Preprint Computing, 103(6):1085-1104, 2021

# The impact of biased sampling of event logs on the performance of process discovery

**Mohammadreza Fani Sani**[1] · **Sebastiaan J. van Zelst**[1,2] · **Wil M. P. van der Aalst**[1,2]

Received: 22 May 2020 / Accepted: 18 January 2021


**Abstract**
With Process discovery algorithms, we discover process models based on event data, captured during the execution of business processes. The process discovery algorithms tend to use the whole event data. When dealing with large event data, it is no longer feasible to use standard hardware in a limited time. A straightforward approach to overcome this problem is to down-size the data utilizing a random sampling method. However, little research has been conducted on selecting the right sample, given the available time and characteristics of event data. This paper systematically evaluates various biased sampling methods and evaluates their performance on different datasets using four different discovery techniques. Our experiments show that it is possible to considerably speed up discovery techniques using biased sampling without losing the resulting process model quality. Furthermore, due to the implicit filtering (removing outliers) obtained by applying the sampling technique, the model quality may even be improved.

**Keywords** Process mining · Biased sampling · Process discovery · Event log preprocessing · Performance enhancement

✉ Mohammadreza Fani Sani
fanisani@pads.rwth-aachen.de

Sebastiaan J. van Zelst
s.j.v.zelst@pads.rwth-aachen.de

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Process and Data Science Chair, RWTH Aachen University, Aachen, Germany

[2] Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany

## 1 Introduction

Process mining is a research discipline that provides both data-oriented and business process oriented analysis at the same time. Process discovery, one of the main branches of process mining, aims to discover a process model that accurately describes the underlying process captured within the event data [1]. In conformance checking, the goal is to assess to what degree a given process model and event data conform to one another. Finally, process enhancement aims at improving or enhancing process mining results, e.g., by reflecting bottleneck information directly onto a (given) process model.

The result of process discovery algorithms can be used by other process mining branches like process simulation [2] and prediction [3]. Currently, the main research focus in process discovery is on quality issues of discovered process models. However, the ever-increasing size of the data handled by the process mining algorithms leads to performance issues when applying the existing process discovery algorithms [4]. Most process discovery algorithms first build an internal data structure, based on the whole event log, then an optional filter step is applied.

However, such an approach may be infeasible in big data settings, where the event data are too large to process. Moreover, some process mining tools impose constraints on the size of event data, e.g., the number of events. Also, in many cases, we do not require the whole event log, and an approximation of the process can already be discovered by only using a small fraction of the event data.

In real life, process discovery is often of an exploratory nature, that means sometimes we need to apply different process discovery algorithms with several parameters to generate different process models and select the most suitable process model. When the discovery algorithms are used repeatedly, such an exploratory approach makes sense only if performance is reasonable. Thus, even a small performance improvement may accumulate to a significant performance increase when applied several times. Furthermore, many process discovery algorithms are designed to generalize the behavior observed in the event data. In other words, these algorithms are able to reproduce process behavior extends beyond the example behavior used as input. Therefore, it may still be possible to discover the underlying process using a subset of event data.

This research studies the effectiveness of applying biased sampling on event data in advance of invoking process discovery algorithms, instead of using all the available event data. In this regard, we present and investigate different biased sampling strategies and analyze their ability to improve process discovery algorithm scalability. Furthermore, the techniques presented allow us to select a user-specified fraction of inclusion of the total available event data. Utilizing the `ProM`-based [5] extension of `RapidMiner` [6], i.e., `RapidProM`, we study the usefulness of these sampling approaches, using real event logs. The experimental results show that applying biased sampling techniques reduces the required discovery time for all the evaluated discovery algorithms.

This paper extends the work in [7]. Here, we formally define the proposed method and explain it with more details. The proposed method is also applied on many real event logs with state-of-the-art process discovery algorithms, i.e., Inductive Miner, Split Miner and ILP Miner. We return sampled event logs based on variants or traces. Finally, it is shown that using variant-based sampling, we are able to improve the per-

formance of process discovery procedure using different process discovery algorithms and at the same time having high-quality process models.

The remainder of this paper is structured as follows. In Sect. 2, we discuss related work. Section 3 defines preliminary notation. We present different biased sampling strategies in Sect. 4. The evaluation and corresponding results are given in Sect. 5. Finally, Sect. 6 concludes the paper and presents some directions for future work.

## 2 Related work

Most process model discovery algorithms, e.g., Alpha Miner [8] and the basic Inductive Miner [9] were designed to depict as much as possible behaviors seen in the event log into the process model. Existence of high behavior variability in real event logs leads this approach to result in complex and imprecise process models [10]. Other process discovery algorithms, e.g., Split Miner [11], the extended versions of Inductive Miner [12], and ILP Miner [13] were designed to capable of filtering infrequent behavior within their internal data structure, in advance of discovering a process model. The performance of all these methods depends on different parameters, e.g., the number of (unique) process instances, the number of unique activities, and the average length of process instances in the given event log.

Two main categories of methods are proposed in the literature to address this problem. In the first category, outlier behaviors [10], uncertainty about the execution and the order of activities [14], and missing data [15] are considered as the main reasons of the behavioral variability in the process event log. Therefore, they aim to detect such behaviors and remove them from the event log and give the preprocessed event logs to process discovery algorithms. However, in the second direction, the goal is to consider the mainstream behavior in the event log in a fast way to be able to discover a process model similar to the one that is discovered using the original event log. In the following, we provide some works in each category.

There are different methods to detect and deal with outlier behavior in event logs. In [10,16,17] the authors propose to remove outlier behavior that is detected in the event log. However, in [18,19] the authors propose to apply automated ordering modification algorithms. Moreover, [20] provides an interactive approach to repair noisy activity labels. In addition, the authors in [21,22] propose to consider activities that could be executed in different parts of the process that lead to less structure in the discovered process model. In [23] an interactive filtering toolkit is provided that let user choose different filtering methods in combination with several process discovery algorithms. Filtering techniques effectively reduce the size of given process instances (i.e., traces) used by process discovery algorithms. In this regard, these filtering techniques have non-linear time complexity that does not scale in the context of big data. However, sometimes the required time for applying these filtering algorithms is longer than the required time of discovering a process model from the original event log. Also, these filtering techniques have no accurate control over the size of the reduced event log.

Sampling methods reduce the number of process instances and increase the performance of different process mining algorithms [24]. Moreover, they can improve the confidentiality aspects of event logs [25]. In [26], the authors proposed a sam-

pling approach based on Parikh vector of traces to detect the behavior in the event log. However, we can not use this sampling technique for process discovery purpose; because the Parikh vector does not store the sequences of activities that are critical for discovering process models. In [27], the authors recommend a random trace-based sampling method to decrease the discovery time and memory footprint. This method assumes that process instances have different behavior if they have different sets of directly follows relations. However, using a unique set of directly follows relations may show different types of process behavior. Furthermore, [28] recommends a trace-based sampling method specifically for the Heuristic miner [29]. Both [27] and [28] have no control on the size of the final sampled event data. Also, they depend on the defined behavioral abstraction that may lead to the selection of almost all the process instances.

