Toward Efficient Support for Business Process Event Log Sampling

Xuan Su, *Graduate Student Member, IEEE*, Cong Liu, *Member, IEEE*, Shuaipeng Zhang, Qingtian Zeng, *Member, IEEE*, Qi Mo, and Long Cheng, *Senior Member, IEEE*

***Abstract*—Large volumes of event logs have been accumulated by business information systems. Accompanied by that, various process discovery techniques are invented to uncover underlying business processes based on event logs. Event log sampling, recognized as one of the most effective techniques for accelerating discovery efficiency, has gained significant attention in recent days. However, achieving high performance in sampling while maintaining superior sample log quality remains a challenge for current techniques. To tackle the problem, a novel event log sampling technique, denoted as *sigRank*, is introduced to improve both the sampling efficiency and the quality of the sample log by quantifying the significance of each trace. The proposed sampling technique has been implemented as a publicly available tool in the open-source process mining platform ProM. Compared with state-of-the-art techniques using 12 public event logs, we experimentally illustrate that the proposed approach can significantly accelerate sampling efficiency while guaranteeing superior sample log quality for process discovery.**

***Index Terms*—Petri nets, process discovery, event log sampling, efficiency, quality evaluation.**

I. Introduction

PROCESS mining is recognized as a pivotal technique for enabling business process automation in the era of digital transformation. Specifically, it provides a set of techniques to extract process insights from event logs to monitor, analyze and control the execution of processes [1]. As one of the most fundamental process mining techniques, process discovery has received much attention recently. Dozens of discovery approaches, e.g., *Alpha Miner* [2], *Heuristic Miner* [3], *Inductive Miner* [4], *Hierarchical Miner* [5], *Split Miner* [6] and *CrossOrg Miner* [7], have been proposed to construct process models from event logs. However, existing process discovery techniques still meet severe performance problem when processing large-scale event logs [8].

To tackle the limitation, process discovery techniques over distributed platforms such as the ones based on MapReduce and Spark frameworks become desirable [9]. These platforms scale computing resources effectively and provide straightforward scale-out for event log processing [10]. For example, Evermann introduced the MapReduce-based distributed *Alpha Miner* and *Heuristic Miner* via reprogramming existing programs in [11]. Although these approaches have shown remarkable ability to speed up process discovery from large-scale event logs, the re-programming process itself is actually not easy, and in-depth knowledge of the process discovery algorithms as well as the programming syntax is required. Moreover, high-end computer clusters or cloud instances are always required for execution, the analysis cost of which will be high for end users. In contrast, event log sampling techniques are proved to be one of the most promising techniques to improve process discovery efficiency [12], [13], [14].

Although event log sampling improves process discovery efficiency, existing techniques may still struggle on large and complex event logs, e.g., those with thousands of variants. To tackle this problem, a novel event log sampling technique, named *sigRank*, is introduced to improve both sampling efficiency and sample log quality. More specifically, we give the general design of the proposed technique. In addition, the proposed technique is integrated into the open-source process mining platform ProM, and experimental evaluation using 12 public event logs illustrated that the proposed *sigRank* can significantly accelerate the sampling efficiency while guaranteeing superior sample log quality for process discovery, compared to existing event log sampling techniques.

The remainder of this paper is organized below. Section II introduces the related work. Section III presents the background knowledge. Section IV details the *sigRank* sampling technique. Section V details the tool implementation. Section VI presents an experimental evaluation using public event logs. Finally, Section VII concludes the whole paper.

Received 26 May 2025; revised 1 February 2026; accepted 9 February 2026. Date of publication 16 February 2026; date of current version 10 April 2026. This work was supported in part by the National Key R&D Program of China under Grant 2022ZD0119501, in part by the National Natural Science Foundation of China under Grant 62472264, Grant 52374221, and Grant 62562063, in part by the Natural Science Distinguished Youth Foundation of Shandong Province under Grant ZR2025QA13, and in part by the national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UID/04152/2025 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS and UID/PRR/04152/2025. *(Corresponding authors: Cong Liu; Qi Mo.)*

Xuan Su and Qingtian Zeng are with the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China (e-mail: su_xuan2021@163.com; qtzeng@163.com).

Cong Liu is with the NOVA Information Management School, Nova University of Lisbon, 1070-312 Lisbon, Portugal, and also with the School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China (e-mail: cliu@novaims.unl.pt).

Shuaipeng Zhang is with the School of Software, Shandong University, Jinan 250000, China (e-mail: 15994069715@163.com).

Qi Mo is with the School of Software, Yunnan University, Kunming 650091, China (e-mail: moqi@ynu.edu.cn).

Long Cheng is with the School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China (e-mail: lcheng@ncepu.edu.cn).

Digital Object Identifier 10.1109/TSC.2026.3665370

II. Related Work

In this section, we briefly introduce existing work related to process discovery, distributed process discovery, and event log sampling.

A. Process Discovery

Since 2000 s, dozens of process discovery algorithms are invented to uncover process models from event logs. As one of the most well-known process discovery techniques, *Alpha Miner* constructs a Petri net-based process model by analyzing directly-follow relation, concurrency relation, and causality relations among activities from event logs [2]. To cope with limitations of *Alpha Miner*, some extensions have been made, such as non-free-choice constructs [15], duplicated activities [16], short loops [17], and invisible activities [18]. In addition, *Heuristic Miner* [19] is introduced to handle infrequent behavior by considering the dependency frequency. As the state-of-the-art process discovery technique, *Inductive Miner* [4], has been proposed to deal with noisy logs [20] and incomplete logs [21] to support high-quality discovery. Furthermore, *Split Miner* [6] is proposed to ensure the generation of deadlock-free process models. More recently, *Hierarchical Miner* is proposed to support multi-layered sub-process discovery in [22] and *CrossOrg Miner* is proposed to support cross-organization process model discovery in [7].

B. Distributed Process Discovery

With the continually increasing volume of accumulated event logs in the Big Data era, existing process discovery techniques are bothered by performance challenges. This may attributes to the fact that existing process discovery techniques typically running on a single machine, i.e., CPU, GPU, memory, and other hardware resources are limited. In this setting, by leveraging a large amount of computing resources, performing process discovery over distributed platforms becomes desirable. For example, the well-known MapReduce framework [23] has been adopted and reprogrammed to support process discovery in [11]. Generally speaking, the distributed implementations mainly rely on constructing log abstractions for a specific technique, and then, the whole computation tasks are decomposed into a set of MapReduce jobs and executed by multiple machines in parallel. Although such approaches have been proved to achieve speedups over traditional single-machine discovery architecture, their computation paradigm typically follows the traditional manner, i.e., re-programming existing implementations and performing computations to all traces in an event log.

C. Event Log Sampling

Different from distributed process discovery techniques that reprogram existing process discovery techniques, event log sampling techniques offer a novel way to speed up process discovery efficiency. Generally, it enables fast process discovery over event logs via reusing existing algorithms (tools or codes) directly. For instance, the well-known *LogRank* sampling technique presented in [8] and [24] first obtains a small representative sample log from a large-scale one, then the retrieved sample log is used instead of the original one by process discovery techniques. In [25], a series of log sampling techniques based on biased trace selection strategies are introduced. Among these, frequency-based and similarity-based sampling techniques are shown to be particularly effective in preserving representative behavior. More specially, the frequency-based sampling technique assigns high priority to traces that have higher occurrence frequency. Differently, the similarity-based sampling technique first computes a similarity for each trace, and the priority of each trace is assigned based on its similarity value. Note that both the similarity-based and the *LogRank* sampling techniques aim to keep the main-stream behavior in the retrieved sample event log. Additionally, Pasquadibisceglie et al. proposes *PROMISE+*, an abstraction-based event log summarization technique that leverages predictive process mining using LSTM to generate representative traces for process discovery in [26]. In [27], an incremental prototype selection algorithm is presented, which applies K-medoids clustering based on edit distance to group similar process instances and selects the central trace of each cluster as a representative.

However, these event log sampling techniques may be very slow if the original event log is large and complex. For example, the sampling time required by the *LogRank* may be much longer than the discovery time required from the original log for some special cases. To improve sampling efficiency, a sampling technique $LogRank^{+}$ has been proposed in [14]. In addition, several machine learning- and clustering-based summarization techniques, such as *PROMISE+* and K-medoids-based selection, achieve high representativeness but typically require model training or iterative optimization, which increases computational cost. These limitations motivate lightweight and training-free heuristic techniques that can be applied directly to event logs. As shown in this work, such designs improve the sampling efficiency while preserving representativeness.

III. Background Knowledge

This section introduces background knowledge on event log, process discovery, Petri nets, and event log sampling.

A. Event Logs

An *event log* refers to a finite set of events with multiple attributes.

Definition 1 (Event Log): Let U_A be the universe of activities. A trace $\sigma \in U_A^*$ is defined as a sequence of activities/events. $L \in \mathbf{B}(U_A^*)$ is an event log such that $\mathbf{B}(U_A^*)$ is the set of all multisets of sequences on U_A.

Besides activity, events contain additional attributes, i.e., the *resource* (e.g., person or equipment) to execute or initiate an activity, or the *timestamp* of an event to conduct performance evaluation. Considering, for example, some process discovery techniques depend on extra information like *resource* and *cost*. Note that this paper is limited to event logs of activity sequences and other attributes are not considered.

Considering, for example, an event log L_C with 75 traces (23 distinct traces, also known as variants) and 8 activities.

