastly, Section VII concludes the paper and discusses future work.

II. Notations and Definitions

In this section, we first formalize notations and definitions related to event logs used in this paper, and then introduce the process model in terms of a Petri net. Finally, we introduce the definitions related to *Hidden Markov Models*.

A. Event logs

The starting point for process mining is an event log. An event log records information about activities as they take place. For this paper, we adopt definitions that are similar to the ones given in [16]. In particular, for a given activity set Σ, an event entry e in an event log records an activity happening within the operation of a process. An event trace σ is a finite sequence of event entries that are ordered by their occurrence time. An event log L consists of a set of event traces. The formal definitions of an event entry, an event trace, and an event log are given below.

Definition 1 (Event Entry). *Given an activity set Σ, an event entry e is a tuple (α, τ), where $\alpha \in \Sigma$ is an activity and τ is the timestamp of α. Labeling functions act: $e \mapsto \alpha$ is used to get the activity of the event entry.* □

Definition 2 (Event Trace). *An event trace σ is a finite sequence of event entries $[e_1, \dots, e_i, \dots, e_n]$ where $e_i = (\alpha_i, \tau_i)$.* □

Definition 3 (Event Log). *An event log L is a set of event traces $\{\sigma\}$. $|L|$ denotes the cardinality of L.* □

For an illustrative purpose, we consider a simplified event log as shown in Table I. This event log L contains information about five traces, i.e., $L = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$. Note that the ordering of activities within a trace is relevant, while the ordering of activities among different traces is of no importance. The log shows that for **trace** 1, the activities A, B, C, D were executed, i.e., $\sigma_1 = [e_1, e_2, e_3, e_4]$ where $e_1 = (A, "08:15")$, $e_2 = (B, "10:24")$, $e_3 = (C, "10:25")$, and $e_4 = (D, "13:19")$. Due to the page limit, we omit the representations of the **trace** 2, 3, 4, 5, and 6 which are similar to the representation of **trace** 1. Each trace starts with the execution of activity A and ends with activity D. We also observe that if activity C is executed within a trace, activity B is also executed. However, for some traces, activity C is executed before activity B, while for some others, it is the other way around. The activity of event entries in a event trace is obtained a by labeling function. For instance, the activity of e_1 in **trace** 1 is A, i.e., *act*$(e_1) = A$.

TABLE I
An example of event logs

Trace Identifier	Activity	Timestamp
Trace 1	A	08:15
Trace 2	A	08:24
Trace 3	A	09:30
Trace 1	B	10:24
Trace 3	B	10:24
Trace 2	C	10:26
Trace 1	C	10:25
Trace 4	A	11:45
Trace 2	B	11:46
Trace 2	D	12:23
Trace 5	A	13:14
Trace 4	C	13:17
Trace 1	D	13:19
Trace 6	A	13:39
Trace 3	C	14:09
Trace 6	B	14:19
Trace 3	D	14:29
Trace 4	B	14:43
Trace 5	E	15:22
Trace 6	C	15:29
Trace 5	D	15:45
Trace 4	D	16:10
Trace 6	D	16:43

B. Petri net

The majority of process mining algorithms [17], [18], [5] use a Petri net[4] to represent process models.

Definition 4 (Petri net [19]). *A Petri net N is a tuple (P, T, F), where*

- *P is a finite set of places;*
- *T is a finite set of transitions, $P \cap T = \emptyset$; and*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs.* □

Given a Petri net $N = (P, T, F)$ and a place $p \in P$, we use notation $\bullet p$ to represent a set of transitions taking place immediately before a place p, i.e., $\bullet p = \{t | t \in T \wedge (t, p) \in F\}$, we also call $\bullet p$ the pre-set of p. Notation $p\bullet$ to represent a set of transitions immediately after place p, i.e., $p\bullet = \{t | t \in T \wedge (p, t) \in F\}$, $p\bullet$ is also called the post-set of p.

The state of a Petri net $N = (P, T, F)$ is represented by its markings which is a distribution of tokens on places P. A marking of N is defined by a mapping function $m : P \rightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. A place p is *marked* by a marking m if $m(p) > 0$.

The execution semantics of a Petri net $N = (P, T, F)$ are defined by transition firings which specify the enabling conditions and the marking transformations of the Petri net. A transition $t \in T$ is enabled by a marking m, if m marks all places in its pre-set $\bullet t$, i.e., $\forall p \in \bullet t : m(p) > 0$. In a Petri net $N = (P, T, F)$, with transition $t \in T$, and marking m, $(N, m) \xrightarrow{t}$ denotes that transition t is enabled at the marking m under the Petri net N. The firing of an enabled transition t transforms the marking m to m' as below:

$$m'(p) = \begin{cases} m(p) - 1 & \texttt{if} \quad p \in \bullet t \wedge p \notin t\bullet \\ m(p) + 1 & \texttt{if} \quad p \notin \bullet t \wedge p \in t\bullet \\ m(p) & \texttt{otherwise} \end{cases} \tag{1}$$

Given a Petri net $N = (P, T, F)$, a transition $t \in T$, and two markings m, m', according to formula 1, $(N, m) \xrightarrow{t} (N, m')$ denotes that the enabled transition t results in marking m'. We use the similar way to represent the firing sequence of the transitions $t_1, \dots, t_n$, i.e.,

$(N, m^{start}) \xrightarrow{t_1}\xrightarrow{t_2} \ldots \xrightarrow{t_n} (N, m^{end})$, where m^{start} is the initial marking, m^{end} is the new marking derived by firing the sequence of $t_1, \ldots, t_n$.

To explain on the definitions listed above, we use the Petri net shown in Fig. 1 to provide a more concrete example. In the graphical representation, *places*, *transitions*, *arcs*, and *tokens* are represented by circles, squares, arrows, and black dots respectively. The Petri net N shown in Fig. 1 is defined as (P, T, F), where $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2\}$, and $F = \{(t_1, p_1), (p_1, t_2), (t_2, p_2), (t_2, p_3), (t_2, p_4), (p_2, t_1)\}$. The current marking is $m = \{(p_1, 3), (p_2, 0), (p_3, 0), (p_4, 1)\}$. Hence, the transition t_2 is enabled by the marking m denoted by $(N, m) \xrightarrow{t_2}$. According to formula (1), the firing of the transition t_2 transforms the marking m to the new marking m', i.e., $(N, m) \xrightarrow{t_2} (N, m')$, where $m' = \{(p_1, 2), (p_2, 1), (p_3, 1), (p_4, 2)\}$. Note that, for a marking m, if a place does not contain any token, i.e., $(p, 0)$, we omit it for simplicity. For instance, we can simplify $m = \{(p_1, 3), (p_2, 0), (p_3, 0), (p_4, 1)\}$ to $m = \{(p_1, 3), (p_4, 1)\}$.