Moreover, as these methods are unbiased, we have non-deterministic results after each sampling. In this paper, we will offer and analyze random and biased sampling methods in which the size of the sampled event data is adjustable. Therefore, we can control the size and variability of process models at the same time.

## 3 Preliminaries

In this section, we briefly introduce basic process mining terminologies and notations that ease the readability of the paper.

Given a set $X$, a multiset $\mathbb{B}$ over $X$ is a function $\mathbb{B}: X \rightarrow \mathbb{N}_{\geq 0}$, i.e, it allows certain elements of $X$ to appear multiple times. We show a multiset as $\mathbb{B}=[e_1^{k_1}, e_2^{k_2}, ..., e_n^{k_n}]$, where for $1 \leq i \leq n$ we have $\mathbb{B}(e_i)=k_i$ with $k_i \in \mathbb{N}_{>0}$. If $k_i=1$, we do not show its superscript, and if for some $e \in X$ we have $\mathbb{B}(e)=0$, we omit it from the multiset notation. Furthermore, the empty multiset, i.e. $\mathbb{B}(e)=0, \forall e \in X$ is written as [ ]. Moreover, $\overline{\mathbb{B}}=\{e \in X \mid \mathbb{B}(e)>0\}$ is the set of all elements that are presented in the multiset. The set of all possible multisets over a set $X$ is written as $\mathbb{B}(X)$.

Let $X^*$ denote the set of all possible sequences over a set $X$. A finite sequence $\sigma$ of length $n$ over $X$ is a function $\sigma: \{1, 2, ..., n\} \rightarrow X$, alternatively written as $\sigma = \langle x_1, x_2, ..., x_n \rangle$ where $x_i=\sigma(i)$ for $1 \leq i \leq n$. The empty sequence is written as $\epsilon$. The concatenation of sequences $\sigma$ and $\sigma'$ is written as $\sigma \cdot \sigma'$. Function $hd: X^* \times \mathbb{N}_{\geq 0} \nrightarrow X^*$, returns the "head" of a sequence, i.e., given a sequence $\sigma \in X^*$ and $k \leq |\sigma|$, $hd(\sigma, k) = \langle x_1, x_2, .., x_k \rangle$, i.e., the sequence of the first $k$ elements of $\sigma$. In case $k=0$, we have $hd(\sigma, 0)=\epsilon$, i.e., an empty sequence. Symmetrically, $tl: X^* \times \mathbb{N}_{\geq 0} \nrightarrow X^*$ returns the "tail" of a sequence and is defined as $tl(\sigma, k)=\langle x_{n-k+1}, x_{n-k+2}, ..., x_n \rangle$, i.e., the sequence of the last $k$ elements of $\sigma$, with, again, $tl(\sigma, 0)=\epsilon$. Sequence $\sigma'$ is a subsequence of sequence $\sigma$, which we denote as $\sigma' \in \sigma$, if and only if $\sigma_1, \sigma_2 \in X^*$ such that $\sigma=\sigma_1 \cdot \sigma' \cdot \sigma_2$. Let $\sigma, \sigma' \in X^*$. We define the frequency of occurrence of $\sigma'$ in $\sigma$ by $freq: X^* \times X^* \rightarrow \mathbb{N}_{\geq 0}$ where $freq(\sigma', \sigma)=|\{1 \leq i \leq |\sigma| \mid \sigma'_1=\sigma_i, ..., \sigma'_{|\sigma'|}=\sigma_{i+|\sigma'|}\}|$. For example, $freq(\langle b \rangle, \langle a, b, b, c, d, e, f, h \rangle)=2$ and $freq(\langle b, d \rangle, \langle a, b, d, c, e, g \rangle)=1$, etc.

Event logs describe sequences of executed business process activities, typically in the context of some cases (or process instances), e.g., a customer or an order-id. The

execution of an activity in the context of a case is referred to as an *event*. A sequence of events for a specific case is also referred to as a *trace*. Thus, it is possible that multiple traces describe the same sequence of activities, yet, since events are unique, each trace itself contains different events.

Note that for many process mining purposes, e.g., process discovery and conformance checking, the case and event attributes are not mandatory, and it is sufficient to have a sequence of events for each case. We usually call this basic information control-flow information. In this regard, we show the trace that represents case 1 as $\langle a, b, c, d, e, f, h\rangle$ (using short-hand activity names), and for case 2 the trace is shown as $\langle a, b, g\rangle$. In the context of this paper, we formally define event logs as a multiset of sequences of activities. This assumption leads to ignoring the execution order of different process instances, as it is not important for the process discovery purpose.

**Definition 1** (*Event Log*) Let $\mathcal{A}$ be the universe of activities, and let $A \subseteq \mathcal{A}$ be a non-empty set of activities. An event log is a multiset of sequences over $A$, i.e. $L \in \mathbb{B}(A^*)$.

Observe that each $\sigma \in \overline{L}$ describes a *trace-variant* whereas $L(\sigma)$ describes how many traces of the form $\sigma$ are presented within the event log.

By sampling an event log, we choose some of the process instances of it. Sampling could be done with/without replacement. If an object is selected once, it is not selectable anymore in the sampling methods without replacement. Here, we use sampling without replacement. In other words, it is not possible to put an object more than once in the sampled event log. In the following, we formally define sampled event logs.

**Definition 2** (*Sampled event log*) Let $L \subseteq \mathbb{B}(A^*)$ be an event log. We define $S_L$ as a trace-based sampled event log of $L$, if $\forall \sigma \in S_L$ $(0 < S_L(\sigma) \leq L(\sigma))$. $S_L$ is a variant-based sampled event log of $L$ if for $\forall \sigma \in S_L$ $(1 = S_L(\sigma) \leq L(\sigma))$.

In other words, a variant-based sampled event log is a subset of trace-variants in $L$. Note that it is not possible to have a variant in a sampled event log that does not exist in the original event log.

We could define different types of behavior in an event log. One behavior in an event log is the directly follows relation between activities that can be defined as follows.

**Definition 3** (*Directly follows relation*) Let $a, b \in A$ be two activities and $\sigma = \langle \sigma_1, .., \sigma_n\rangle \in A^*$ is a trace in the event log. A directly follows relation from $a$ to $b$ exists in trace $\sigma$, if there is $i \in \{1, .., n-1\}$ such that $\sigma_i = a$ and $\sigma_{i+1} = b$ and we denote it by $a >_\sigma b$.

For example, in sequence $\langle a, b, c\rangle$, we have a directly follows relation from $b$ to $c$.

An alternative behavior which has negative affects on the results of process discovery algorithms is the occurrence of a low probable sub-pattern, i.e., a sequence of activities, between pairs of frequent surrounding behavior, which we refer to it as behavioral contexts [30].