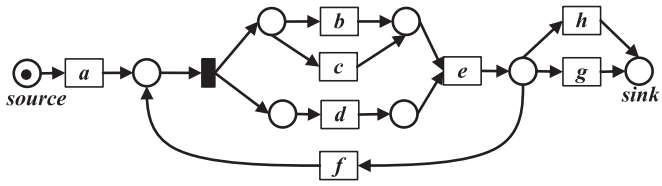


Fig. 1. Process model discovered from L_C using IM.

$$
\begin{aligned}
L_C = [&\langle a,c,d,e,h\rangle^{16}, \langle a,b,d,e,g\rangle^{9}, \langle a,d,c,e,h\rangle^{8},\\
&\langle a,b,d,e,h\rangle^{8}, \langle a,c,d,e,g\rangle^{4}, \langle a,d,c,e,g\rangle^{4},\\
&\langle a,d,b,e,h\rangle^{4}, \langle a,d,b,e,g\rangle^{3}, \langle a,c,d,e,f,d,b,e,h\rangle^{4},\\
&\langle a,c,d,e,f,b,d,e,h\rangle^{2}, \langle a,c,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,c,d,e,h\rangle^{1}, \langle a,d,c,e,f,d,b,e,h\rangle^{1},\\
&\langle a,c,d,e,f,d,b,e,g\rangle^{1}, \langle a,d,c,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,b,d,e,f,c,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,d,b,e,g\rangle^{1},\\
&\langle a,d,c,e,f,b,d,e,f,b,d,e,g\rangle^{1}, \langle a,d,c,e,f,d,b,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,b,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,b,d,e,h\rangle^{1},\\
&\langle a,d,b,e,f,b,d,e,f,d,b,e,g\rangle^{1},\\
&\langle a,c,d,e,f,b,d,e,f,d,b,e,g\rangle^{1}].
\end{aligned}
$$

B. Process Discovery and Petri Nets

By taking as input an event log, a process model that encodes the execution behavior of a business process is built by existing process discovery techniques. To describe process behavior, various process models, e.g., Petri nets, Event-driven Process Chains (*EPC*), and Business Process Modelling Notations (*BPMN*s), can be used.

Definition 2 Process Discovery): Let U_M be the universe of process models. A process discovery technique is defined as a function Θ from an event log $L \in \mathbf{B}(U_A^*)$ to a process model $M \in U_M$, i.e., we have $\Theta(L) = M$.

Petri nets are broadly applied to formalize business processes [28], [29], [30], service processes [31], software processes [32],emergency response processes [33], [34], etc. Without loss of generality, *Petri nets* is used to describe discovered process models in the following discussion.

Definition 3 (Petri Nets [35]): $PN = (P, T, F, l)$ is a *Petri net* satisfying: (1) P is a set of places and T is a set of transitions such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$; (2) $F \subseteq (P \times T) \cup (T \times P)$ is a set of flow relations; and (3) $l : T \to \mathscr{A}$ is defined as a transition labeling function such that $\mathscr{A}$ is a label set and $\tau \in \mathscr{A}$ refers to an invisible label.

For any $x \in P \cup T$, ${}^\bullet x = \{y | (y, x) \in F\}$ is the pre-set of x and $x^\bullet = \{y | (x, y) \in F\}$ is the post-set of x. $m \in \mathbf{B}(P)$ is a marking of a Petri net, which denotes the number of tokens each place contains. A *marked Petri net* is defined as (PN, m_0) with m_0 be its *initial marking*.

The Petri net-based process model discovered from L_C using *Inductive Miner (IM)* [4] should look like the one shown in Fig. 1 where $[source]$ is its initial marking.

C. Event Log Sampling Techniques

In general, event log sampling techniques typically implement an inner trace ranking mechanism to build sample logs by selecting highly-ranked traces from the original log.

Definition 4 (Log Sampling Techniques): Let U_L be the universe of event logs. A log sampling technique is defined as a function Π from an original log $L_0 \in \mathbf{B}(U_A^*)$ to a sample log $L_s \in \mathbf{B}(U_A^*)$, i.e., $\Pi(L_0) = L_s$, where $\forall \sigma \in L_s : \sigma \in L_0$, i.e., L_s is a subset of L_0.

Definition 4 indicates that an event log sampling technique typically takes as input an event log and returns one of its subsets as the sample log based on a certain strategy. Taking the *LogRank* sampling technique in [8] as an example, a sample log of L_C with the sampling ratio 15%, denoted as L_C^s, is obtained and shown below. It is observed that L_C^s contains 12 traces/variants, 152 events, and 8 activities.

$$
\begin{aligned}
L_C^s = [&\langle a,c,d,e,f,d,b,e,h\rangle^{1}, \langle a,c,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,h\rangle^{1}, \langle a,c,d,e,f,d,b,e,g\rangle^{1},\\
&\langle a,d,c,e,f,b,d,e,f,c,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,d,b,e,g\rangle^{1},\\
&\langle a,d,c,e,f,b,d,e,f,b,d,e,g\rangle^{1}, \langle a,d,c,e,f,d,b,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,b,d,e,f,b,d,e,g\rangle^{1},\\
&\langle a,d,c,e,f,d,b,e,f,b,d,e,h\rangle^{1},\\
&\langle a,d,b,e,f,b,d,e,f,d,b,e,g\rangle^{1},\\
&\langle a,c,d,e,f,b,d,e,f,d,b,e,g\rangle^{1}].
\end{aligned}
$$

IV. SIGRANK SAMPLING TECHNIQUE

A novel and efficient event log sampling technique is first detailed, and then introduce how to quantitatively compare different sampling techniques.

A. An Approach Overview

Fig. 2 shows an approach overview of the novel sampling and evaluation techniques with the following two phases:

- *Phase 1: sigRank Sampling Technique:* Given a large-scale event log and a user-defined ratio, *sigRank* applies an inner ranking of traces and selects the most important ones to construct the sample log.
- *Phase 2: Effectiveness Evaluation:* The effectiveness of an arbitrary sampling technique is evaluated from both the efficiency and quality aspects.
  - *Efficiency Evaluation:* The sum of sampling time and model discovery time using the sample log is computed and compared to the model discovery time using the original log. If the former is smaller than the latter, it is concluded that the sampling is effective.
  - *Quality Evaluation:* Assume L_0 is an arbitrary log and L_S is its sample log based on a sampling technique. We take these logs as input and perform process discovery. The discovered models are denoted as M_0 and M_S, respectively. Then, the sample log quality can be evaluated as follows:

$$Qua(L_S, L_0) = \frac{Q(L_0, M_S)}{Q(L_0, M_0)}, \tag{1}$$

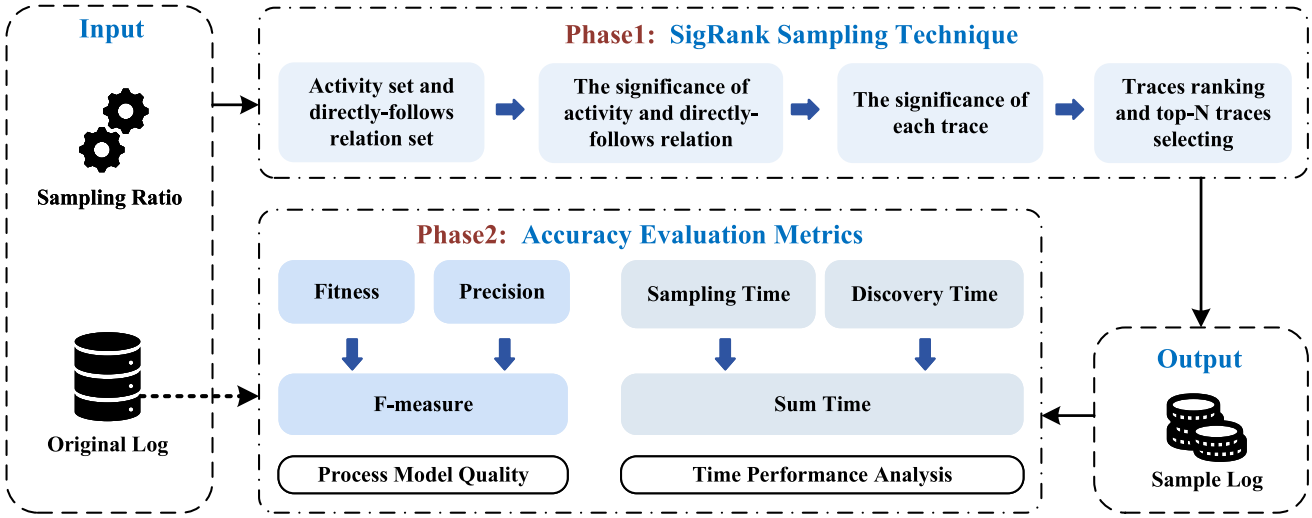


Fig. 2. An approach overview.

where $Q(L_0, M_S)$ represents the quality of the model discovered from the sample log M_S against original log, and $Q(L_0, M_0)$ represents the quality of the model discovered from the original log against original log.

B. Sigrank *Sampling Technique*

Sampling techniques are designed to extract a subset of significant traces from the original log to be analyzed using less time and computing resources. The significance of a trace can be quantified by the behavior it contains. Here, the behavior refers to activities, directly-follows relations, etc, if the sampling targets at improving discovery efficiency. In this way, the significance of a trace can be quantified by the significance of its activities and directly-follows relations. The proposed *sigRank* technique first computes a significance score for each trace in the original log. It then selects the most significant traces to construct the sample log.

Given an arbitrary, the activity set and the directly-follows relation set are first defined.

Definition 5 (Activity Set): Let $L \in \mathbf{B}(U_A^*)$ be an event log, its activity set is: $actS(L) = \bigcup_{\sigma \in L} \{\sigma(i) | 1 \leq i \leq |\sigma|\}$ where $\sigma(i)$ represents the ith element of σ.