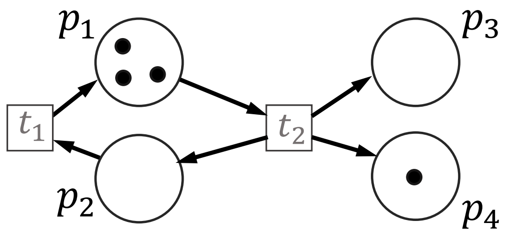


Fig. 1. A Petri Net Example

Note that the process models that are mined by most of the existing algorithms [5], [18], [17] are subclasses of the Petri net. One such sublcass known as a *workflow net* expands the Petri net by introducing three further constraints: (1) there is one and only one *input place* where a process starts, (2) there is one and only one *output place* where the process ends, and (3) all elements are on a path from the input place to the output place. The formal definition is given below.

Definition 5 (*Workflow net* [20]). *A net $N = (P, T, F)$ is a workflow net, if it is a Petri net and satisfies the following constraints:*

- *There is one and only one input place i, i.e.*
  1) $\exists i \in P$, s.t. $\forall t \in T, (t, i) \notin F$,
  2) $\exists i_1, i_2 \in P$, $(\forall t \in T, (t, i_1) \notin F \wedge (t, i_2) \notin F) \rightarrow i_1 = i_2$.
- *There is one and only one output place o, i.e.*
  1) $\exists o \in P$, s.t. $\forall t \in T, (o, t) \notin F$,
  2) $\exists o_1, o_2 \in P$, $(\forall t \in T, (o_1, t) \notin F \wedge (o_2, t) \notin F) \rightarrow o_1 = o_2$
- *For a pseudo transition $\xi \notin T$, the net $(P, T \cup \{\xi\}, F \cup \{((o, \xi), (\xi, i)\})$ is a strongly connected net.* □

It is worth pointing out that the three constraints do not change neither the syntax or the execution semantics of a Petri net. Since our work is based on the workflow nets that are derived from existing process mining algorithms, the process model refers to the workflow net in the later sections of this paper.

C. Hidden Markov Models

A hidden Markov model is an extension of a discrete Markov process, which is a combination of two probability distribution processes wherein one of them is hidden. A hidden process can only be determined through another process that produces a sequence of observations [21]. Each observation depends on the states in a hidden process. The hidden Markov model can also be interpreted as Markov model wherein, its proper states are not directly observed [22]. The hidden Markov model is defined as follows.

Definition 6 (hidden Markov model [22]). *A Hidden Markov Model λ is a tuple $(\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$, where:*

- *$\mathbf{\Phi}$ is a finite set of hidden states;*
- *$\mathbf{O}$ is a finite set of observations;*
- *$\mathbf{A}$: $(\mathbf{\Phi} \times \mathbf{\Phi}) \rightarrow [0, 1]$ is a hidden state transition matrix, such that $\forall_{s_1 \in \mathbf{\Phi}} \sum_{s_2 \in \mathbf{\Phi}} \mathbf{A}(s_1, s_2) = 1$;*
- *$\mathbf{B}$: $(\mathbf{\Phi} \times \mathbf{O}) \rightarrow [0, 1]$ are the observation probabilities, such that $\forall_{s \in \mathbf{\Phi}} \sum_{o \in \mathbf{O}} \mathbf{B}(s, o) = 1$;*
- *π: $\mathbf{\Phi} \rightarrow [0, 1]$ is the initial state distribution, such that $\sum_{s \in \mathbf{\Phi}} \pi(s) = 1$.* □

From the above definition, we know that all of the observation elements could be produced in each of the hidden states. The probability of an observation $o \in \mathbf{O}$ in a hidden state $s \in \mathbf{\Phi}$ is denoted by the observation probability $\mathbf{B}(s, o)$. Note that, for a given event log L with an activity set Σ, we want to use a hidden Markov model to represent process mainstream behaviors observed in the event log L, and then link the observations that can be produced by the hidden Markov model to the activities in the event log by using the same identifier, i.e., $\mathbf{O} = \Sigma$.

Given a hidden Markov model $\lambda = (\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$ and an observation sequence $\Theta = o_1, o_2, \ldots, o_n$, the probability of the occurrence of Θ, i.e., $P(\Theta|\lambda)$, can be calculated by using the *Forward-Backward* [21], [23] algorithm. We apply this fundamental problem for Hidden Markov Models to identify whether an event trace is outlier in event logs.

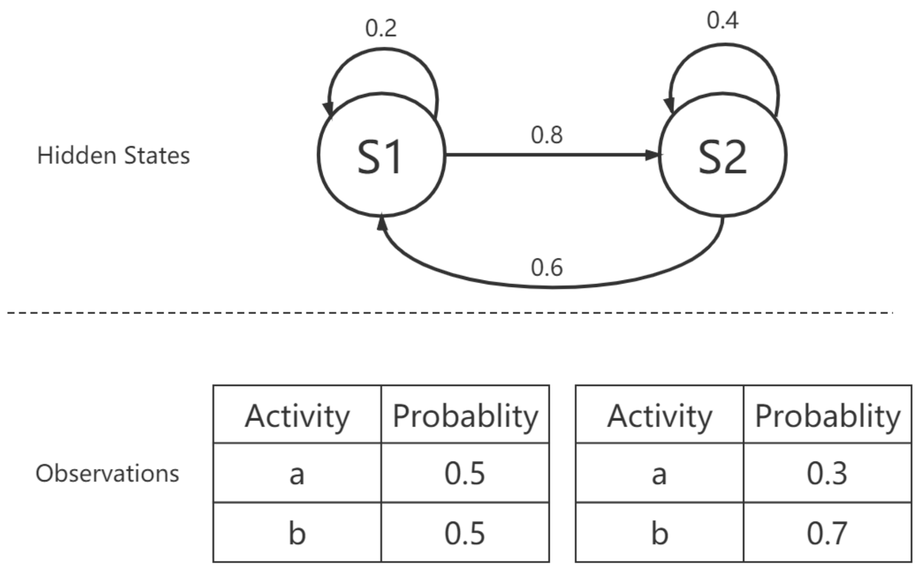
For an illustration purpose, we consider a simplified hidden Markov model $\lambda = (\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$ shown in Fig. 2. The depicted hidden Markov model has two hidden states $\mathbf{\Phi} = \{S_1, S_2\}$, two observations $\mathbf{O} = \{a, b\}$, and the initial state distribution is $\pi = [1.0, 0.0]$. The labeled arrows in this model correspond to the $\mathbf{A}$ matrix, i.e., $\mathbf{A} = \begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$. In this example, the probability of a transition from hidden state S_1 to state S_2 is 0.8. The $\mathbf{B}$ matrix shown underneath the hidden state, i.e., $\mathbf{B} = \begin{bmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{bmatrix}$, gives the probability of producing each activity in that state. The probability of producing the activity a in hidden state S_1 is 0.5. Given an observations sequence $\Theta = aba$, the probability of occurrence of Θ corresponding to the λ is 0.1682.