**Definition 4** (*Behavioral context*) Let $L \in \mathbb{B}(\mathcal{A}^*)$ be an event log. A behavioral context $c$ is a pair of non-empty sequences of activities, i.e., $c \in \mathcal{A}^* \times \mathcal{A}^*$. Furthermore, we define the set of behavioral contexts present in $L$, i.e., $\beta_L \in \mathcal{P}(\mathcal{A}^* \times \mathcal{A}^*)$, as:

$$\beta_L = \left\{ (\sigma_l, \sigma_r) \in \mathcal{A}^* \times \mathcal{A}^* : \exists_{\sigma \in L, \sigma' \in \mathcal{A}^* \setminus \epsilon} \left( \sigma_l \cdot \sigma' \cdot \sigma_r \in \sigma \right) \right\} \tag{1}$$

For example, in trace $\sigma = \langle a, b, c, d, e, f, h \rangle$, $\langle a, b \rangle$ and $\langle e \rangle$ are two subsequences that surround $\langle c, d \rangle$; hence, the pair $(\langle a, b \rangle, \langle e \rangle)$ is a behavioral context. Note that the surrounded behavior could not be an empty sequence or $\epsilon$. As we are more interested in the contexts that frequently occur throughout the event log.

We inspect the probability of contextual sub-patterns, i.e., the behavior that is surrounded by the frequent behavioral contexts. Thus, we simply compute the empirical conditional probability of a behavioral sequence, being surrounded by a certain context.

**Definition 5** (*Conditional contextual probability*) Let $\sigma_s, \sigma_l, \sigma_r \in \mathcal{A}^*$ be three sequences of activities and let $L \in \mathbb{B}(\mathcal{A}^*)$ be an event log. We define the *conditional contextual probability* of $\sigma_s$, w.r.t., $\sigma_l$ and $\sigma_r$ in $L$, i.e., representing the sample based estimate of the conditional probability of $\sigma_s$ being surrounded by $\sigma_l$ and $\sigma_r$ in $L$. Function $\gamma_L : \mathcal{A}^* \times \mathcal{A}^* \times \mathcal{A}^* \rightarrow [0, 1]$, is based on:

$$\gamma_L(\sigma_s, \sigma_l, \sigma_r) = \frac{\sum_{\sigma \in L} \left( |\sigma_{\sigma_l \cdot \sigma_s \cdot \sigma_r}| \right)}{\sum_{\sigma \in L} \left( \sum_{\sigma' \in \mathcal{A}^* \setminus \epsilon} |\sigma'_{\sigma_l \cdot \sigma' \cdot \sigma_r}| \right)} \tag{2}$$

We alternatively write $P_L(\sigma_s | \sigma_l, \sigma_r)$ to represent $\gamma_L(\sigma_s, \sigma_l, \sigma_r)$.

Based on these probabilities, we are able to detect unstructured behavior in a trace.

## 4 Sampling event data

In this section, we present different sampling strategies to increase the discovery procedure's performance. We propose to sample different behavioral elements of an event log, e.g., events, directly follow relations, traces, and variants. By sampling events, we can choose events from different parts of a process instance that may not show the correct behavior of that process and leads to the inapplicability of it for the process discovery purpose. Sampling directly follows relations is useful for some process discovery algorithms like Alpha Miner and a version of inductive Miner. But, we need to modify these algorithms to accept a set of directly follows relations instead of an event log as an input. Also, such data structures do not apply to all process discovery algorithms. Thus, here we only consider trace and variant-based sampling. Consequently, these sampling methods take an event log as input and return a subset of traces or variants. The schematic of the sampling methods is illustrated in Fig. 1. Note that in some standard of storing event logs, e.g., XES [5], we do not have event logs in a multiset view. Therefore, we need to traverse the event log to find out variants and their frequency. Afterward, in variant-based sampling, we choose one process-instance for each of the selected variants, and consequently, the frequency of each sample is 1. In trace-based sampling, the frequency of each unique sample is $1 \leq n_i \leq m_i$.

For many process discovery algorithms such as ILP Miner, the family of Alpha miners and Inductive Miner, it is enough to have unique variants to discover a corresponding process model. In other words, the frequency of variants is mostly just used for post-processing algorithms like filtering. Therefore, here we mainly focus on

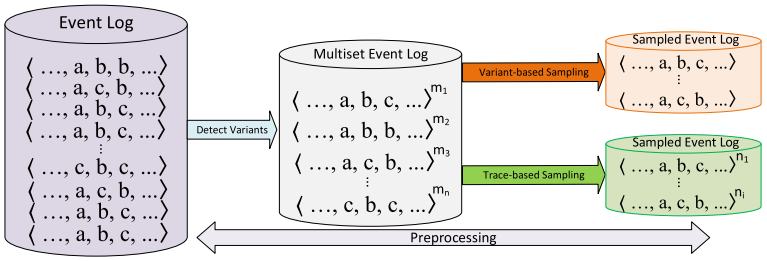


**Fig. 1** Schematic overview of the sampling methods. We first detect variants and afterward sample variants or traces based on different criteria

variant-based sampling, but, all these methods easily can be extended to trace-based sampling methods. Moreover, we also just used control-flow related information that is available in all event logs, and this is consistent with the way.

We are able to consider three dimensions for sampling event logs. The first one is the number of process instances that are placed in the sampled event log, i.e., $|S_L|$. In the worst case, it is the same as the original event log, i.e., we do not have any reduction in size. We can set the size of the sampled event logs (i.e., the sample ratio) as follows.

$$c = \begin{cases} \frac{|S_L|}{|L|} & \textit{Variant-based sampling} \\ \frac{|S_L|}{|L|} & \textit{Trace-based sampling} \end{cases} \tag{3}$$

Note that, in the above equation, $0 < c \leq 1$. The second dimension is the completeness of the sampled event log. If a sample event log contains few relations of the original event log, process discovery algorithms are not able to discover an appropriate process model from the sampled event log. However, including all the behavior in the original event log in the sampled event log is also leads to complex and imprecise process models. Note that there is a difference between the size of event data and behavior that it contains. Therefore, we should put the most important behavior in the event log. The last dimension is the required sampling time as a preprocessing phase. Some preprocessing techniques require too much time to return the preprocessed event log especially when we deal with large event logs. Consequently, sampling an event log in a shorter time is an advantage point of it.

We are able to sample behavioral elements in an event log randomly or by some sampling biases. In the following, we will explain both of these methods.

### 4.1 Random sampling

The first method is to randomly sample $c \times |L|$ traces in the event log without replacement and return these traces (i.e., trace-based sampling) or just unique trace-variants among them (i.e., variant-based sampling). This method is fast because we do not need to traverse the original event log. However, it is possible that many of the sampled traces have similar behavior, and we just return a few unique variants. Moreover, we may return variants that do not generalize the whole process.

Another approach first finds all the unique variants in an event log, after that, randomly select $c \times |\overline{L}|$ variants from them. This approach is a bit slower, but it is able to return more behaviors compared to the previous approach.

### 4.2 Biased sampling strategies

In general, traversing event logs have a linear complexity considering the number of process instances in the event log. It gives us a motivation that instead of randomly sampling the variants, we are able to use more advanced strategies (biases) to sample variants in the event log.

In biased sampling methods, we first traverse the event log to find unique trace-variants in it; then, we rank the variants based on different strategies. The top $N \times |\overline{L}|$ variants with the highest rank will be selected to be placed in the sampled event log. Different ranking strategies can be used for this purpose that is discussed in follows.