Definition 6 (Directly-follows Relation Set): Let $L \in \mathbf{B}(U_A^*)$ be an event log, its directly-follows relation set is: $dfgS(L) = \bigcup_{\sigma \in L} \{(\sigma(i), \sigma(i+1)) | 1 \leq i \leq |\sigma| - 1\}$.

Consider an example log $L_E = [\sigma_1 = \langle a, b, d, e \rangle, \sigma_2 = \langle a, c, e \rangle, \sigma_3 = \langle b, c \rangle, \sigma_4 = \langle b, d \rangle]$, we have $actS(L_C) = \{a, b, c, d, e\}$, and $dfgS(L_C) = \{\langle a, b \rangle, \langle b, d \rangle, \langle d, e \rangle, \langle a, c \rangle, \langle c, e \rangle, \langle b, c \rangle\}$.

The significance of activity a in event log L, denoted as $sig(a, L)$, is calculated below:

$$sig(a, L) = \frac{|\bigcup_{\sigma \in L} \{\sigma | \exists 1 \leq i \leq |\sigma| \wedge \sigma(i) = a\}|}{|L|}, \quad (2)$$

where $|\bigcup_{\sigma \in L} \{\sigma | \exists 1 \leq i \leq |\sigma| \wedge \sigma(i) = a\}|$ denotes the number of traces that contain activity a in L.

The significance of directly-follows relation $\langle a, b \rangle$ in L, denoted as $sig(\langle a, b \rangle, L)$, can be calculated as follows:

$$sig(\langle a, b \rangle, L) = \frac{|\bigcup_{\sigma \in L} \{\sigma | \exists 1 \leq i \leq |\sigma| - 1 \wedge \sigma(i) = a \wedge \sigma(i+1) = b\}|}{|L|}, \quad (3)$$

where $|\bigcup_{\sigma \in L} \{\sigma | \exists 1 \leq i \leq |\sigma| - 1 \wedge \sigma(i) = a \wedge \sigma(i+1) = b\}|$ denotes the number of traces that contain directly-follows relation $\langle a, b \rangle$ in L.

Given a trace $\sigma \in L$, its average activity significance in L, denoted as $sigAvgAct(\sigma, L)$, is calculated as follows:

$$sigAvgAct(\sigma, L) = \frac{\sum_{i=1}^{|\sigma|} sig(\sigma(i), L)}{|\sigma|}, \quad (4)$$

where $\sum_{i=1}^{|\sigma|} sig(\sigma(i), L)$ denotes the sum significance values of all activities in trace σ. Given a trace $\sigma \in L$, its average directly-follows relation significance in L, denoted as $sigAvgDfr(\sigma, L)$, is calculated below:

$$sigAvgDfr(\sigma, L) = \frac{\sum_{i=1}^{|\sigma|-1} sig(\langle \sigma(i), \sigma(i+1) \rangle, L)}{|\sigma| - 1}, \quad (5)$$

where $\sum_{i=1}^{|\sigma|-1} sig(\langle \sigma(i), \sigma(i+1) \rangle, L)$ denotes the sum of significance values of all directly-follows relations in trace σ.

Given a trace $\sigma \in L$, its significance is calculated as follows:

$$sigTotal(\sigma, L) = w_{act} \cdot sigAvgAct(\sigma, L) + w_{dfr} \cdot sigAvgDfr(\sigma, L), \quad (6)$$

where w_{act}, $w_{\text{dfr}} \geq 0$ and $w_{\text{act}} + w_{\text{dfr}} = 1$. By default, $w_{\text{act}} = w_{\text{dfr}} = 0.5$. $sigTotal$ combines $sigAvgAct$ and $sigAvgDfr$ to balance complementary signals. $sigAvgAct$ captures activity frequency, while $sigAvgDfr$ reflects structural connectivity. Equal weighting avoids bias toward either frequent activities or dense directly-follows relations, yielding a more stable and representative importance score.

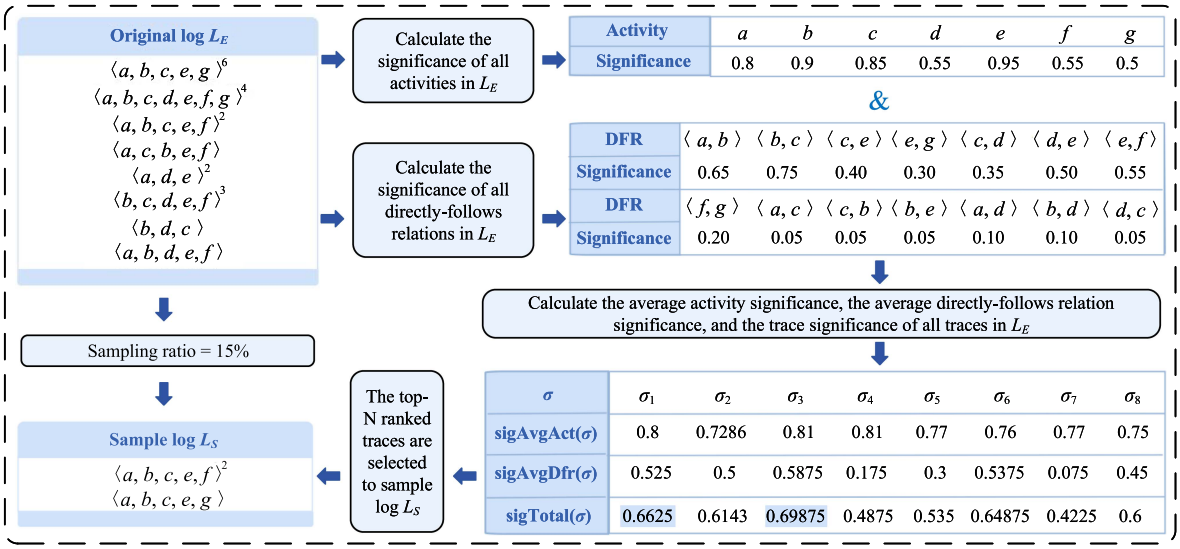


Fig. 3. An illustrative example of the sigRank sampling algorithm.

Given an arbitrary event log and a ratio, the working process of the *sigRank* sampling technique is described below.

- *Step 1:* the activity set and the directly-follows relation set are first obtained from the original event log based on Definitions 5–6;
- *Step 2:* the significance of each activity and directly-follows relation is calculated based on (2)–(3);
- *Step 3:* the significance of each trace is computed using (4)–(6);
- *Step 4:* all traces are ranked according to their significance values, and the top-N ranked traces are selected to build the sample log based on the input sample ratio.

By design, a larger *sigTotal* indicates a trace dominated by higher-frequency activities and representative directly-follows relations. During sampling, we rank traces in descending order by *sigTotal* and select the top-N according to the user-defined ratio. The *sigRank* sampling technique has an overall time complexity of $\mathcal{O}(n \log n)$, where n is the number of traces in the event log. The dominant operations include computing trace significance ($\mathcal{O}(n \cdot m)$) and sorting traces by significance ($\mathcal{O}(n \log n)$). Since $m \ll n$ in real-world event logs, the sorting step dominates the overall complexity.

An illustrative example of the sigRank working process is given in Fig. 3, by taking the example log L_C and 15% sampling ratio as input. The sample log, denoted as $L_C^{s\prime}$, is obtained, based on which we see that $L_C^{s\prime}$ contains 12 traces, 132 events, and 8 activities.

$$
\begin{aligned}
L_C^{s\prime} = [&\langle a,c,d,e,h\rangle^1, \langle a,d,c,e,h\rangle^1,\\
&\langle a,b,d,e,h\rangle^1, \langle a,c,d,e,g\rangle^1,\\
&\langle a,c,d,e,f,d,b,e,h\rangle^1, \langle a,d,c,e,f,c,d,e,h\rangle^1,\\
&\langle a,d,c,e,f,b,d,e,f,c,d,e,f,b,d,e,g\rangle^1,\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,d,b,e,g\rangle^1,\\
&\langle a,d,c,e,f,d,b,e,f,c,d,e,f,b,d,e,f,b,d,e,g\rangle^1,\\
&\langle a,d,c,e,f,d,b,e,f,b,d,e,h\rangle^1,\\
&\langle a,d,b,e,f,b,d,e,f,d,b,e,g\rangle^1,\\
&\langle a,c,d,e,f,b,d,e,f,d,b,e,g\rangle^1].
\end{aligned}
$$

C. Effectiveness Evaluation

This section introduces how to quantify the effectiveness of the sampling technique from both the sampling efficiency and the sample log quality aspects. To evaluate the quality of the sample log for process discovery purposes, we need to investigate whether the process model discovered from a sample log is good enough by comparing to the process model discovered directly from the original log directly. Therefore, it makes no sense if the model discovered in the sample log cannot guarantee 100% (replay) fitness. To meet this requirement, we select *Inductive Miner* [4] as the underlying process discovery technique, as it guarantees perfect fitness.