III. Identify Mainstream Behaviors in Event Logs Based on Frequency-based Approaches

In this section, we present a frequency-based approach to obtain the mainstream process model and the mainstream sublog for representing the primary process behaviors across the entire event log. Given an event log, the process mainstream behaviors imply either a set of event traces that occur frequently or a set of event traces that cover all of the frequently occurring activities. Our approach consists of three steps: (1) selecting a set of event traces based on frequency-based approaches; (2) applying an existing process mining algorithm on the set of event traces selected from (1) to obtain the mainstream process model; (3) obtaining the mainstream sublog by replaying the original event log on the mainstream process model.

To obtain the mainstream process model and the mainstream sublog that represents the mainstream process behaviors across a given event log, we convert the event log to the *grouped event log*. Before we give the *grouped event log* formal definition, we first introduce the *distinct activity sequence*.

Definition 7 (Distinct Activity Sequence). *Given an event log L with an activity set Σ, the distinct activity sequence I is a finite sequence of activities $[\alpha_1, \ldots, \alpha_n]$ satisfying $\exists \sigma \in L, |\sigma| = |I| \wedge (\forall 0 < i < |\sigma|, act(\sigma_i) = I(i))$* □



Activity	Probablity
a	0.5
b	0.5

Activity	Probablity
a	0.3
b	0.7

Fig. 2. A hidden Markov model Example with $\mathbf{\Phi} = \{S_1, S_2\}, \mathbf{O} = \{A, B\}$, and $\pi = [1.0, 0.0]$

For instance, given an event log as shown in Table I, the trace σ_1 is $[e_1, e_2, e_3, e_4]$ where $e_1 = (A, \text{"}08:15\text{"})$, $e_2 = (B, \text{"}10:24\text{"})$, $e_3 = (C, \text{"}10:25\text{"})$, $e_4 = (D, \text{"}13:19\text{"})$. The corresponding distinct activity sequence I_1 is $[A, B, C, D]$

Definition 8 (Grouped Event Log). *Given an event log L, the grouped event log G corresponding to L is a set of distinct activity sequences $\{I_i\}$ satisfying $\exists I_1, I_2 \in G, (|I_1| = |I_2| \wedge \forall 0 < n < |I_1|, I_1(n) = I_2(n)) \rightarrow I_1 = I_2$. The superscript n of a distinct activity sequence indicates the number of corresponding event traces in the L. Labeling function $supp : I \mapsto \mathbb{N}$ is used to get the superscript of the distinct activity sequences.* □

Consider the event log L in Table I, for trace 1, 3, and 6. The corresponding distinct activity sequence for these traces is $[A, B, C, D]$. For trace 2 and trace 4, the corresponding distinct activity sequence is $[A, C, B, D]$. For the distinct activity sequence $[A, E, D]$, only one correspond event trace exists. Hence, the grouped event log G is $\{[A, B, C, D]^3, [A, C, B, D]^2, [A, E, D]\}$. The number of corresponding event traces has a $supp([A, B, C, D]) = 3$. Note that, if the superscript n equals 1, we omit it for simplicity. Using this representation of an event log, we illustrate the steps to obtain the mainstream process model and the mainstream sublog for representing the process mainstream behaviors in the L.

In the first step, we use three different strategies to select the event traces to represent the following mainstream behaviors: 1.) frequent event traces, 2.) frequent activities, or 3.) a combination of both from the event log. For the frequent event traces, the event traces are selected from an event log based on the top xth frequency of the distinct activity sequences in the grouped event log, where x is user-defined. For instance, assuming that x equals two, the distinct activity sequences are the two most frequently occurring traces in G: $[A, B, C, D], [A, C, B, D]$ and the corresponding set of event traces is $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_6\}$. For the frequent activities, we first calculate the frequency of each activity in the grouped event log. Then we obtain a set of frequent activities whose frequency is larger than a user-defined threshold. The event traces that are selected from the event log are based on the top frequencies of the distinct activity sequences that cover the frequent activities set. For instance, the frequencies of activities A, B, C, and D in the G are 3/11, 2/11, 2/11, 3/11, and 1/11 respectively. Assuming the threshold is 2, the set of frequent activities is $\{A, D\}$. The corresponding frequent distinct activity sequences, which cover the frequent activities set, are $[A, B, C, D]$ and the corresponding event traces in the L are $\{\sigma_1, \sigma_3, \sigma_6\}$. For the combination of frequent event traces and activities, we first pick up the event traces based on frequent activities. Then, from the remaining event traces, we select event traces based on the frequent event traces.

For the second step, we apply an existing process discovery algorithm to the event traces that are derived from the first step to obtain the mainstream process model. Finally, for event traces in the original event log, we identify whether an event trace is replayable defined below on the mainstream process model . We refer the set of replayable event traces as the mainstream sublog.

Definition 9 (Replayable). *Given a workflow net $N = (P, T, F)$ and an event trace $\sigma = [e_1, \ldots, e_n]$, the σ is replayable on the N that satisfies $\forall j, 1 \leqslant j \leqslant n, (N, m^{start}) \xrightarrow{act(e_1)} \ldots \xrightarrow{act(e_j)} \ldots \xrightarrow{act(e_n)} (N, m^{end})$, where marking $m^{start} = (i, 1)$, and marking $m^{end} = (o, 1)$.* □

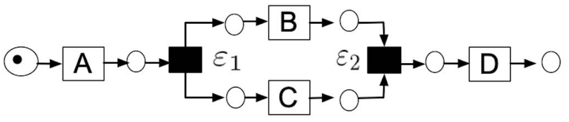


Fig. 3. The Mainstream Process Model Corresponding to Top Two Frequency Event Traces in the Event Log

We use the following example to show how the mainstream sublog is obtained. Consider the event log L, and a mainstream process model $N = (P, T, F)$ shown in Fig. 3 which is derived by applying an inductive miner algorithm [24] on the top two most frequent event traces. From the process model that is constructed by the inductive miner algorithm, there are activities (which are represented as rectangles) and invisible activities (which are represented as the black rectangles, such as ϵ_1 and ϵ_2) in Fig. 3. The invisible activities and circles in the process model are only for routing purposes. These components are produced by process mining algorithms, but are not recorded in event logs.