#### 4.2.1 Frequency-based sampling

The first ranking strategy is sampling variants based on their frequencies. This sampling method gives more priority to a variant that has a higher occurrence frequency in the event log. Hence, we sort the variants based on their frequencies or $L(\sigma)$ and return the top $c \times |\overline{L}|$ of variants as a sampled event log.

This strategy was proposed beforehand to simplify the discovered process models in some process mining tools. The advantage of this strategy is that we could grantee a minimum replay fitness of the future process model that will be discovered based on the sampled event log. Note that, in the random sampling strategy, the probability of choosing a more frequent variant is also higher. However, in some event logs, the majority of process instances have a unique trace-variant or variants with very low frequencies. So, to differentiate between them will be challenging, and using this strategy is not efficient anymore. Therefore, a drawback of this strategy is that the sampled event log may not contain many behaviors from the original event log.

#### 4.2.2 Length-based sampling

We can rank variants based on their length. In other words, we give a higher score to a shorter variant or to a longer one. If we want to keep more behaviors in our sampled event log, we need to choose traces with longer variants first. However, if we are interested in retaining the main-stream behaviors of the event log, usually it is better to choose traces with shorter variants. Thus, in this strategy, we sort variants based on their length, i.e., $|\sigma|$, and choose the longest or the shortest ones first.

Using the *longer* strategy, we are able to leave out incomplete traces, that improves the quality of resulted process models. However, if there are self-loops and other longer loops in the event log, there is a high probability to consider many infrequent variants with the same behavior for process discovery algorithms. For example, if we use directly follows information, it does not matter if trace $\sigma$ has $a >_\sigma a$ one time or more.

On the other hand, we may keep variants with more simple behavior if we use the shorter strategy; however, some incomplete process instances may be selected. Note that incomplete traces leads to having imprecise process models.

#### 4.2.3 Similarity-based sampling

If we aim to sample variants that contain general behavior of the whole event log, we need to use the similarity-based sampling methods. Using this approach, we first find the general behaviors of the event log. We are able to use different behavior; however, the simplest and the most critical behavior for process discovery is the directly follows relation. Therefore, we compute the occurrence probability of each directly follows relation $B_i{=}(a_1, a_2)$ (that $a_1, a_2{\in}A$) according to the following formula.

$$Prob(B_i){=}\frac{num|\sigma \in \overline{L}|a_1 >_\sigma a_2|}{|\overline{L}|} \tag{4}$$

Hence, we compute the probability of observing each directly follows relation $B_i$ in a variant. If $Prob(B_i)$ is high (i.e., be higher than a defined threshold $T_P$), we expect that the sampled variants should contain it. Thus, any variant contains such a high probable behavior (that here is a directly follows relations), will its ranking (by $+1$). Otherwise, if a variant does not contain a probable behavior, we decrease its ranking by giving a negative value (i.e., $-1$). Contrariwise, if a variant contains a low probable behavior (i.e., $Prob_{B_i} {\leq} 1{-}T_P$), we decrease its ranking by 1. Thus, we are searching for variants that have very high probable behaviors and have less low probable behaviors. Note that, it is possible that some behaviors be neither high probable nor low probable that we do nothing for such behaviors. Note that we normalize the rankings based on the length of variants. Afterward, we sort the variants based on their rankings and return the $c{\times}|\overline{L}|$ ones with the highest ranking.

The main advantage of this method is that it helps process discovery algorithms to depict the main-stream behavior of the original event log in the process model. However, it needs more time to compute a similarity score of all variants. Especially, if we use more advanced behavioral structures such as eventually follow relations, this computation will be a limitation for this ranking strategy.

#### 4.2.4 Structured-based sampling

It is shown in [18] that unstructured behavior in event logs leads to imprecise and complex process models. In this sampling strategy, we consider the presence of unstructured behavior (i.e., based on Definition 5) in each variant. In this regard, we first compute the occurrence probability of each sub-patten among its specific contextual context (i.e., $P_L(\sigma_s, \sigma_l, \sigma_r)$). If this probability is below a given threshold, i.e., $T_S$, we call it an odd structure or unstructured behavior. Thus, for each unstructured behavior in a variant, we define a penalty to it and decrease its ranking by $-1$. Consequently, a variant with higher odd structures receives more penalties, and it is not appealing to be placed in the sampled event log. Note that in this strategy, we do not normalize the negative values based on the length of the variant.

**Table 1** Details of real event logs that are used in the experiment

| Event log | Activities# | Traces# | Variants# | DF relations# |
|---|---|---|---|---|
| BPIC_2012 [31] | 23 | 13,087 | 4336 | 138 |
| BPIC_2013 [32] | 4 | 7554 | 1511 | 11 |
| BPIC_2017_All [33] | 26 | 31,509 | 1593 | 178 |
| BPIC_2017_Offer [33] | 8 | 42,995 | 169 | 14 |
| BPIC_2018_Control [34] | 7 | 43,808 | 59 | 12 |
| BPIC_2018_Inspection [34] | 15 | 5485 | 3190 | 67 |
| BPIC_2018_Reference [34] | 6 | 43,802 | 515 | 15 |
| *Hospital* [35] | 18 | 100,000 | 1020 | 143 |
| *Road* [36] | 11 | 150,370 | 231 | 70 |
| *Sepsis* [37] | 16 | 1050 | 846 | 115 |

#### 4.2.5 Hybrid sampling

In hybrid strategies, we combine two or three of other sampling strategies. In this way, we expect to have the benefits of different methods. To do so, we normalize various ranking strategies to values between 0 and 1. Then, we use a weighting average method to aggregate normalized values. Here, we combine the frequency and similarity-based methods; however, other combinations are also possible.

In the next section, we show the influence of the sampling strategies on the performance of process discovery procedure and quality of their results.

## 5 Evaluation

We conduct some experiments to answer the following research questions:

- (**Q1**) Does sampling event logs improve the performance of different process discovery algorithms?
- (**Q2**) Does variant-based sampling outperform trace-based sampling?
- (**Q3**) Is the quality of process models that are discovered using sampled event logs similar to process models that are discovered from the original event logs?
- (**Q4**) Which sampling strategies are faster and result in high-quality process models?
- (**Q5**) How does the sampling threshold affect the sampling and discovery time?

### 5.1 Implementation

To apply the proposed sampling strategies, we implemented the *Sample Variant* plug-in in the `ProM` framework[1] [5]. In this implementation, we used static thresholds

[1] Sample Variant plug-in in the LogFiltering package: https://svn.win.tue.nl/repos/prom/Packages/LogFiltering.

for similarity and structured based sampling strategies. The user is able to specify the desired percentage of the sampling traces/variants and the ranking strategy. The plug-in takes an event log as an input and produces an event log contains top $c \times 100$ percentage of traces/variants as an output. In addition, to apply our proposed method on various event logs with different parameters, we ported the *Sample Variant* plug-in to `RapidProM` [6] that is an extension of `RapidMiner` that combines scientific work-flows with a range of (`ProM`-based) process mining algorithms.