Three alignment-based evaluation metrics, including *fitness*, *precision*, and *F-measure*, are used to quantify the quality of the model. *Fitness* quantifies the proportion of traces that can be replayed by a process model. If all traces can be replayed using a process model, then *fitness* is 1. In this study, we adopt the definition of *fitness* as proposed in [36]. In contrast, *precision* quantifies the proportion of traces generated from the process model that can be covered by the event log. If all traces generated from a process model are covered by the event log, then the precision is 1. We adopt the definition of precision introduced in [37]. As discussed in [38], there exists a balance between *fitness* and *precision*, adding a small fraction of behavior in the event log can lead to a slight decrease in *fitness*, but the *precision* value may increase a lot. Based on the above observation, the *F-measure* is introduced.

$$F{-}measure = \frac{2 \times fitness \times precision}{fitness + precision} \tag{7}$$

The quality of a sample log against the original one from the process discovery point of view is computed as follows:

$$Quality(L_S, L_0) = \frac{F{-}measure(L_0, M_S)}{F{-}measure(L_0, M_0)}. \tag{8}$$

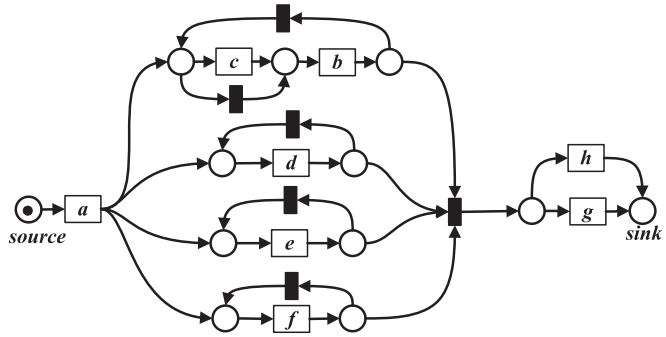


Fig. 4. Model of sample log L_C^s using IM.

To measure the quality of sampling techniques, we first need to discover the underlying process models. Consider, for example, a sample log L_C^s, the process model discovered by *IM*, denoted as M_S, is shown in Fig. 4. Similarly, the process model discovered by *IM* from sample log $L_C^{s\prime}$, denoted as M_S', is identical to that obtained from the original log in Fig. 1. This indicates that the sample log obtained by the proposed sampling technique with 15% ratio plays the same role as the original log from a process discovery point of view, i.e., the model re-discoverability is guaranteed in this case. By comparing Figs. 1 to 4, we see that their behavior varies a lot. It is argued that the selection of sampling technique and sampling ratio have a huge impact on the sample log quality as well as the discovered process models.

More specifically, we have $fitness(L_C, M_S) = 0.89$ and $fitness(L_C, M_S') = 1$, indicating that the process model obtained from the 15% sample log is capable of replaying all behavior in the original log. For precision, we have $precision(L_C, M_S) = 0.38$ and $fitness(L_C, M_S') = 0.92$. In this way, we have $F{-}measure(L_C, M_S) = 0.53$ and $F{-}measure(L_C, M_S') = 0.96$. According to Fig. 4, the model involves a flower like structure, i.e., branches can be skipped and repeated in an arbitrary number, which indicates low precision. In summary, we argue that the sample log obtained from *sigRank* technique with 15% ratio performs better than the *LogRank* sampling technique from the discovery viewpoint.

Moreover, it should be noted that the applicability and usefulness of a sampling technique heavily depends on the efficiency. The reason is that a sampling technique will be not interesting for end-users if the sampling process itself is time costly, even in the condition that the quality of discovered models is high. This paper also evaluates the running time of sampling techniques. Specifically, for a given event log, the less runtime a sampling technique requires, the more efficient the underlying technique is.

V. Tool Implementation

The novel proposed sampling technique is fully implemented as an independent tool called *sigRank-based Event Log Sampling* in the open-source framework *ProM 6*.[1] The tool produces as output an *XES*-based sample log by taking as input a large-scale event log and sampling ratio. A screenshot of the implemented plugin is given in Fig. 5.

[1]https://github.com/promworkbench/SoftwareProcessMining

Fig. 5. A screenshot of the sampling plugin.

TABLE I
Basic Statistical Information ($\#Trace$: Number of Traces, $\#Event$: Number of Events, $\#Variant$: Number of Variants, and $\#Activity$: Number of Activities)

	Event Log	#Event	#Trace	#Variant	#Activity
Synthetic Logs	trainingLog1	13608	1000	851	18
	trainingLog2	9251	1000	907	20
	trainingLog3	8197	1000	590	22
	trainingLog5	22299	1000	970	28
	trainingLog7	15187	1000	993	28
	SimulateLog2000	52000	2000	2000	26
Real-life Logs	2000AllNoise	128823	2000	2000	113
	NASA	73638	2566	2513	47
	ETMC4200	23168	4203	16	7
	order	6358	1000	9	8
	Final	21348	4580	226	14
	BPI2012_A	146044	13087	32	10

VI. Experimental Evaluation

An extensive experimental evaluation of the proposed sampling technique is performed by comparing with state-of-the-art event log sampling techniques. In particular, 12 public event logs are first introduced as datasets. Next, baseline sampling techniques are introduced for comparison. Finally, the comparative evaluation results are analyzed.

A. Datasets

This section introduces 12 public event logs, including 6 synthetic logs and 6 real-life ones to perform the comparative experiments. Table I summarizes the basic information.

B. Baseline Techniques

The following seven state-of-the-art event log sampling techniques, along with a filter-based model discovery technique, are used as baselines in the experimental evaluation.

- $LogRank$ sampling technique [24], known as one of the pioneered sampling techniques, is a graph-based ranking model that applied PageRank mechanism to compute the

importance of traces, based on which a subset is selected to construct the sample log.

- $LogRank^{+}$ sampling technique [14] defines the importance of a trace as its similarity against all remaining traces in the log. It aims to distill mainstream traces as a sample log.
- *Frequency-based* sampling technique [25] ranks the importance of traces according to their frequencies. Then, a sample log is constructed by selecting the most frequent traces.
- *Similarity-based* sampling technique [25] first computes directly-follows relations as well as their weight. A directly-follows relation is considered as the mainstream behavior of the log if its weight exceeds an input threshold. Then, the similarity of a trace is obtained by counting the number of mainstream directly-follows relations.
- *Hybrid-based* sampling technique [25] combines frequency- and similarity-based techniques. It normalizes them to values between 0 and 1 and uses a weighting average mechanism to aggregate normalized values.
- *Longer-based* sampling technique [25] sorts trace variants based on their length and chooses the longest ones. It keeps long traces in the sample log and leaves out incomplete traces.
- *Shorter-based* sampling technique [25] sorts variants based on their length and chooses the shortest ones. It may keeps variants with simpler behavior.
- *Inductive Miner Infrequent (IMi)* model discovery technique [20] extends the classical *Inductive Miner (IM)* by incorporating infrequent behavior filters at all stages, enabling localized filtering of uncommon behavior patterns during process discovery.

C. Experimental Results

For each event log, we generated sample logs using *sigRank* and seven state-of-the-art sampling techniques at ratios 30% $\rightarrow$ 5% (step -5%). To mirror this setting for the filter-based discovery technique, *IMi* was executed with complementary noise_threshold values 0.70 $\rightarrow$ 0.95 (step +0.05), corresponding to the 30% $\rightarrow$ 5% sampling ratios. These *IMi* noise thresholds were not tuned for each event log. We followed a fixed heuristic and set the threshold to $\theta = 1 - r$ for each sampling ratio r. This choice aligns the filtering aggressiveness with the degree of log reduction. Unless otherwise stated, results per dataset are averages across the six settings for each technique. To assess sample log quality, we first discover a Petri-net model from each sample log using *Inductive Miner* (*IM*), then compute *fitness*, *precision*, and *F-measure* against the original event log.

We evaluate with *IM* because it produces sound, block-structured models with deterministic outputs and exhibits strong robustness to noise, which isolates the effect of sampling. While *sigRank* is especially compatible with discovery approaches that rely on directly-follows relations (e.g., *IM* and *Split Miner*), it remains applicable to other algorithms (e.g., *Heuristics Miner*) as an algorithm-agnostic sampling layer that reduces log size while retaining salient behavior. As future work, we plan a broader cross-algorithm study, e.g., *Heuristics Miner* and ILP-based discovery. We will quantify how *sigRank* interacts with different discovery biases, e.g., causal-net and frequency-thresholding. We will also identify settings where *sigRank* provides the greatest benefit.

Detailed *F-measure* values of the *LogRank*, *LogRank*$^{+}$, *sigRank*, *Frequency-based*, *Similarity-based*, *hybrid-based*, *Longer-based*, *shorter-based* sampling techniques, and a filter-based model discovery technique *IMi* over the 12 experimental event logs are reported in Table II. In addition, the efficiency of the eight sampling techniques is evaluated by quantifying their respective sampling time. Note that we run each sampling technique five times for a given sampling ratio and report the average values. Performance comparison results are shown in Fig. 6 . According to Table II, Fig. 6 and Table III, we have the following observations.

Generally speaking, the *F-measure* quantifies the quality of the sample log. A large *F-measure* normally means a sample log with better quality. We first compare the Frequency-based sampling technique with the Similarity-based one. The former builds sample log by selecting traces with higher frequency while the latter selecting traces that are computed as more similar to the mainstream behavior in the original log. According to Table II, the quality of sample logs obtained by the *Similarity-based* technique is higher than that of the *Frequency-based* technique for most of the experimental logs (9/12). An exception is the Final log where these two techniques return sample logs with identical *F-measure* value.

Among *LogRank*, *LogRank+*, and *sigRank*, *sigRank* achieves the best sample-log quality on most datasets. Table II shows that *sigRank* attains the highest *F-measure* (10/12). In contrast, *LogRank* and *LogRank+* exhibit similar quality. Each method performs best on six event logs.

Then a comparative analysis between *sigRank* and *IMi* is also presented. As shown in Table II, *sigRank* outperforms *IMi* in most of the evaluated logs (8/12) or exhibits alternating performance depending on the sampling ratio. These results indicate that the sample logs generated by *sigRank* are of higher quality for process discovery, particularly under lower sampling ratios. *sigRank* demonstrates a stronger ability to preserve key behavioral characteristics and extract representative traces effectively.