For the event trace σ_1 shown in Table I, the activity sequence is $[A, B, C, D]$. The initial marking of the mainstream process model $N = (P, T, F)$ is $\{(i, 1)\}$. According to formula 1, there is a firing sequence of transition $[A, B, C, D]$ in the process model, i.e., $(N, \{(i, 1)\}) \xrightarrow{A} (N, \{(P_1, 1)\}) \xrightarrow{\epsilon_1} (N, \{(P_2, 1), (P_3, 1)\}) \xrightarrow{B} (N, \{(P_3, 1), (P_4, 1)\}) \xrightarrow{C} (N, \{(P_4, 1), (P_5, 1)\}) \xrightarrow{\epsilon_2} (N, \{(P_6, 1)\}) \xrightarrow{D} (N, \{(o, 1)\})$. Based on definition 9, the event trace σ_1 is replayable on the N. Similarly, $\sigma_2, \sigma_3, \sigma_4$, and σ_6 are replayable on the mainstream process model. However, σ_5 is not replayable, since there is no firing sequence of transitions in the process model N. Hence, the mainstream sublog is $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_6\}$.

IV. Applying Hidden Markov Models to Filter out Outliers in Event Logs

In this section, we present an approach to apply the hidden Markov model derived from the mainstream sublog and the mainstream process model to filter out outliers within the event logs. The first step of our approach is to construct a hidden Markov model based on the mainstream process model and the mainstream sublog. The second step is to apply the obtained hidden Markov model to identify the outliers and remove them from the event log. Our hypothesis states that this removal should improve the process discovery results. We will validate this hypothesis in Section V.

While a hidden Markov model is an inherently stochastic model, a workflow net is an analytical representation and does not directly support a probabilistic description. Because of this, we need to infer the probabilistic parameters of a hidden Markov model from the mainstream sublog and the mainstream process model.

Given a grouped event log G that corresponds to the mainstream sublog and a mainstream process model $N = (P, T, F)$, we use following rules to construct the hidden Markov model $\lambda = (\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$.

- Hidden states ($\mathbf{\Phi}$): each place in the mainstream process model is represented by exactly one state in the hidden Markov model λ, i.e. $|\mathbf{\Phi}| = |P|$;
- Observations ($\mathbf{O}$): the observations are the grouped event logs G that correspond to the mainstream sublog
- Hidden state transition matrix ($\mathbf{A}$): given two hidden state $s_1, s_2 \in \mathbf{\Phi}$, and the corresponding two places $p_1, p_2 \in P$, the element $\mathbf{A}(s_1, s_2)$ of $\mathbf{A}$ is

$$\frac{\sum_{i \in [0,|G|]} sup(I_i)}{\sum_{i \in [0,|G|]} sup(I_i) \quad + \quad \sum_{j \in [0,|G|]} sup(I_j)}$$

  satisfying $\exists t_1 \in \bullet p_1, t_2 \in (p_1 \bullet \bigcap \bullet p_2) : [t_1, t_2] \in I_i$ and $\exists t_1 \in \bullet p_1, t_2 \in (p_1 \bullet \bigcap \bullet p_x) \wedge p_x \in (P - \{p_2\}) : [t_1, t_2] \in I_j$;
- Observation probabilities matrix ($\mathbf{B}$): given a hidden state $s \in \mathbf{\Phi}$, the corresponding place $p \in P$, and a transition $t \in T$, the element $\mathbf{B}(s, t)$ of $\mathbf{B}$ is assign as follows:
  1) When p is not in the output place, $\mathbf{B}(s, t) = \frac{\sum_{i \in [0,|G|]} sup(I_i)}{\sum_{j \in [0,|G|]} sup(I_j)}$ satisfying $\exists t_1 \in \bullet p : [t_1, t] \in I_i$ and $\exists t_1 \in \bullet p, t_2 \in p\bullet : [t_1, t_2] \in I_j$
  2) When p is the output place, the corresponding hidden state s is a final state, which does not produce any observable activity from the event log. Hence, the element $\mathbf{B}(s, t) = 0$. Instead, it is associated to a dummy end element ϵ, where the element $\mathbf{B}(s, \epsilon) = 1$.
- Initial state distribution (π): the probability of the hidden state corresponding to the input place in N equals one, and probabilities of all other hidden states equal zero.

In the following figure, we will use an example to illustrate the steps required to construct a hidden Markov model $\lambda = (\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$ that is based on a mainstream sublog and a mainstream process model $N = (P, T, F)$.

Example 1. *Consider the new grouped event log G shown in Fig. 4, we select the event traces corresponding to the top three frequency of the distinct activity sequences from the event log, i.e., $[ABD], [ACD]$, and $[ACECD]$. Then, we apply an existing process mining algorithm, for instance, the inductive miner algorithm developed by [24], on the selected event traces. The obtained mainstream process model is represented by a workflow net $N = (P, T, F)$ that is depicted in Fig. 4. Based on the mainstream process model, the event traces corresponding to the distinct activity sequence $[ABEBD]$ are replayable. Hence, the grouped event log G that is corresponding to the mainstream sublog is $\{[ABD]^{10}, [ACD]^{10}, [ACECD]^{10}, [ABEBD]^3\}$.*

According to the rules to construct the hidden Markov model $\lambda = (\mathbf{\Phi}, \mathbf{O}, \mathbf{A}, \mathbf{B}, \pi)$, the hidden states $\mathbf{\Phi}$ contains s_1, s_2, s_3, s_4 that represent the places p_1, p_2, p_3, p_4 respectively. $\mathbf{O}$ is the grouped event log G that corresponds to the mainstream sublog, i.e., $\mathbf{O} = \{[ABD]^{10}, [ACD]^{10}, [ACECD]^{10}, [ABEBD]^3\}$. For initial state distribution π, the input place of N is p_1, which corresponds to the hidden state s_1 and $\pi = [1, 0, 0, 0]$.

We take the hidden states s_3 and s_4 as an example to illustrate the steps for element $\mathbf{A}(s_3, s_4)$ in a hidden state transition matrix. The places corresponding to the hidden states s_3 and s_4 are p_3 and p_4, respectively. As $\bullet p_3 = \{B, C\}$ and $p_3 \bullet \bigcap \bullet p_4 = \{D\}$, we sum up the superscripts of the distinct activity sequences containing the subsequent $[B, D]$ or $[C, D]$, such that the $sup([ABD]) + sup([ACD]) + sup([ACECD]) + sup([ABEBD])$ results in 33 as the numerator of $\mathbf{A}(s_3, s_4)$. Since $p_3 \bullet \bigcap \bullet p_2 = \{E\}$, we sum up the superscripts of the distinct activity sequences containing the subsequent $[C, E]$ or $[D, E]$, i.e., the $sup([ACECD]) + sup([ABEBD])$ results in 13. Hence the $\mathbf{A}(s_3, s_4)$ is $\frac{33}{33+13} \approx 0.717$.