### 5.2 Experimental setup

Information about ten real event logs that are used in the evaluation is given in Table 1. These event logs are accessible at https://data.4tu.nl/search?q=process%20mining&contentTypes=collection. For process discovery, we used a family of Alpha Miner [8] (i.e., the basic Alpha Miner, Alpha++ and Alpha#), Inductive Miner [12], ILP Miner [38], and Split Miner [11]. In cases whereas the event logs were sampled, we applied process discovery algorithms just without their built-in filtering mechanisms.

We sampled event logs with different variant and trace-based sampling strategies, and $c$ in [0.05, 0.10, 0.15, 0.20]. Each experiment was repeated five times and the average values are shown in these figures. The y-axis represents the average performance-improvements using a logarithmic scale.

### 5.3 Experimental result

Here, we show how experimental results address the mentioned research questions.

#### 5.3.1 (Q1 and Q2)

To measure the performance improvement, we consider both the discovery time and sampling time of event logs using the following formulas. Higher values for these measures shows the number of times we are faster using the sampling methods.

$$DiscoveryTime_{Improvement} = \frac{DiscoveryTime_{WholeLog}}{DiscoveryTime_{SampledLog}} \tag{5}$$

$$TotalTime_{Improvement} = \frac{DiscoveryTime_{WholeLog}}{DiscoveryTime_{SampledLog} + SamplingTime} \tag{6}$$

Figures 2 and 3 show the discovery time and total improvement when we sample event logs with/without considering sampling time corresponding to Eqs. 5 and 6. In these figures, a higher value shows a higher improvement in the performance of process discovery procedure. It is evident that by reducing the size of the event log, the process discovery time is reduced. Therefore, the $DiscoveryTime_{Improvement}$ for variant-based sampling is significantly higher than trace-based sampling. Since the $|S_L|$ for variant-based sampling is usually remarkably lower than trace-based one. For some

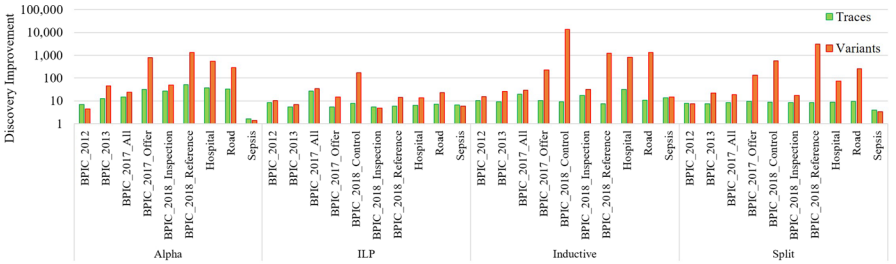


**Fig. 2** Process discovery performance improvement for different process discovery algorithms using variant and trace-based sampling methods

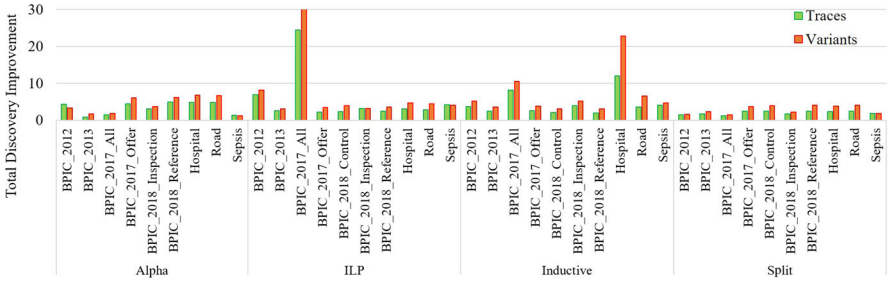


**Fig. 3** Total time improvement for discovering process using sampling methods

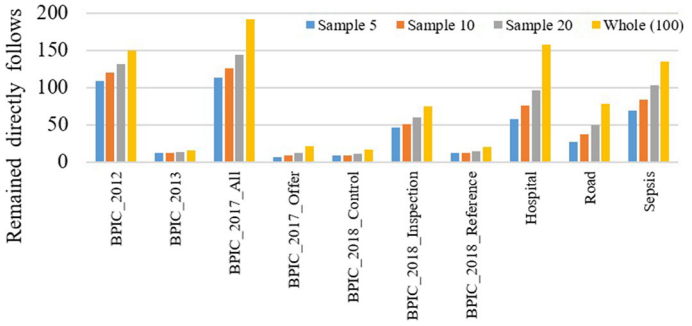


**Fig. 4** Reduction in the number of directly follows relations by sampling event logs

event logs, process discovery is more than 10,000 times faster on the sampled event log using variant-based sampling compared to using the whole event log. However, for event logs such as *Sepsis*, where most of the traces have unique control-follow related behavior in the original event log, trace-based sampling methods are faster.

Furthermore, by considering the sampling phase as a preprocessing step, we are able to reduce the required time to discover a process model for most of the event logs. Variant-based methods need more time to perform the sampling because they need to discover variants among traces. However, they usually have a higher improvement the total required time for discovering process (i.e., $SamplingTime + DiscoveryTime_{SampledLog}$).


**Fig. 5** The number of remained traces in the sampled event logs when we used variant and trace-based sampling with c equals to 5 and 20. For event logs that have frequent variants the trace reduction is more

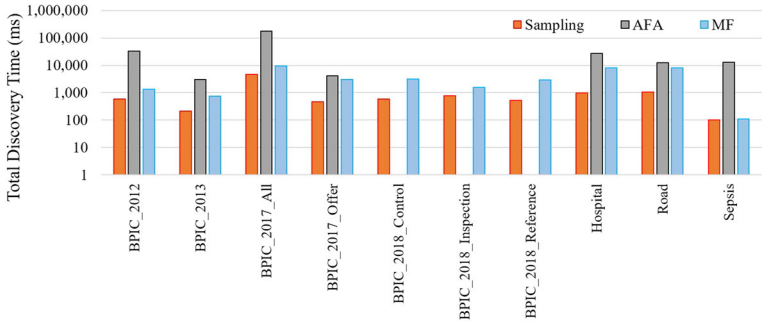


**Fig. 6** Comparing the average of total discovery time of using variant sampling methods and event log filtering methods

Sampling methods improve the performance of process discovery by first reducing the number of traces and also decreasing the behaviors in the event log. The required time for process discovery algorithms depends on different factors. In ILP Miner, this time is mainly related to the number of activities and the number of unique variants in the event log that sampling can reduce both of them. However, in the family of Alpha Miner and Split Miner, we firstly discover possible directly follow relations and then create a process model from these relations. For these algorithms, we increase the performance of process discovery by reducing the number of possible directly follow relations. In Inductive Miner, we iteratively divide the process to block structured sub-processes. Hence, in each step, we aim to divide the event log and find the corresponding directly follow relation of the sub-event log. Sampling reduces the number of possible directly follow relations and the number of process instances in the event log which leads to the performance improvement of process discovery using Inductive Miner.

Figure 4 shows how many behaviors (here the directly follows) are reduced in the sampled event logs. Figure 5 indicates the average number of traces in the sampled event log using trace/event sampling. For most of the event logs, if we use variant-based sampling, the size of the event logs (i.e., $|L|$) is reduced significantly. Also, the number of remaining variants in sampled event logs is reduced to 5–20% of the number of variants in the original event log (i.e., $|\overline{L}|$).