Finally, we compare the *Similarity-based* and *sigRank* techniques in detail. In general, the *sigRank* technique performs better in most cases (8/12). As a result, the *sigRank* technique achieves the best sample quality based on the experimental evaluation. The rationale is that *sigRank* scores each trace by combining the significance of its activities and directly-follows relations, which capture core behavioral characteristics of log.

In a small subset of event logs (e.g., trainingLog7), other sampling techniques outperform *sigRank*. These logs exhibit: (i) many unique or low-frequency variants and sparse, weakly connected directly-follows relations; and (ii) rare structural patterns that frequency-weighted signals capture poorly. Under such conditions, similarity-based sampling retains a wider spectrum of structurally distinct traces and can yield better model quality. By contrast, *sigRank* prioritizes statistically dominant activities

TABLE II
QUALITY COMPARISON RESULTS (F-MEASURE–BASED SAMPLE-LOG QUALITY COMPARISON ACROSS SAMPLING RATIOS (5% - 30%, STEP -5%) ON 12 EVENT LOGS; HIGHER IS BETTER)

training-Log1	Sample Ratio						**training-Log2**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	0.8943	0.8943	0.8943	**0.8943**	**0.8943**	0.8281	IMi	0.6789	0.6789	0.6789	0.6789	0.6789	0.4882
Fre_Based	0.8126	0.8006	0.8787	0.8464	0.849	0.8351	Fre_Based	0.24438	0.6590	0.5304	0.5215	0.4632	0.5880
Sim_Based	0.915	**0.8999**	**0.8999**	0.7767	0.7767	0.8464	Sim_Based	**0.7859**	**0.7819**	0.6191	0.4502	0.4963	0.6611
hybrid	0.915	**0.8999**	**0.8999**	0.8097	0.7767	0.7767	hybrid	0.6634	0.7819	0.5896	0.5367	0.5246	**0.6937**
Longer	0.7914	0.7902	0.8186	0.8186	0.8186	0.8186	Longer	0.5812	0.4707	0.4531	0.6879	0.5574	0.6204
shorter	0.8835	0.8824	0.8824	0.8859	0.8859	**0.8859**	shorter	0.2443	0.4802	0.6512	0.7170	0.3863	0.4949
LogRank	0.7902	0.8186	0.8186	0.8186	0.8186	0.8186	LogRank	0.4147	0.4802	0.3327	0.6080	0.4831	0.5353
LogRank+	0.7902	0.7902	0.8186	0.8186	0.8186	0.8186	LogRank+	0.5014	0.6675	0.4775	**0.7275**	0.6013	0.6252
sigRank	**0.9253**	0.8344	0.8787	0.8491	0.847	0.8404	sigRank	0.7038	0.7123	**0.7135**	0.6123	**0.6802**	0.6629
training-Log3	Sample Ratio						**training-Log5**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	0.7387	0.7387	0.7387	0.8062	0.8404	0.8916	IMi	0.5929	0.5929	**0.6406**	**0.5987**	**0.5987**	0.5062
Fre_Based	0.5994	0.8773	0.8826	0.6229	0.5568	0.5568	Fre_Based	0.2097	0.1708	0.1761	0.1724	0.1661	0.1661
Sim_Based	0.8576	0.8054	0.8054	0.8153	0.5568	0.5978	Sim_Based	0.3281	0.3281	0.2546	0.1974	0.1824	0.1824
hybrid	0.8374	0.8576	0.8054	0.8153	0.5988	0.5988	hybrid	0.3281	0.3281	0.2546	0.1974	0.1824	0.1824
Longer	0.4167	0.3605	0.3605	0.3605	0.3605	0.3605	Longer	0.1865	0.1652	0.1661	0.1661	0.1661	0.1661
shorter	0.5597	0.5994	0.7343	0.4599	0.4599	0.4654	shorter	**0.6226**	**0.6226**	0.5262	0.565	0.5403	**0.5689**
LogRank	0.5994	0.5994	0.3413	0.3376	0.3376	0.3366	LogRank	0.2171	0.2092	0.2092	0.2056	0.2056	0.1904
LogRank+	0.401	0.4073	0.4073	0.4073	0.4073	0.4073	LogRank+	0.2137	0.1901	0.1661	0.1661	0.1661	0.1676
sigRank	**0.9018**	**0.9018**	**0.9018**	**0.9018**	**0.9018**	**0.9018**	sigRank	0.2754	0.2877	0.2766	0.2349	0.1857	0.1857
training-Log7	Sample Ratio						**Simulate-Log2000**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	0.5662	0.3393	0.4534	**0.4368**	**0.6347**	**0.6170**	IMi	**0.9338**	**0.9338**	**0.9071**	**0.9071**	**0.9188**	**0.9188**
Fre_Based	0.2205	0.2055	0.2018	0.2018	0.2018	0.2026	Fre_Based	0.4457	0.4458	0.4442	0.4786	0.4786	0.4787
Sim_Based	0.2085	0.2001	0.2021	0.1997	0.2085	0.2085	Sim_Based	0.5613	0.5613	0.5613	0.5613	0.5613	0.5613
hybrid	0.3074	0.2001	0.2021	0.1997	0.2085	0.2085	hybrid	0.5613	0.5613	0.5613	0.5613	0.5613	0.5613
Longer	0.2052	0.2004	0.2004	0.2004	0.2004	0.2085	Longer	0.4457	0.4458	0.4442	0.4786	0.4786	0.4787
shorter	**0.7202**	**0.4811**	**0.6061**	0.3323	0.2566	0.238	shorter	0.4457	0.4458	0.4442	0.4786	0.4786	0.4787
LogRank	0.3593	0.253	0.2218	0.2004	0.1997	0.1997	LogRank	0.5678	0.5678	0.5235	0.4786	0.5154	0.5154
LogRank+	0.2457	0.2337	0.2174	0.1919	0.2036	0.2036	LogRank+	0.4824	0.4825	0.5677	0.4785	0.4785	0.4787
sigRank	0.6935	0.3769	0.2367	0.1927	0.2118	0.2342	sigRank	0.5613	0.5613	0.5613	0.5613	0.5613	0.5613
2000All-Noise	Sample Ratio						**NASA**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	**0.8490**	**0.8490**	**0.8490**	**0.8490**	**0.8490**	**0.8490**	IMi	0.1366	0.1333	0.1333	0.1333	0.1333	0.1481
Fre_Based	0.7072	0.7159	0.7273	0.7273	0.7363	0.7363	Fre_Based	0.3683	0.2333	0.2333	0.2333	0.2333	0.2333
Sim_Based	0.6099	0.7273	0.7363	0.7363	0.7363	0.7363	Sim_Based	0.346	0.3361	0.3417	0.342	0.3421	0.3528
hybrid	0.6125	0.7159	0.7273	0.7791	0.7791	0.7791	hybrid	0.346	0.3361	0.3417	0.342	**0.3528**	0.3528
Longer	0.7017	0.7226	0.7226	0.7226	0.8305	0.8414	Longer	0.2105	0.2105	0.2105	0.2105	0.2105	0.2105
shorter	0.6105	0.6105	0.6105	0.6105	0.6105	0.6105	shorter	0.3432	0.349	0.3422	0.3422	0.3422	**0.368**
LogRank	0.6403	0.6403	0.6826	0.7363	0.7363	0.7363	LogRank	**0.3696**	**0.3597**	**0.3886**	0.2425	0.2425	0.2333
LogRank+	0.1888	0.5015	0.6037	0.6045	0.6394	0.6924	LogRank+	0.2105	0.2105	0.2105	0.2105	0.2105	0.2105
sigRank	0.3146	0.6878	0.7379	0.828	0.8341	0.8463	sigRank	0.3146	0.3587	0.3489	**0.349**	0.3424	0.3424
ETMC4200	Sample Ratio						**order**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	0.9773	0.9773	0.9773	0.9773	0.9773	0.9773	IMi	**0.8938**	**0.8938**	**0.8938**	**0.8938**	**0.8363**	**0.8363**
Fre_Based	0.867	0.867	0.9251	0.9251	0.9251	0.9762	Fre_Based	0.4351	0.4351	0.4351	0.4351	0.689	0.689
Sim_Based	0.867	0.867	0.9236	0.9236	0.9762	0.9762	Sim_Based	0.1584	0.5337	0.5337	0.5337	0.5337	0.5337
hybrid	0.867	0.867	0.9251	0.9251	0.9251	0.9762	hybrid	0.5706	0.5706	0.5706	0.5706	0.5706	0.5706
Longer	0.9284	0.9773	0.9773	0.9773	0.9773	0.9773	Longer	0.7097	0.7097	0.7097	0.7097	0.7097	0.7097
shorter	0.967	0.867	0.9236	0.9236	0.9762	0.9762	shorter	0.6837	0.6837	0.6837	0.6837	0.6837	0.6837
LogRank	0.9036	0.9161	0.9773	0.9773	0.9773	0.9773	LogRank	0.214	0.214	0.3512	0.3512	0.4009	0.4645
LogRank+	0.9773	0.9773	0.9773	0.9773	0.9773	0.9773	LogRank+	0.5295	0.5295	0.5295	0.5295	0.5295	0.5295
sigRank	**1**	**1**	**1**	**1**	**1**	**1**	sigRank	0.8076	0.8076	0.8076	0.8076	0.8076	0.8076
Final	Sample Ratio						**BPI2012_A**	Sample Ratio					
Techniques	5%	10%	15%	20%	25%	30%	Techniques	5%	10%	15%	20%	25%	30%
IMi	0.6789	0.6789	0.6789	0.7717	0.7717	0.7717	IMi	0.5050	0.5050	0.5050	0.5050	0.5050	0.5050
Fre_Based	0.9551	0.9551	0.9551	0.9551	0.9551	0.9551	Fre_Based	**0.5251**	**0.569**	0.3892	0.3942	0.3479	0.3124
Sim_Based	0.9551	0.9551	0.9551	0.9551	0.9551	0.9551	Sim_Based	0.1556	0.2633	0.2593	0.2593	0.2672	0.286
hybrid	0.9551	0.9551	0.9551	0.9551	0.9551	0.9551	hybrid	0.2593	0.2593	0.2593	0.2593	0.2672	0.2672
Longer	0.6099	0.6099	0.5768	0.6646	0.6645	0.6646	Longer	0.1556	0.2633	0.2593	0.2593	0.2672	0.286
shorter	0.9513	0.9513	0.9513	0.9513	0.9513	0.9513	shorter	0.1782	0.1782	0.2157	0.2593	0.2672	0.286
LogRank	0.641	0.6096	0.6373	0.6015	0.6015	0.6015	LogRank	0.0926	0.2622	0.2622	0.3128	0.3128	0.3128
LogRank+	0.6312	0.6312	0.6295	0.6295	0.6295	0.6295	LogRank+	0.2622	0.2622	0.2622	0.4352	0.4352	0.4352
sigRank	**0.9712**	**0.9712**	**0.9629**	**0.9629**	**0.9629**	**0.9629**	sigRank	0.5243	0.5243	**0.5243**	**0.5243**	**0.5243**	**0.5243**