We take the hidden states s_2 and s_4 as examples to illustrate different scenarios of constructing the observation probabilities matrix. The place corresponding to the hidden state s_2 is p_2. Consider the transition C, since $\bullet p_2 = \{A\}$, we sum up the superscripts of distinct activity sequences containing the subsequent $[A, C]$, i.e., $sup([ACD]) + sup([ACECD]) = 20$ as the numerator for $\mathbf{B}(s_2, C)$. Since $p_2 \bullet = \{B, C\}$, we sum up the superscripts of distinct activity sequences containing the subsequent $[A, B]$ or $[A, C]$, i.e., $sup([ACD]) + sup([ABD]) + sup([ABEBD]) + sup([ACECD])$, and the following result is 33. Hence the $\mathbf{B}(s_2, C)$ is $\frac{20}{33} \approx 0.606$. For the hidden state s_4, the place p_4 corresponding to s_4 is output place in the mainstream process model. According to the rule,

the element $\mathbf{B}(s_4, A) = \mathbf{B}(s_4, B) = \mathbf{B}(s_4, C) = \mathbf{B}(s_4, D) = \mathbf{B}(s_4, E) = 0$, and $\mathbf{B}(s_4, \epsilon) = 1$. Similarly, based on the rules, we construct other elements of $\mathbf{A}$ and $\mathbf{B}$, and get the following results:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0.28 & 0 & 0.72 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.39 & 0.61 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.28 & 0.72 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In the second step, we take the event log L and use the hidden Markov model λ produced by the first step as input. Given an event

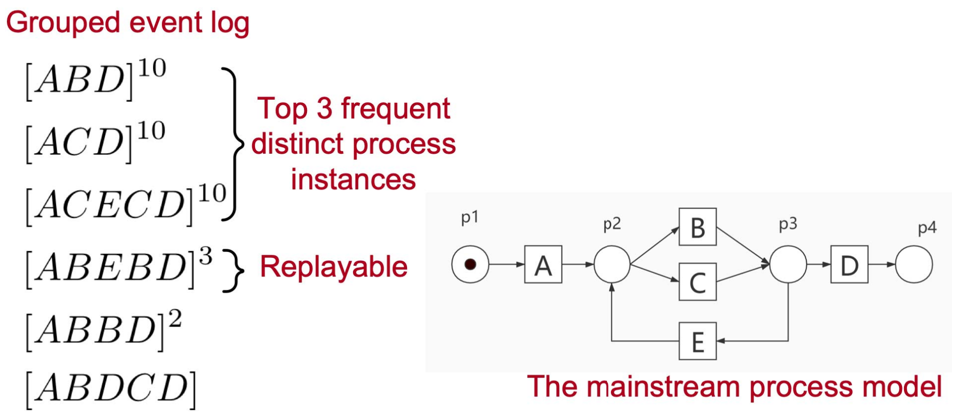


Fig. 4. A mainstream process model corresponding the mainstream sublog

trace $\sigma = [e_1, \ldots, e_n]$, we compute the probability of σ, given the model λ, i.e., $Pr(\sigma|\lambda)$, using the *Forward-Backward Procedure* [23], [25]. Then, we compare the resulted probability with the user-defined threshold κ. We identify each event trace in the L with the probability conditioned on λ being higher than the threshold, i.e., $Pr(\sigma|\lambda) > \kappa$ as an outlier. Finally, we remove the outliers from the event log in order to improve process discovery results. For example, consider the event trace corresponding to the distinct activity sequence $[ABDCD]$ in Example 1. We apply the *Forward-Backward Procedure* to the event trace, and the probability is 0.038. Assuming the threshold of κ is 0.01, the event trace corresponding to $[ABDCD]$ is not an outlier. Similarly, we calculate the probability of the event trace corresponding to $[ABBD]$ and find that it is under the λ, so we treat it as an outlier. Finally, the resulted event log contains event traces corresponding to distinct activity sequence $[ABD], [ACD], [ACECD], [ABEBD], [ABDCD]$.

V. Experimental Evaluation

In this section, we evaluate our proposed approach using both artificial event logs and real-life event logs from [26], [27], [28], [29], [30]. The artificial logs are generated from the Processes and Logs Generator (Plg) [31], which is an open-source tool that is used to generate artificial event logs based on the user-defined process model. The characteristics of the real-life event logs we selected are summarized in Table II. The objectives of our evaluations consist of two components. First, we evaluate if filtering out outliers by using the hidden Markov model can improve the quality of the process models that are discovered from event logs. To do so, we choose an existing Inductive mining algorithm [24], and apply it on the same real-life event logs with and without the hidden Markov model approach. Second, we compare the performance of the proposed approach with two commonly used filtering approaches, i.e., the *Matrix Filter* approach [15] and the *Anomaly Free Automation* approach [13], using the artificial and real-life event logs. Before the evaluation, we first define our evaluation criteria.

TABLE II
Characteristics of the Event Logs Used for the Evaluation

DataSet	Number of Traces	Number of Event Entry	Average number of Event Entry
BPIC2013 [26]	819	2351	2.871
BPIC2020 [27]	10500	56,437	5.374
Hospital Billing [28]	100000	451359	4.514
Road Traffic [30]	150370	561470	3.734

A. Performance Metrics

The quality of a process model is evaluated by two criteria: (1) fitness f, and (2) precision p. The *fitness* measures how well a model can reproduce the process behavior that is contained within a log, and the *precision* measures the degree to which the behavior that is made possible by a model is found within a log [13].The higher the fitness and precision values are, the better quality of the process model. We use the pm4py [32] library to calculate these performance metrics value. However, the fitness and precision are two aspects of a process model which may not always be consistent. The $F1$ score [6] is defined as the harmonic mean of the weighted average fitness f^W and precision p^W, i.e. $F1 = \frac{2 \times f^W \times p^W}{f^W + p^W}$. We will also use the $F1$ score as an evaluation criteria.

For any given artificial event log, we will have two process models that represent the reference model and the mined model. As such, we need to determine the behavioral and structural similarity between these two models [33]. The behavioral similarity metrics analyze the event log to quantify how similar the behavior of the mined model is to that of its reference model in terms of precision and recall [34]. The higher that the behavioral precision and recall is, the greater the similarity is between the referenced model and the mined model. The structural similarity metrics reflects the degree of correct causality relations that exist in the referenced model or the mined model. This is typically measured in terms of precision and recall. A value of both structural precision and recall close to 1 indicates two process models are structurally, very similar.