In Fig. 6, we compared variant-based sampling methods and event log filtering methods on their ability to improve process discovery performance. In this experiment, two automated event log filtering methods are considered to be Anomaly-Free Automaton (AFA) [16] and Matrix Filter (MF) [10]. For some even logs, because of

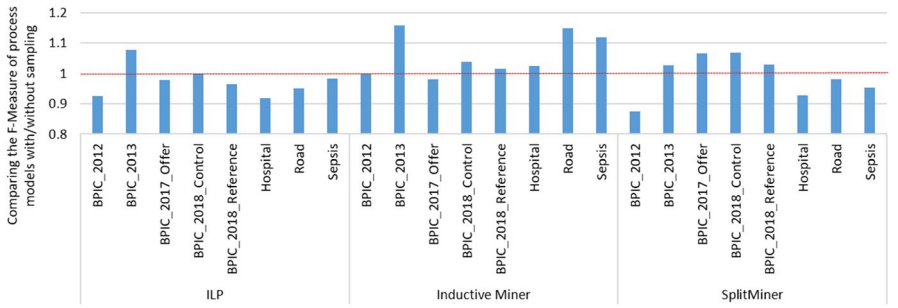


**Fig. 7** Analyzing the F-measure similarity of discovered process models with/without sampling using Eq. 8. for the cases with values higher than 1, the F-measure of discovered process models are higher when we used the sampled event logs

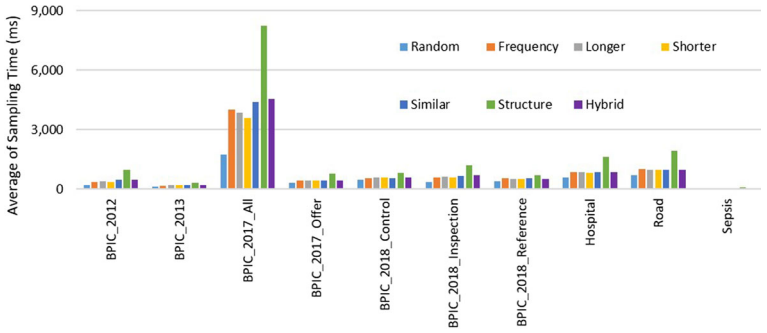


**Fig. 8** The average sampling time for different sampling strategies. The random method is the fastest and the structure method is the slowest sampling method

some technical problems, we could not filter the event log using the AFA method. Results show that variant-based sampling reduces the total discovery time.

### 5.3.2 (Q3)

Here, we aim to analyze the quality of discovered process models from sampled event logs. For this purpose, we use *fitness* and *precision*. Fitness measures what percentage of event log's behaviors are also replayable by the process model. Thus, a fitness value equals to 1, indicates all the behaviors in the event log, are described by the process model. Precision measures to what extend behaviors that are described by the process model are also presented in the event log. A low precision value means that the process model allows for more behaviors compared to the event log. There is a trade-off between these measures [39], sometimes, putting aside a small amount of behavior causes a slight decrease in the fitness value, whereas the precision value increases dramatically. Thus, we use the F-Measures metric that combines both of them.

$$\text{F-Measure} = 2 \times \frac{\text{Precision} \times \text{Fitness}}{\text{Precision} + \text{Fitness}} \tag{7}$$

In Fig. 7, we compared the quality of best process models that are discovered with/without sampling according to the following formula.

$$\text{F-Measure}_{Similarity} = \frac{\text{F-Measure}_{SampledLog}}{\text{F-Measure}_{WholeLog}} \tag{8}$$

We used sampled event logs just for discovery purpose, and the original event logs were used for computing the F-Measure. For process models that are discovered without sampling methods, we iterate the experiment with 100 different embedded filtering parameter(s) and considering their best F-Measure (just for Inductive Miner and Split Miner). We do not consider Alpha miner algorithms, because they usually result in unsound process models that we are not able to find their corresponding F-Measure.

Note that, for some event logs (e.g., *BPI*-2017-All), we could not compute the F-Measure of process models. An $F\text{-}Measure_{Similarity} > 1$ means that sampling methods increase the quality of the corresponding discovered process model (even compared to the cases that we used process discovery algorithms with their filtering mechanism). Note that, we did not use filtering mechanisms of the process discovery algorithm for sampled event logs. For some event logs, sampling methods can increase the quality of discovered process models, specifically, if we use Inductive Miner. It shows the weakness of process discovery algorithms in dealing with infrequent behaviors [10].

By applying variant-based sampling methods, we will lose the frequency of variants. As a result, some embedded filtering mechanisms in process discovery algorithms become unusable. However, the results of this experiment show that we can discover process models with high quality from sampled event logs, even without these filtering methods. If for any reason we need to use frequency of variants, it is recommended to apply trace-based sampling methods.

#### 5.3.3 (Q4)

Here, we want to compare different variant biased sampling strategies. In Fig. 8 the average of the sampling time (in milliseconds) for different variant-based sampling strategies is shown. The random sampling is the fastest strategy as it does not need to rank variants, and the structure-based sampling is the slowest one. After the structure-based sampling, the similarity-based sampling and the hybrid sampling that use directly follows relations are slower than other strategies. Table 2 compares the quality of the process models that are discovered using these sampling methods. For some combinations, we could not compute the F-Measure in the specific time, and their corresponding cell in the table is empty. Results show that no unique sampling method results in process models with the highest possible F-Measure for all process discovery algorithms.

#### 5.3.4 (Q5)

Finally, we analyzed the impact of the sampling threshold (i.e., $c$) on the sampling and process discovery time. Thus, we sampled *Road* event log with $c$ in [0.05, 0.1, 0.15,

**Table 2** The best F-measure of discovered process models using different process discovery algorithms and different variant-based sampling strategies