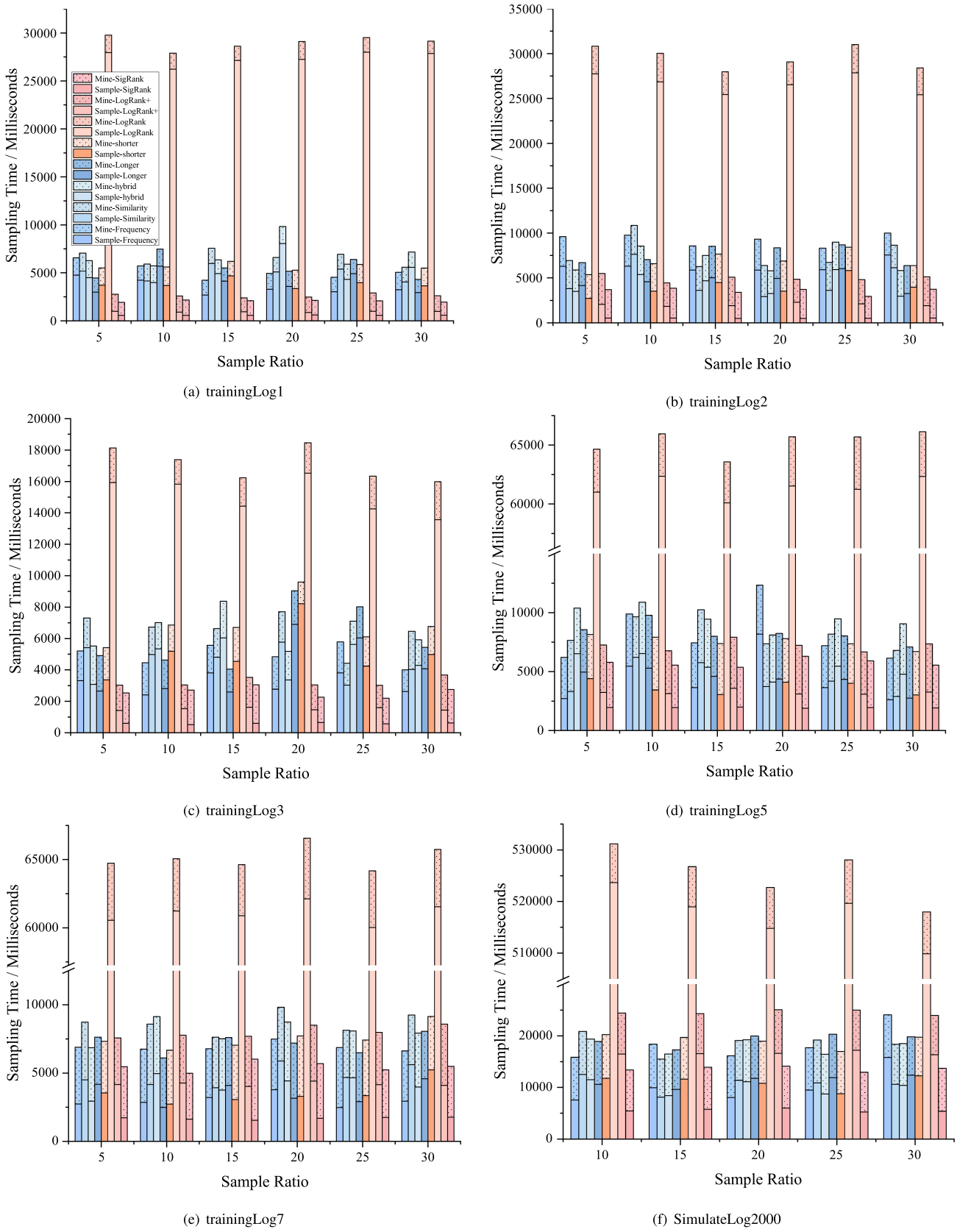
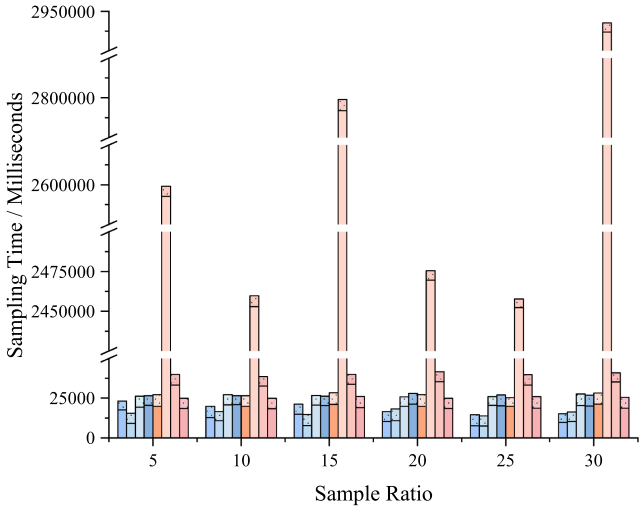
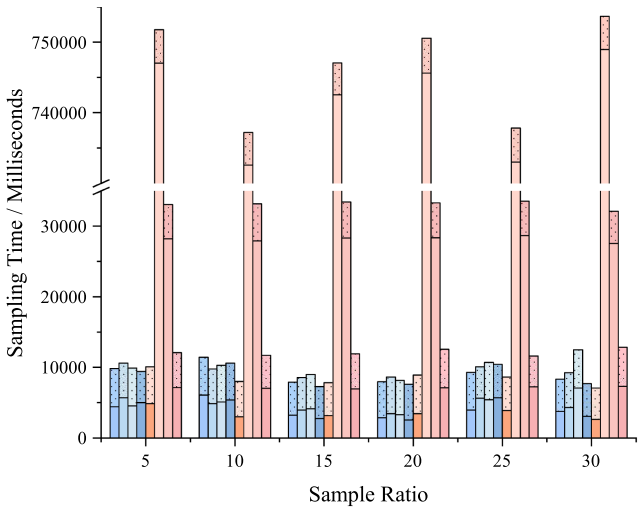


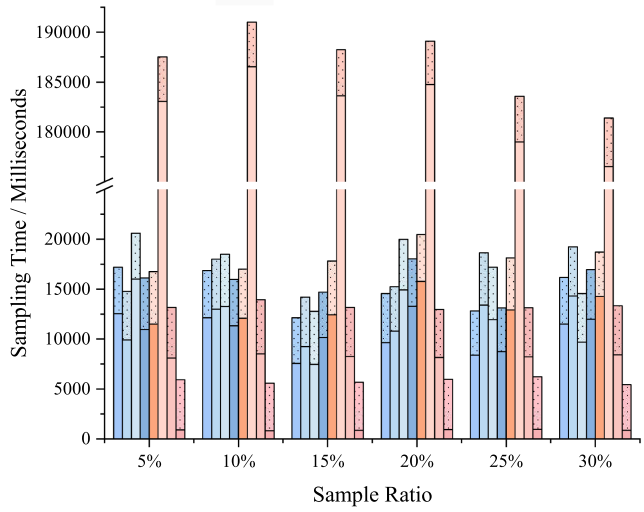
Fig. 6. Time performance comparison results.