B. Process Model Discovery Performance Improvement Using the Hidden Markov Model

This set of experiments is to evaluate whether filtering out outliers by using the hidden Markov model can improve the quality of process models discovered from event logs. More specifically, given an event log, we apply the three different approaches presented in Section III to obtain the mainstream sublog and mainstream process model. For each mainstream sublog and mainstream process model , we apply the proposed approach in Section IV to construct the hidden Markov model and filter out outliers from the original event log. Then, we discover a process model from event logs using the inductive mining algorithm presented in [24]. The *fitness*, *precision*, and *F1-score* of each event log and their respective process models are depicted in Table III.

From the experiment results, it is clear that when we use our proposed approach to filter out the outliers from an event log, prior to applying process discovery algorithms, we are able to achieve higher fitness, precision, and F1 score with different real-life event

TABLE III
FITNESS, PRECISION, AND F1 SCORE

	BPIC 2013			BPIC 2020			Road Traffic			Hospital Billing		
	Fitness	Precision	F1	Fitness	Precision	F1	Fitness	Precision	F1	Fitness	Precision	F1
No Filtering Approach	0.922	0.478	0.629	0.913	0.679	0.778	0.768	0.561	0.648	0.785	0.539	0.639
Activity Approach and HMMs	0.935	0.595	0.798	0.945	0.785	0.858	0.791	0.604	0.684	0.858	0.543	0.665
Trace Approach and HMMs	0.937	0.577	0.786	0.947	0.787	0.860	0.812	0.667	0.732	0.826	0.597	0.693
Combination and HMMs	0.947	0.641	0.832	0.958	0.842	0.896	0.829	0.709	0.764	0.889	0.61	0.723

logs. For the real-life event logs, the average improvement percentage of fitness, precision, and F1 score are 7.23%, 24.44%, and 19.57%, respectively.

C. Hidden Markov Model Approach Vs Two Commonly Used Filtering Approaches

This set of experiments is to compare our proposed approach with the two commonly used filtering approaches such as the *Matrix Filter* approach (**MF**) [15] and the *Anomaly Free Automation* approach (**AFA**) [13] using the aforementioned artificial and real-life event logs.

Artificial event logs: For benchmarking the different filtering approaches, we consider two different complexity levels for the reference models by using four basic structures found within a process. The first level of the reference model contains a parallel structure, an exclusive-choice structure, a loop structure, and a sequence structure. The second level of the reference model contains the interactions between these basic structures by combining them together. For each level of the reference model, we create ten different reference models that contain an increasing activity number. Then, for each referenced model, we use the *Plg* tool to generate ten event logs by injecting an incremental amount of noise ranging from 1% to 30% to produce an event log that contains one thousand event traces.

For each artificial event log, we first use the three different approaches presented in Section III to obtain the mainstream sublog and mainstream process model , and construct the hidden Markov models. Then, we filter out the outliers from the given event log based on the hidden Markov model. We also apply the two commonly used approaches, i.e., **MF** and **AFA** on the given artificial event log to filter out the outliers. For each clean event log derived from five different filtering approaches , we apply the inductive mining algorithm to discover a process model, and calculate the behavioral recall, behavioral precision, structural recall, and structural precision. The results are depicted in Fig 5, Fig 6.

As shown in Fig 5, Fig 6, the proposed approach outperforms the two commonly used approaches on average cases. More specifically, the performance of our proposed approach (Combination + HMMs) is 10.43%, 11.92%, 9.61%, 9.20% higher than the performance of the **MF** approach with respect to behavioral recall, behavioral precision, structural recall ,and structural precision, respectively. The performance of our proposed approach (Combination + HMMs) is 8.89%, 9.57%, 7.47%, 8.85% higher than the performance of **ANA** approach with respect to behavior recall, behavior precision, structural recall ,and structural precision, respectively. We also observe that the approach using the combination strategy to construct the hidden Markov models outperforms the approaches using frequent activity or event traces to construct a hidden Markov models with respect to the four criteria.

Real-life event logs: We use five different approaches, i.e., **MF** approach, **AFA** approach, HMMs derived from frequent activity approach, HMMs derived from frequent event trace approach, and HMMs derived from combination approach on the real-life event logs to filter out outliers. Then, we apply the inductive mining algorithm on clean event logs to discover a process model and calculate fitness, precision, and F1 score. The results are depicted in Fig 7, Fig 8, and Fig 9.

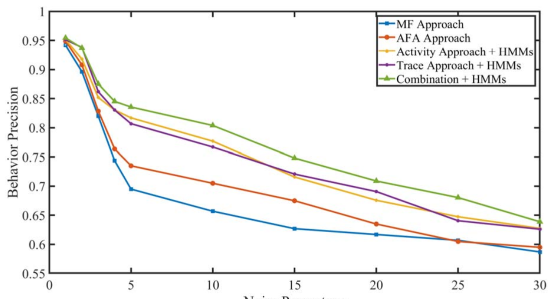
From the experiment results, the performance of the proposed approach (Combination + HMMs) is 0.4%, 11.3%, and 6.7% higher than the **MF** approach with respectively for *fitness*, *precision*, and *F1 score*. The fitness derived from the proposed approach (Combination + HMMs) is 0.54% lower than the **ANA** approach, but the precision is higher. Hence, we are able to achieve 3.9% higher F1 score with different real-life event logs.

VI. RELATED WORK

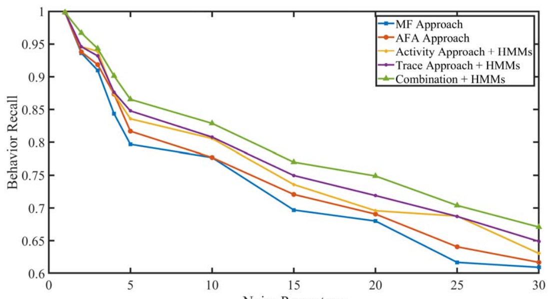
The presence of outlier data in the event logs can adversely affect the quality of discovered process models. The ProM framework offers several plugins for filtering anomalous data. These filtering methods are performed based on activity frequencies/positions, certain attributes associated with events, and prefix-based rules. i.e, whether a trace is a prefix of another trace in the given event log or not, etc. However, these plugins require user input and domain knowledge. As a result, several studies have leveraged various methods to remove the so-called outliers from event logs before implementing process discovery algorithms. Among these works, some require a reference model to replay process instances and filter out outliers. However, these methods are not often applicable due to the unavailability of reference models.