| Discovery | Event Log | Frequency | Hybrid | Longer | Random | Shorter | Similar | Structure |
|---|---|---|---|---|---|---|---|---|
| ILP | BPIC_2012 | | **0.77** | | **0.77** | 0.44 | | 0.69 |
| | BPIC_2017_Offer | 0.70 | **0.99** | 0.98 | 0.93 | 0.93 | **0.99** | **0.99** |
| | BPIC_2018_Control | 0.99 | **1.00** | 0.99 | 0.99 | **1.00** | **1.00** | **1.00** |
| | BPIC_2018_Reference | 0.92 | **0.99** | 0.92 | 0.96 | **0.99** | 0.96 | 0.96 |
| | Hospital | 0.59 | **0.90** | 0.60 | 0.81 | 0.85 | **0.90** | **0.90** |
| | Road | 0.70 | 0.88 | 0.87 | **0.92** | 0.91 | 0.91 | 0.91 |
| | Sepsis | 0.44 | 0.80 | 0.43 | **0.81** | 0.51 | 0.46 | 0.46 |
| Inductive | BPIC_2012 | 0.73 | **0.77** | 0.75 | **0.77** | 0.73 | 0.66 | 0.66 |
| | BPIC_2013 | 0.87 | 0.93 | 0.95 | 0.93 | **0.96** | 0.95 | 0.95 |
| | BPIC_2017_Offer | 0.98 | **0.99** | 0.98 | 0.94 | 0.94 | 0.93 | 0.93 |
| | BPIC_2018_Control | **1.00** | **1.00** | 0.97 | **1.00** | 0.97 | **1.00** | **1.00** |
| | BPIC_2018_Reference | 0.96 | **1.00** | 0.93 | 0.96 | 0.96 | 0.99 | 0.99 |
| | Hospital | | **0.90** | | 0.81 | **0.90** | **0.90** | **0.90** |
| | Road | 0.92 | 0.93 | 0.92 | 0.92 | 0.93 | **0.94** | **0.94** |
| | Sepsis | 0.73 | 0.81 | 0.71 | 0.76 | **0.82** | 0.75 | 0.75 |
| Split | BPIC_2012 | 0.81 | 0.81 | 0.79 | **0.85** | 0.81 | 0.81 | 0.81 |
| | BPIC_2013 | **0.96** | **0.96** | **0.96** | **0.96** | **0.96** | **0.96** | **0.96** |
| | BPIC_2017_Offer | **1.00** | 0.99 | 0.98 | 0.93 | 0.93 | 0.96 | 0.96 |
| | BPIC_2018_Control | **1.00** | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| | BPIC_2018_Reference | | **0.96** | | **0.96** | | **0.96** | **0.96** |
| | Hospital | | | | **0.81** | | | |
| | Road | | 0.88 | | **0.93** | | 0.88 | 0.88 |
| | Sepsis | | **0.75** | | **0.75** | | | |

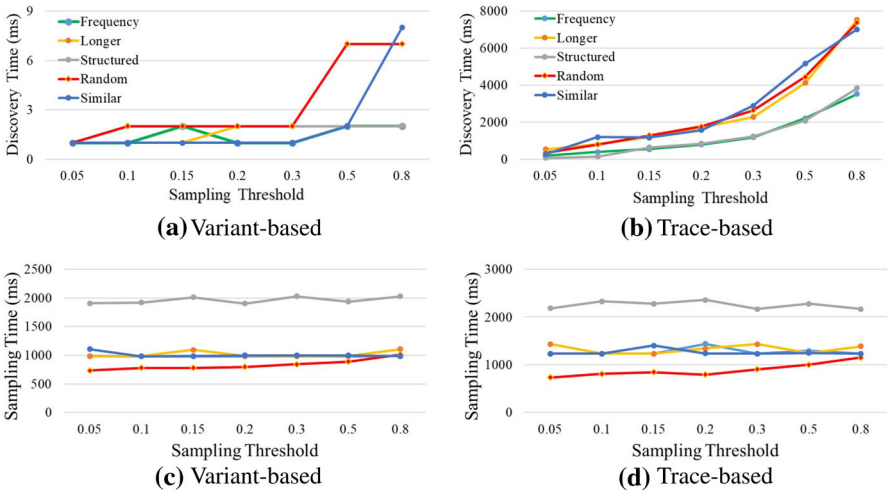


**Fig. 9** The median of the sampling and discovery time for sampling *Road* event log using different sampling thresholds

0.2, 0.3, 0.5, 0.8] and using the basic inductive miner for process discovery. This experiment was repeated 10 times, and the median values are shown in Fig. 9. We have not considered the hybrid method as its results are similar to frequency based for this event log. In variant-based sampling, except for the *random* sampling, the sampling time is independent of the sampling threshold. Because in biased variant-based sampling methods, most of the sampling time goes for ranking the variants. It is also a similar case for trace-based samplings. Structured-based sampling bias is the slowest among the other biases. In variant-based sampling, the discovery time is dramatically faster than traced-based sampling. Among trace-based sampling methods, structured and frequency-based strategies are faster than others.

The experimental results show that using biased sampling methods, we are able to improve the performance of process discovery procedure and it is possible to have this improvement when different process discovery algorithms are applied. As in most of the cases, there will be fewer traces in the sampled event logs using variant-based sampling methods, the improvement in the performance of process discovery is higher using the variant-based approaches. Moreover, results show that the process models that are discovered using sampled event logs have similar quality compared to the case that the original event logs are used (considering F-Measure metric). There are some cases that the sampling approach improves the quality of discovered models. In addition, considering different sampling strategies, we could not find a strategy that always results in the highest process model quality. However, the hybrid approach results in process models with high quality on the used event logs. According to the sampling time, the random strategy outperforms other ones and the structure strategy has the lowest performance. Finally, results indicate that the sampling threshold mainly affects the discovery time when the trace-based approaches are used. Also, results show that sampling time is independent of the sampling threshold.

## 6 Conclusion

In this paper, we proposed several variant and trace-based sampling strategies to increase the performance of the process discovery procedure. We recommend applying process discovery algorithms on the sampled event logs, especially when dealing with large or complex event logs. We implemented different sampling strategies in ProM and RapidProM. Then, we applied them on many real event logs using different process discovery algorithms. Experimental results showed that sampling an event log decreases the required time used by state-of-the-art process discovery algorithms. We found out that variant-based sampling approach results in considerably higher process discovery performance improvement compared to the trace-based approach. Results showed that by applying sampling methods, we are able to discover acceptable approximations of final process models in a short time. Moreover, results indicated that, for some event logs, sampling methods can also improve the quality of discovered process models according to the F-Measure metric.

As future work, we aim to define more computationally affordable ranking strategies. Furthermore, we are interested in finding out the best sampling strategy considering the process discovery quality and performance when dealing with different event logs and process discovery algorithms.

**Acknowledgements** We thank the Alexander von Humboldt (AvH) stiftung for supporting this research.

**Funding** Open Access funding enabled and organized by Projekt DEAL.



## References

1. van der Aalst WMP (2016) Process mining—data science in action, 2nd edn. Springer, Berlin
2. Pourbafrani M, van Zelst SJ, van der Aalst WMP (2020) Supporting automatic system dynamics model generation for simulation in the context of process mining. In: Abramowicz W, Klein G (eds) Business information systems. Springer, Cham, pp 249–263
3. Park G, Song M (2020) Predicting performances in business processes using deep neural networks. Decis Support Syst 129:113191
4. van der Aalst WMP et al (2011) Process mining manifesto. In: Business process management BPM workshops, Clermont-Ferrand, France, pp 169–194
5. Verbeek HMW, Buijs JCAM, van Dongen BF, van der Aalst WMP (2010) Xes, xesame, and prom 6. In Soffer P, Proper E (eds) Information systems evolution-CAiSE Forum 2010, Hammamet, Tunisia, June 7–9, 2010, Selected Extended Papers. Volume 72 of lecture notes in business information processing. Springer, pp 60–75
6. van der Aalst WMP, Bolt A, van Zelst S (2017) RapidProM: mine your processes and not just your data. CoRR abs/1703.03740