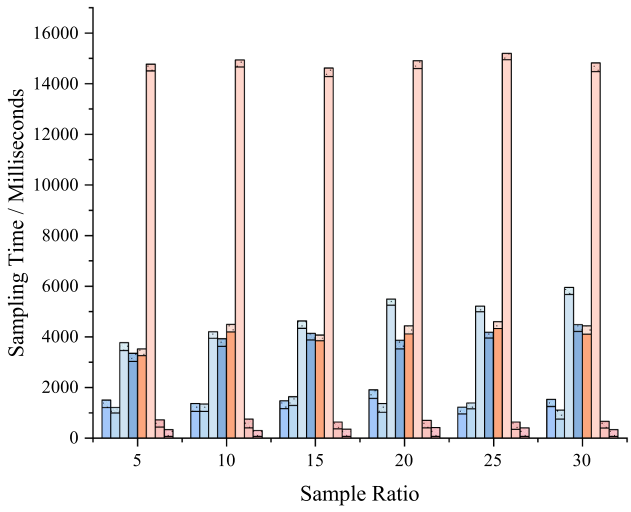
(g) 2000AllNoise



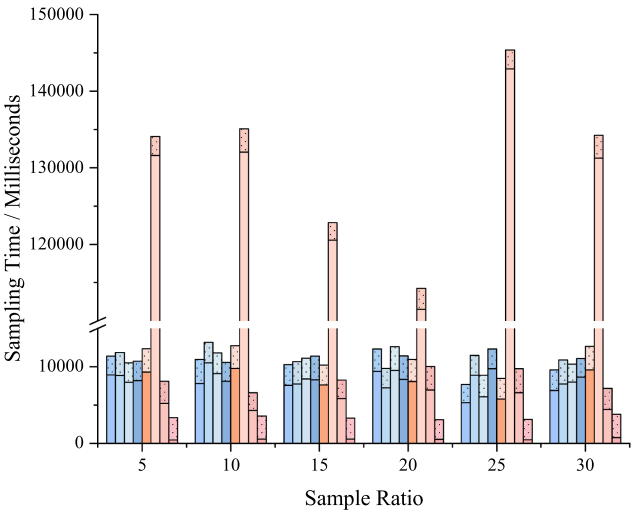
(h) NASA



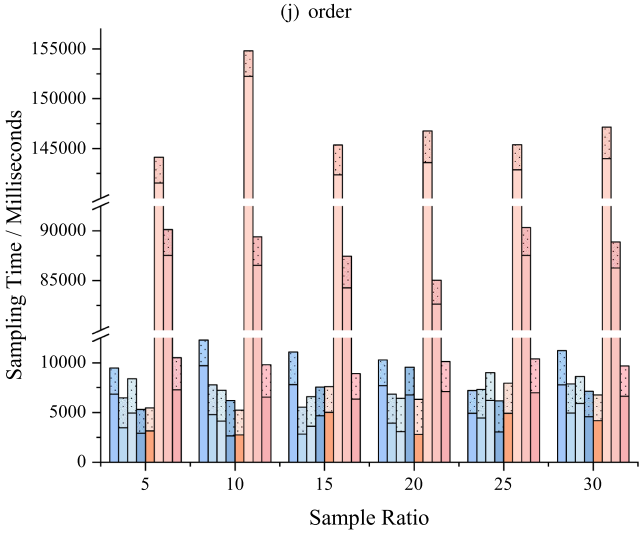
(i) ETMC4200



(j) order



(k) Final



(l) BPI2012_A

Fig. 6. *Continued.*

TABLE III
OVERALL TIME PERFORMANCE (SUMMARY OF TIME PERFORMANCE IN MILLISECONDS: SAMPLING TIME PLUS DISCOVERY TIME, AVERAGED OVER SAMPLING RATIOS FROM 30% TO 5% (STEP -5%) FOR EACH EVENT LOG; LOWER IS BETTER.)

Average Overall Time(ms)	Fre_Based	Sim_Based	hybrid	Longer	shorter	LogRank	LogRank+	sigRank
trainingLog1	5171.16	6608.83	6876.83	5557.66	5665.16	29018.66	2625.49	**2057.99**
trainingLog2	9257.16	7638.33	7095.32	7618.83	6886.99	29548.99	4972.16	**3564.5**
trainingLog3	4978.66	6541.66	6518.16	6013.99	6907.5	17089.66	3225	**2585.33**
trainingLog5	8201.82	8309.33	9554.49	8277.66	7545.16	65275.49	7194.33	**5739.16**
trainingLog7	6906.32	8700.16	8047.32	7188.16	7560.33	65152.82	8024.99	**5486**
SimulateLog2000	19137.82	18623.33	18232.66	21001.66	20523.66	524498.5	24589.66	**13539.83**
2000AllNoise	18426.49	**15846.49**	26591.66	26850.16	27145.66	2622291.82	40099.99	25360.82
NASA	9125	9489.49	10089.99	8839.49	**8426.66**	746341.5	33092.16	12119.83
ETMC4200	14949.99	16685.32	17269.49	15819.49	18144.66	186804.32	13286.66	**5799.49**
order	1499.16	1341.32	4875.33	3989.66	4257.49	14872.16	685.66	**359.33**
Final	10367.82	11319.83	10884.49	11258.49	11240.16	130979.33	8325.82	**3380.5**
BPI2012_A	10260.32	6961.66	7705.16	6989.83	**6559.16**	147261.16	88527.16	9899.32

and relations to ensure scalability, which can under-sample rare but informative variants. In a few datasets with substantial noise or unstable control-flow, *IMi* yields higher quality results by filtering infrequent behavior during discovery. It often produces simpler models. This comes at the cost of potential information loss, sensitivity to the noise threshold, and the absence of a reusable sampled log. Overall, these cases indicate that *sigRank* may be less suitable when rare variants are semantically critical or when the log exhibits high behavioral diversity with weakly connected directly-follows relations. In such settings, diversity-oriented sampling or discovery-time filtering can be preferable, depending on whether the objective is to preserve rare behavior or to simplify noisy control-flow. These exceptions do not contradict the strengths of *sigRank* on large and complex logs. They highlight different objectives across techniques. Similarity-based sampling prioritizes behavioral diversity. *IMi* emphasizes noise filtering during discovery. *sigRank* provides efficient and representative pre-discovery reduction.

Efficiency of different sampling techniques is quantified by the total time for sampling and model discovery. The shorter the time required, the higher the sampling efficiency. According to Fig. 6, the *LogRank* requires longer time than others. The efficiency is even worse for large event logs, e.g., for the BPI2012_A log, the sampling time of *LogRank* is 141549 ms, which is approximately 100 times longer than other techniques. It is because *LogRank* computes the similarity between each pair of trace and then implements a PageRank-based ranking strategy that is relatively complex and time consuming. The *sigRank* achieves the best performance than others for most cases (9/12). Exceptions are the 2000AllNoise, NASA and BPI2012 A logs where the Frequency-based, Similarity-based, hybrid-based, shorter-based and Longer-based techniques show roughly the same efficiency with the *sigRank* (10296 ms vs 9127 ms vs 12006 ms, 6093 ms vs 5690 ms vs 6544 ms, 4933 ms vs 4448 ms vs 6217 ms vs 3040 ms vs 4929 ms vs 6984 ms). The rationale behind is that all these three techniques require to traverse the event log once trace-by-trace to obtain their importance values based on different strategies.

To provide an overall view of efficiency comparison, Table III reports each sampling technique's average total time (sampling time + model discovery time) across all twelve event logs. Values are averaged over six sampling ratios (from 5% to 30%), and the most efficient technique for each dataset is boldfaced. As shown in Table III, *sigRank* achieves the lowest average total time in 9/12 datasets, confirming superior efficiency compared to state-of-the-art techniques.

In summary, it is shown that the *sigRank* sampling technique achieves the best sample log quality and sampling efficiency according to the comparative evaluation against state-of-the-art sampling techniques.

D. Threats to Validity

In this section, we discuss the primary factors that may affect the validity and generalizability of the proposed approach.

- The effectiveness of the proposed *sigRank* is based on the assumption that activities and directly-follows relations with higher frequency reflect more significant behavior. This assumption enables efficient log reduction. However, it can omit infrequent yet important behavior and overlook critical aspects of the global process. This issue is particularly relevant when rare variants are semantically critical, such as compliance violations or risk-sensitive exceptions. In such settings, *sigRank* may not be the most suitable choice.
- The experimental evaluation is conducted on a limited number of publicly available event logs. Although these logs are selected to cover a range of scenarios, they may not fully capture the diversity and complexity of real-world industrial processes. As a result, the generalizability of the proposed approach to other domains and larger-scale event logs may be constrained; and
- The experimental analysis evaluates *sigRank* using fixed sampling ratios ranging from 5% to 30%. However, these predefined ratios may not generalize well across different event logs, as the optimal sampling ratio can vary significantly depending on the characteristics of the dataset. The effectiveness of alternative or adaptive sampling strategies remains an open question for future investigation.

VII. CONCLUSION

We introduce a novel event log sampling technique named sigRank. It improves sampling efficiency and sample log quality.

More specifically, we give the general design of the proposed technique. In addition, the proposed sampling technique is integrated as an independent tool into the open-source ProM framework, and experimental evaluation using 12 public event logs illustrated that the proposed *sigRank* sampling technique can significantly accelerate sampling efficiency while guaranteeing superior sample log quality for process discovery, compared to existing event log sampling techniques.

Our future work mainly lies in extending the applicability of the proposed *sigRank* sampling technique to a broader range of process mining tasks, e.g., hierarchical process discovery [39], [40], cross-organization process mining [30], conformance checking [41], [42], and process prediction [43], [44], [45], [46]. In addition, developing novel sampling techniques that preserve behavioral invariants remains a critical direction for future research. We also intend to systematically evaluate the performance of *sigRank* and other sampling strategies across various process model discovery paradigms. In addition, we plan to incorporate structural complexity metrics, such as the *Extended Cardoso* metric [47], to enable a more comprehensive assessment of the trade-offs between model quality, complexity, and efficiency.

REFERENCES

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

[2] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, 2004.

[3] A. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2011, pp. 310–317.

[4] S. J. Leemans, D. Fahland, and W. van der Aalst, "Discovering block-structured process models from event logs-a constructive approac," in *Application and Theory of Petri Nets and Concurrency*. Berlin, Germany: Springer, 2013, pp. 311–329.

[5] C. Liu, L. Cheng, Q. Zeng, and L. Wen, "Formal modeling and discovery of hierarchical business processes: A petri net-based approach," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 53, no. 2, pp. 1003–1014, Feb. 2023.