On the other hand, several studies perform filtering based on sequence mining algorithms. Some of these generic methods are based on creating data models to represent normal behaviour. The constructed models are then used for filtering out anomalous traces. An approach of this type is proposed in [13], which creates an abstraction of the observed behaviors based on event transitions using an Anomaly-Free Automaton (AFA). Subsequently, this algorithm removes the infrequent event transitions from the constructed automaton, and considers non-replayable log traces on the so-called automaton as outlier traces. Although showing improvement in performance measures, this method cannot take incomplete traces or the ones with missing events into account. In [15], the authors propose a method for computing the occurrence likelihood of particular activities based on their preceding sequence of activities. In this method, an event with a likelihood lower than a pre-defined threshold is considered an outlier. Consequently, using trace-level filtering, the corresponding trace is opted out. The filtering method proposed in [35] yields an improvement upon the latter method by repairing traces which contain outlier event(s) instead of totally removing them. Outliers in this study are identified based on contexts frequency in traces. A drawback of this method is the addition of unreal activities to the event log thorough repairing process.

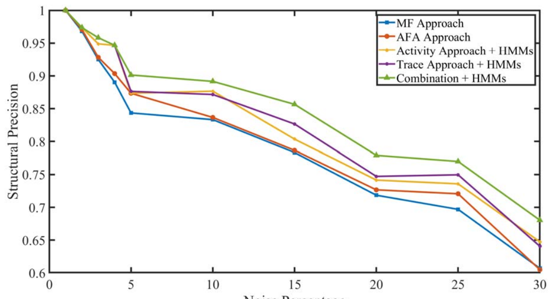
All the aforementioned algorithms which are based on direct event relationships ignore the existence of parallel activities and long term dependencies in the traces. In [15], the length of sequence can be increased but it results in increasing the complexity of this



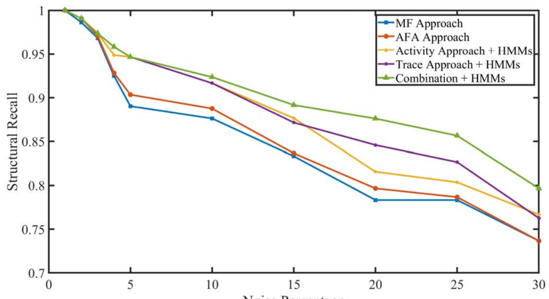
(a) Behavior Precision



(b) Behavior Recall

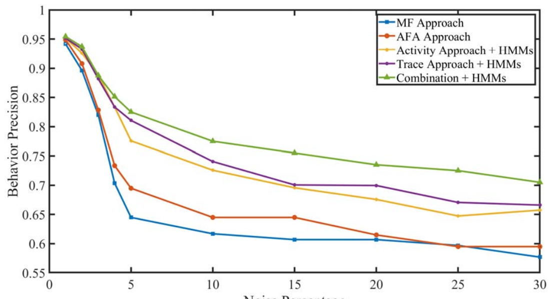


(c) Structural Precision

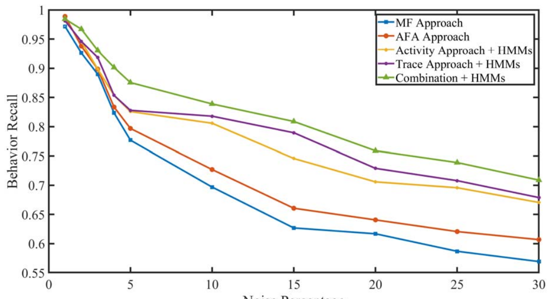


(d) Structural Recall

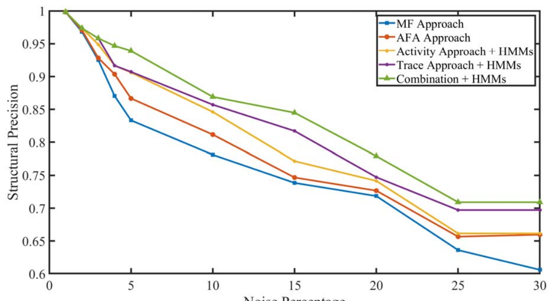
Fig. 5. Results Under First Level Model Complexity



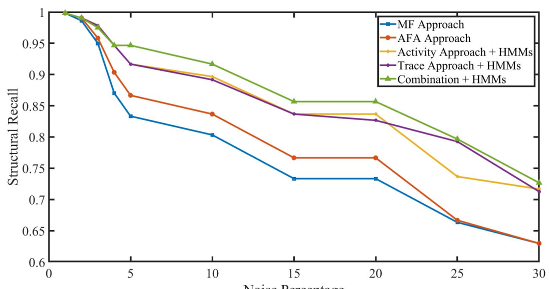
(a) Behavior Precision



(b) Behavior Recall



(c) Structural Precision



(d) Structural Recall

Fig. 6. Results Under Second Level Model Complexity

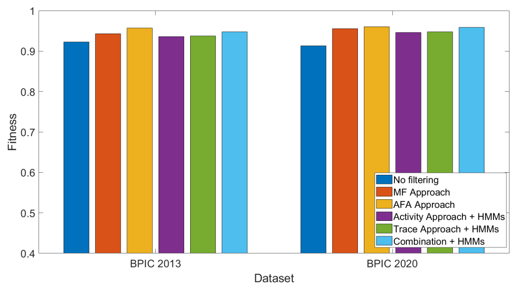


Fig. 7. Fitness Values in Real Life Event logs

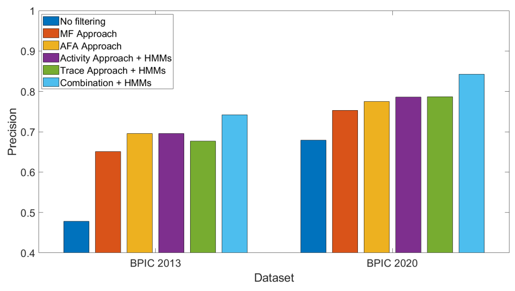


Fig. 8. Precision Values in Real Life Event logs

method. Also, long term relations cannot be identified in the proposed AFA algorithm. To address these issues and capture long term flow relationships between activities, [36] proposes a filtering technique based on sequential patterns and rules. e.g, whether a particular event is followed by another one somewhere in the trace or not. A limitation of this method is that it disregards the directly follow relations and merely considers indirectly follow relations. Therefore, this algorithm might fail to capture some outlier behaviours.

In this study, we use a statistical model of sequential data, the hidden Markov model, for filtering outlier traces from event logs. Hidden Markov models (HMM) are applied for anomaly detection in sequenced data as they are believed to capture the sequences in their hidden space built from input sequences. HMM was initially used for anomaly detection in [37] to learn the patterns in processes and classify them as normal and abnormal. In this study, we use HMM to detect outliers in event log and compare our performance results with other state-of-the-art algorithms. Using HMM's ability to preserve the nature of sequences [38], we outperform merely preserving indirect event relations or direct follows respectively proposed in AFA and MF algorithms in [13] and [15].