7. Fani Sani M, van Zelst SJ, van der Aalst WMP (2019) The impact of event log subset selection on the performance of process discovery algorithms. In: New trends in databases and information systems, ADBIS 2019 short papers, workshops BBIGAP, QAUCA, SemBDM, SIMPDA, M2P, MADEISD, and doctoral consortium, bled, Slovenia, September 8–11, 2019, proceedings. Volume 1064 of communications in computer and information science. Springer, pp 391–404
8. van der Aalst WM, Weijters T, Maruster L (2004) Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142
9. Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs—a constructive approach. In Colom JM, Desel J (eds) Application and theory of petri nets and concurrency—34th international conference, PETRI NETS 2013, Milan, Italy, June 24–28, 2013, proceedings. Volume 7927 of lecture notes in computer science. Springer, pp 311–329
10. Fani Sani M, van Zelst SJ, van der Aalst WMP (2017) Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Business process management workshops—BPM 2017, Barcelona, Spain, September 10–11, 2017, revised papers. Volume 308 of lecture notes in business information processing. Springer, pp 216–229
11. Augusto A, Conforti R, Dumas M, Rosa ML, Polyvyanyy A (2019) Split miner: automated discovery of accurate and simple business process models from event logs. Knowl Inf Syst 59(2):251–284
12. Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs containing infrequent behaviour. In Lohmann N, Song M, Wohed P (eds) Business process management workshops-BPM 2013 international workshops, Beijing, China, August 26, 2013, Revised papers. Volume 171 of lecture notes in business information processing. Springer, pp 66–78
13. van Zelst SJ, van Dongen BF, van der Aalst WMP (2015) Avoiding over-fitting in ilp-based process discovery. In Motahari-Nezhad HR, Recker J, Weidlich M (eds) Business process management—13th international conference, BPM 2015, Innsbruck, Austria, August 31–September 3, 2015, proceedings. Volume 9253 of lecture notes in computer science. Springer, pp 163–171
14. Pegoraro M, Uysal MS, van der Aalst WMP (2019) Discovering process models from uncertain event data. In: Business process management workshops-BPM 2019 international workshops, Vienna, Austria, September 1–6, 2019, Revised selected, pp 238–249
15. Horita H, Kurihashi Y, Miyamori N (2020) Extraction of missing tendency using decision tree learning in business process event log. Data 5(3):82
16. Conforti R, Rosa ML, ter Hofstede AHM (2017) Filtering out infrequent behavior from business process event logs. IEEE Trans Knowl Data Eng 29(2):300–314
17. Fani Sani M, van Zelst SJ, van der Aalst WMP (2018) Applying sequence mining for outlier detection in process mining. In: On the move to meaningful internet systems. OTM 2018 conferences-confederated international conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22–26, 2018, Proceedings, Part II. Volume 11230 of lecture notes in computer science. Springer, pp 98–116
18. Fani Sani M, van Zelst SJ, van der Aalst WMP (2018) Repairing outlier behaviour in event logs using contextual behaviour. Inf Syst Arch 14:5:1-5:24
19. Conforti R, La Rosa M, Ter Hofstede AH, Augusto A (2020) Automatic repair of same-timestamp errors in business process event logs. In: International conference on process mining, ICPM 2020, Padua, Italy, October 4–9, 2020. IEEE, pp 327–345
20. Sadeghianasl S, ter Hofstede AH, Suriadi S, Turkay S (2020) Collaborative and interactive detection and repair of activity labels in process event logs. In: International conference on process mining, ICPM 2020, Padua, Italy, October 4–9, 2020. IEEE, pp 41–48
21. Tax N, Sidorova N, van der Aalst WMP (2019) Discovering more precise process models from event logs by filtering out chaotic activities. J Intell Inf Syst 52(1):107–139
22. Dees M, Hompes B, van der Aalst WM (2020) Events put into context (epic). In: International conference on process mining, ICPM 2020, Padua, Italy, October 4–9, 2020. IEEE, pp 65–72
23. Fani Sani M, Berti A, van Zelst SJ, van der Aalst WMP (2019) Filtering toolkit: Interactively filter event logs to improve the quality of discovered models. In: Proceedings of the dissertation award, doctoral consortium, and demonstration track at on business process management BPM 2019, Vienna, Austria, September 1–6, 2019. Volume 2420 of CEUR workshop proceedings. CEUR-WS.org, pp 134–138
24. Fani Sani M, van Zelst SJ, van der Aalst WMP (2020) Conformance checking approximation using subset selection and edit distance. In: Advanced information systems engineering-32nd international conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, proceedings. Volume 12127 of lecture notes in computer science. Springer, pp 234–251

25. Rafiei M, van der Aalst WMP (2020) Privacy-preserving data publishing in process mining. In: Business process management forum-BPM forum 2020, Seville, Spain, September 13–18, 2020, proceedings. Volume 392 of lecture notes in business information processing. Springer, pp 122–138
26. Carmona J, Cortadella J (2010) Process mining meets abstract interpretation. In Balcázar JL, Bonchi F, Gionis A, Sebag M (eds) Machine learning and knowledge discovery in databases, European Cconference, ECML PKDD 2010, Barcelona, Spain, September 20–24, 2010, proceedings, Part I. Volume 6321 of lecture notes in computer science. Springer, pp 184–199
27. Bauer M, Senderovich A, Gal A, Grunske L, Weidlich M (2018) How much event data is enough? A statistical framework for process discovery. In Krogstie J, Reijers HA (eds) Advanced information systems engineering-30th international conference, CAiSE 2018, Tallinn, Estonia, June 11–15, 2018, proceedings. Volume 10816 of lecture notes in computer science. Springer, pp 239–256
28. Berti A (2017) Statistical sampling in process mining discovery. In: The 9th international conference on information, process, and knowledge management, pp 41–43
29. Weijters AJMM, Ribeiro JTS (2011) Flexible heuristics miner (FHM). In: Proceedings of the IEEE symposium on computational intelligence and data mining, CIDM 2011, April 11–15, 2011, Paris, France. IEEE, pp 310–317
30. Fani Sani M, van Zelst SJ, van der Aalst WMP (2018) Repairing outlier behaviour in event logs. In Abramowicz W, Paschke A (eds) Business information systems-21st international conference, BIS 2018, Berlin, Germany, July 18–20, 2018, proceedings. Volume 320 of lecture notes in business information processing. Springer, pp 115–131
31. van Dongen BF (2012) BPIC 2012. Eindhoven University of Technology
32. Ward Steeman: BPIC 2013. Eindhoven University of Technology (2013)
33. van Dongen BF (2017) BPIC 2017. Eindhoven University of Technology
34. van Dongen B, Borchert F (2018) BPIC 2018. Eindhoven University of Technology
35. Mannhardt F (2017) Hospital billing-event log. Eindhoven University of Technology. Dataset 326–347
36. De Leoni M, Mannhardt F (2015) Road traffic fine management process
37. Mannhardt F (2016) Sepsis cases-event log. Eindhoven University of Technology
38. van Zelst S, van Dongen B, van der Aalst WMP, Verbeek HMW (2017) Discovering workflow nets using integer linear programming. Computing
39. Weerdt JD, Backer MD, Vanthienen J, Baesens B (2011) A robust f-measure for evaluating discovered process models. In: Proceedings of the IEEE symposium on computational intelligence and data mining, CIDM 2011, part of the IEEE symposium series on computational intelligence 2011, April 11–15, 2011, Paris, France. IEEE, pp 148–155