[6] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Split miner: Automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, pp. 251–284, 2019.

[7] C. Liu, H. Li, S. Zhang, L. Cheng, and Q. Zeng, "Cross-department collaborative healthcare process model discovery from event logs," *IEEE Trans. Automat. Sci. Eng.*, vol. 20, no. 3, pp. 2115–2125, Jul. 2023.

[8] C. Liu, Y. Pei, L. Cheng, Q. Zeng, and H. Duan, "Sampling business process event logs using graph-based ranking model," *Concurrency Comput.: Pract. Exp.*, vol. 33, no. 5, pp. 1–14, 2021.

[9] L. Cheng, B. F. van Dongen, and W. M. van der Aalst, "Scalable discovery of hybrid process models in a cloud computing environment," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 368–380, Mar./Apr. 2020.

[10] L. Cheng, B. van Dongen, and W. van der Aalst, "Efficient event correlation over distributed systems," in *Proc. IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2017, pp. 1–10.

[11] J. Evermann, "Scalable process discovery using map-reduce," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 469–481, May/Jun. 2016.

[12] X. Su, C. Liu, S. Zhang, and Q. Zeng, "Sampling business process event logs with guarantees," *Concurrency Comput.: Pract. Exp.*, vol. 36, no. 13, 2024, Art. no. e8077.

[13] X. Su, C. Liu, F. Lu, L. Cheng, Q. Zeng, and S. Zhang, "Edgeim: An efficient edge-based process model discovery technique," in *Proc. 2025 IEEE Int. Conf. Web Serv.*, 2025, pp. 1–7.

[14] C. Liu, Y. Pei, Q. Zeng, H. Duan, and F. Zhang, "Logrank^{+}: A novel approach to support business process event log sampling," in *Proc. Int. Conf. Web Inf. Syst. Eng.*, 2020, pp. 417–430.

[15] L. Wen, W. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discov.*, vol. 15, no. 2, pp. 145–180, 2007.

[16] X. Su, C. Liu, F. Lu, L. Cheng, Q. Zeng, and J. Zhou, "Enhancing healthcare process model discovery through duplicate task identification," in *Proc. 2025 IEEE Int. Conf. Web Serv.*, 2025, pp. 477–483.

[17] L. Wen, J. Wang, W. van der Aalst, B. Huang, and J. Sun, "A novel approach for process mining based on event types," *J. Intell. Inf. Syst.*, vol. 32, no. 2, pp. 163–190, 2009.

[18] L. Wen, J. Wang, W. M. van der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks," *Data Knowl. Eng.*, vol. 69, no. 10, pp. 999–1021, 2010.

[19] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[20] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Proc. Int. Conf. Bus. Process Manage.*, 2013, pp. 66–78.

[21] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *Proc. Int. Conf. Appl. Theory Petri Nets Concurrency*, 2014, pp. 91–110.

[22] C. Liu, "Hierarchical business process discovery: Identifying sub-processes using lifecycle information," in *Proc. IEEE Int. Conf. Web Serv.*, 2020, pp. 423–427.

[23] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[24] C. Liu, Y. Pei, Q. Zeng, and H. Duan, "Logrank: An approach to sample business process event log for efficient discovery," in *Proc. Int. Conf. Knowl. Sci., Eng. Manage.*, 2018, pp. 415–425.

[25] M. F. Sani et al., "Improving the performance of process discovery algorithms by instance selection," *Comput. Sci. Inf. Syst.*, vol. 17, no. 3, pp. 927–958, 2020.

[26] V. Pasquadibisceglie, A. Appice, G. Castellano, and W. van der Aalst, "Promise: Coupling predictive process mining to process discovery," *Inf. Sci.*, vol. 606, pp. 250–271, 2022.

[27] M. Fani Sani, M. Boltenhagen, and W. van der Aalst, "Prototype selection using clustering and conformance metrics for process discovery," in *Proc. Bus. Process Manage. Workshops: BPM 2020 Int. Workshops*, Seville, Spain, 2020, pp. 281–294.

[28] C. Liu, Q. Zeng, L. Cheng, H. Duan, and J. Cheng, "Privacy-preserving behavioral correctness verification of cross-organizational workflow with task synchronization patterns," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 3, pp. 1037–1048, Mar. 2021.

[29] C. Liu, Q. Zeng, L. Cheng, H. Duan, and C. Jiujun, "Measuring similarity for data-aware business processes," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 2, pp. 1070–1082, Apr. 2022.

[30] C. Liu, H. Duan, Z. Qingtian, M. Zhou, F. Lu, and J. Cheng, "Towards comprehensive support for privacy preservation cross-organization business process mining," *IEEE Trans. Services Comput.*, vol. 12, no. 4, pp. 639–653, Jul./Aug. 2019.

[31] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Ylä-Jääski, "Automatic composition of semantic web services based on fuzzy predicate petri nets," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 2, pp. 680–689, Apr. 2015.

[32] C. Liu, "Automatic discovery of behavioral models from software execution data," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1897–1908, Oct. 2018.

[33] H. Duan, C. Liu, Q. Zeng, and M. Zhou, "Refinement-based hierarchical modeling and correctness verification of cross-organization collaborative emergency response processes," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 8, pp. 2548–2559, Aug. 2020.

[34] Q. Zeng, C. Liu, H. Duan, and M. Zhou, "Resource conflict checking and resolution controller design for cross-organization emergency response processes," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 10, pp. 3685–3700, Oct. 2020.

[35] W. Reisig, *Petri Nets: An Introduction*, vol. 4. Berlin, Germany: Springer, 2012.

[36] A. Adriansyah, B. F. van Dongen, and W. M. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. IEEE 15th Int. Enterprise Distrib. Object Comput. Conf.*, 2011, pp. 55–64.

[37] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," in *Proc. Int. Conf. Bus. Process Manage.*, 2012, pp. 137–149.

[38] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A robust f-measure for evaluating discovered process models," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2011, pp. 148–155.

[39] C. Liu, Y. Wang, L. Wen, J. Cheng, L. Cheng, and Q. Zeng, "Discovering hierarchical multi-instance business processes from event logs," *IEEE Trans. Services Comput.*, vol. 17, no. 1, pp. 142–155, Jan./Feb. 2024.

[40] C. Liu, "Formal modeling and discovery of multi-instance business processes: A cloud resource management case study," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 12, pp. 2151–2160, Dec. 2022.
[41] L. Cheng, C. Liu, and Q. Zeng, "Optimal alignments between large event logs and process models over distributed systems: An approach based on petri nets," *Inf. Sci.*, vol. 619, pp. 406–420, 2023.
[42] M. Bauer, H. van der Aa, and M. Weidlich, "Estimating process conformance by trace sampling and result approximation," in *Proc. Int. Conf. Bus. Process Manage.*, 2019, pp. 179–197.
[43] N. Guo et al., "Business process remaining time prediction based on incremental event logs," *IEEE Trans. Services Comput.*, vol. 18, no. 3, pp. 1308–1320, May/Jun. 2025.
[44] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, "Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 4, pp. 1–34, 2019.
[45] N. Guo et al., "Explainable and effective process remaining time prediction using feature-informed cascade prediction model," *IEEE Trans. Services Comput.*, vol. 17, no. 3, pp. 949–962, May/Jun. 2024.
[46] L. Cheng, L. Du, C. Liu, Y. Hu, F. Fang, and T. Ward, "Multi-modal fusion for business process prediction in call center scenarios," *Inf. Fusion*, vol. 108, 2024, Art. no. 102362.
[47] K. B. Lassen and W. M. van der Aalst, "Complexity metrics for workflow nets," *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 610–626, 2009.

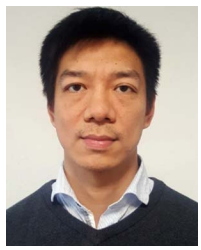
Xuan Su (Graduate Student Member, IEEE) received the BS and the MS degree in the School of Computer Science and Technology from Shandong University of Technology, Zibo, China. She is currently working towards the PhD degree in Shandong University of Science and Technology. Her research interests are in the areas of business process management and process mining.

Cong Liu (Member, IEEE) received the BS and MS degrees in computer software and theory from the Shandong University of Science and Technology, Qingdao, China, in 2013 and 2015 respectively, and the PhD degree in the Department of Mathematics and Computer Science, Eindhoven University of Technology, 2019. He is an invited full professor in the NOVA Information Management School, Nova University of Lisbon. His research interests are in the areas of process mining, business process management, and artificial intelligence.

Shuaipeng Zhang received the BS degree from Zhengzhou University, Zhengzhou, China in 2020, and the MS degree in school of computer science and technology from Shandong University of Technology, Zibo, China, in 2023. He is currently working towards the PhD degree in Shandong University. His research interests are in the areas of blockchain, federated learning, and process mining.

Qingtian Zeng (Member, IEEE) received the PhD degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He is currently a professor with Shandong University of Science and Technology, Qingdao, China. His research interests are in the areas of Petri nets, process mining, and knowledge management.

Qi Mo received the PhD degree from Yunnan University, Kunming, China, in 2015. He is currently an associate professor with the school of software in Yunnan University. His research interests include formal methods, software engineering, business process management, and emergency management.



Long Cheng (Senior Member, IEEE) received the PhD degree from the National University of Ireland Maynooth, in 2014. He is a full professor with North China Electric Power University, in Beijing. He was an assistant professor with Dublin City University, and a Marie Curie fellow with University College Dublin. He has published more than 110 papers in refereed journals and conferences. His research focuses on distributed computing. He is an associate editor of *IEEE Transactions on Consumer Electronics*, and a *Chair of Journal of Cloud Computing*.