VII. Conclusion

In this paper, we present an approach that uses hidden Markov models to filter out outliers from event logs prior to applying process discovery algorithms to improve process discovery results. Our experiments on artificial event logs and real-life event logs show that: (1) process models obtained by filtering out outliers with hidden Markov models have higher fitness, precision, and F1 score than process models obtained by directly applying discovery algorithms on the event logs; (2) the proposed filtering method outperforms two commonly used filtering approaches, namely the *Matrix Filter* approach and the *Anomaly Free Automation* approach for both artificial event logs and real-life event logs. As future work, we want to distinguish between noise and infrequent behaviour based on features of the given event log. Our goal is to better identify

score.png

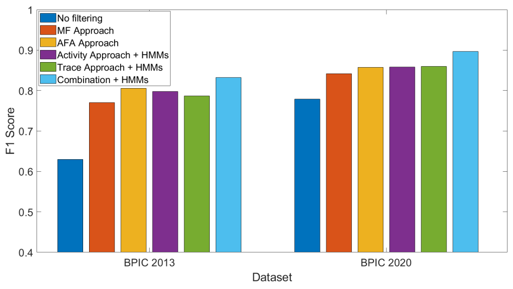


Fig. 9. F1 Scores in Real Life Event logs

the infrequent behaviour information, which is a behaviour that only occurs in an exceptional case within a process, and improve the accuracy for process representation.

Acknowledgement

The work is supported in part by NSF 1952247, NSF 1952225, and NSF 1929469.

References

[1] Ronny S Mans, Wil MP van der Aalst, and Rob JB Vanwersch. Healthcare processes. In *Process Mining in Healthcare*, pages 11–15. Springer, 2015.

[2] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[3] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.

[4] Marc Solé and Josep Carmona. Process mining from a basis of state regions. In *International Conference on Applications and Theory of Petri Nets*, pages 226–245. Springer, 2010.

[5] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer, 2007.

[6] Wil Van Der Aalst. Data science in action. In *Process mining*, pages 3–23. Springer, 2016.

[7] Suriadi Suriadi, Ronny S Mans, Moe T Wynn, Andrew Partington, and Jonathan Karnon. Measuring patient flow variations: A cross-organisational process mining approach. In *Asia-Pacific Conference on Business Process Management*, pages 43–58. Springer, 2014.

[8] Suriadi Suriadi, Moe T Wynn, Chun Ouyang, Arthur HM ter Hofstede, and Nienke J van Dijk. Understanding process behaviours in a large insurance company in australia: A case study. In *International Conference on Advanced Information Systems Engineering*, pages 449–464. Springer, 2013.

[9] Mohammadreza Fani Sani, Sebastiaan J van Zelst, and Wil MP van der Aalst. Applying sequence mining for outlier detection in process mining. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 98–116. Springer, 2018.

[10] Suriadi Suriadi, Robert Andrews, Arthur HM ter Hofstede, and Moe Thandar Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information Systems*, 64:132–150, 2017.

[11] Seppe KLM vanden Broucke and Jochen De Weerdt. Fodina: A robust and flexible heuristic process discovery technique. *decision support systems*, 100:109–118, 2017.

[12] Alejandro Bogarín Vega, Rebeca Cerezo Menéndez, and Cristobal Romero. Discovering learning processes using inductive miner: A case study with learning management systems (lmss). *Psicothema*, 2018.

[13] Raffaele Conforti, Marcello La Rosa, and Arthur HM ter Hofstede. Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):300–314, 2016.

[14] *process-discovery*, 2016. http://pm4py.pads.rwth-aachen.de/documentation/process-discovery/alpha-miner/.

[15] Mohammadreza Fani Sani, Sebastiaan J van Zelst, and Wil MP van der Aalst. Improving process discovery results by filtering outliers using conditional behavioural probabilities. In *International Conference on Business Process Management*, pages 216–229. Springer, 2017.

[16] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011.

[17] AJMM Weijters and Wil MP van der Aalst. Process mining: discovering workflow models from event-based data. In *Belgium-Netherlands Conf. on Artificial Intelligence*. Citeseer, 2001.

[18] Christian W Günther and Wil MP Van Der Aalst. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management*, pages 328–343. Springer, 2007.

[19] Jörg Desel and Wolfgang Reisig. *Place/transition Petri Nets*, pages 122–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[20] Wil MP Van der Aalst, AJMM Weijters, and Laura Maruster. Workflow mining: Which processes can be rediscovered. Technical report, BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven, 2002.

[21] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[22] Anne Rozinat, Manuela Veloso, and Wil MP Van Der Aalst. Using hidden markov models to evaluate the quality of discovered process models. *Extended Version. BPM Center Report BPM-08-10, BPMcenter. org*, 161:178–182, 2008.

[23] Leonard E Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[24] Joachim Herbst and D Karagiannis. An inductive approach to the acquisition and adaptation of workflow models. In *Proceedings of the IJCAI*, volume 99, pages 52–57. Citeseer, 1999.

[25] Leonard E Baum. Growth functions for trasformations on manifolds. *Pac. J. Math.*, 27(2):211–227, 1968.

[26] Bpi challenge 2013. https://data.4tu.nl/repository/uuid:3537c19d-6c64-4b1d-815d-915ab0e479da.

[27] Bpi challenge 2020. https://data.4tu.nl/repository/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5.

[28] Hospital billing - event log. https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741.

[29] Synthetic event logs - review example. https://data.4tu.nl/repository/uuid:da6aafef-5a86-4769-acf3-04e8ae5ab4fe.

[30] Road traffic fine management process. https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[31] Andrea Burattin. Plg2: Multiperspective process randomization with online and offline simulations. In *BPM (Demos)*, pages 1–6, 2016.

[32] *State-of-the-art-process mining in Python*, 2020. https://pm4py.fit.fraunhofer.de/.

[33] Ana Karla A de Medeiros, Anton JMM Weijters, and Wil MP van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[34] Hsin-Jung Cheng and Akhil Kumar. Process mining on noisy logs—can log sanitization help to improve performance? *Decision Support Systems*, 79:138–149, 2015.

[35] Mohammadreza Fani Sani, Sebastiaan van Zelst, and Wil Aalst. *Repairing Outlier Behaviour in Event Logs*, pages 115–131. 06 2018.

[36] Mohammadreza Fani Sani, Sebastiaan van Zelst, and Wil Aalst. Applying sequence mining for outlier detection in process mining: Confederated international conferences: Coopis, ctc, and odbase 2018, valletta, malta, october 22-26, 2018, proceedings, part ii. pages 98–116, 10 2018.

[37] Bo Gao, Hui-Ye Ma, and Yu-Hang Yang. Hmms (hidden markov models) based on anomaly intrusion detection method. In *Proceedings. International Conference on Machine Learning and Cybernetics*, volume 1, pages 381–385 vol.1, 2002.

[38] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 24:1 – 1, 05 2012.