Improving the Performance of Process Discovery Algorithms by Instance Selection*

Mohammadreza Fani Sani[1], Sebastiaan J. van Zelst[1,2], and Wil van der Aalst[1,2]

[1] Process and Data Science Chair, RWTH Aachen University, Aachen, Germany
{fanisani, s.j.v.zelst, wvdaalst}@pads.rwth-aachen.de
[2] Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany

Abstract. Process discovery algorithms automatically discover process models based on event data that is captured during the execution of business processes. These algorithms tend to use all of the event data to discover a process model. When dealing with large event logs, it is no longer feasible using standard hardware in limited time. A straightforward approach to overcome this problem is to down-size the event data by means of sampling. However, little research has been conducted on selecting the right sample, given the available time and characteristics of event data. This paper evaluates various subset selection methods and evaluates their performance on real event data. The proposed methods have been implemented in both the ProM and the RapidProM platforms. Our experiments show that it is possible to considerably speed up discovery using instance selection strategies. Furthermore, results show that applying biased selection of the process instances compared to random sampling will result in simpler process models with higher quality.

Keywords: Process Mining, Process Discovery, Subset Selection, Event Log Preprocessing, Performance Enhancement.

1. Introduction

Process Mining bridges the gap between traditional data mining and business process management analysis[1]. The main subfields of process mining are 1) *process discovery*, i.e., finding a descriptive model of the underlying process, 2) *conformance checking*, i.e, monitoring and inspecting whether the execution of the process in reality conforms to the corresponding designed (or discovered) reference process model, and 3) *enhancement*, i.e, the improvement of a process model, based on the related event data [1]. With process discovery, we aim to discover a process model that accurately describes the underlying process captured within the event data, also referred to as *event logs*, readily available in most modern information systems. In conformance checking, we aim to assess to what degree a given process model (possibly the result of a process discovery algorithm) and event data conform to one another. Finally, process enhancement aims at improving the view on a process by improving or enhancing the corresponding model using related event data, e.g., by projecting bottleneck information directly onto a (given) process model. There are also other dimensions like prediction[39, 38] and business process automation [24].

* This article is the extension of a conference paper entitled "The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms" that was initially published in 23rd European Conference on Advances in Databases and Information Systems 2019 Workshops proceedings (Springer CCIS 1064) [41].

Currently, the main research focus of process discovery is on quality issues of the discovered process models; however, at the same time, the ever-increasing size of the data handled in process mining leads to performance issues when applying the existing process discovery algorithms [2]. Most process discovery algorithms first build an internal abstraction of data, based on the whole event log and then apply a possible filtering step is applied. However, this attitude causes efficiency in real process mining projects with large event data. Some process discovery algorithms are infeasible in big data settings, where the event data are too large to process. Additionally, some process mining tools enforce 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 model is able to be discovered by applying just a small fraction of the event data.

In real life, process discovery is often of an exploratory nature which 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 improvement in performance may accumulate to a significant performance increase when applied several times. Furthermore, many process discovery algorithms are designed to also generalize the behavior that is observed in the event data. In other words, the result of process discovery algorithms contains more behavior compared to the inputted event log. Therefore, it may still be possible to discover the underlying process using a subset of event data. In addition, many of process discovery algorithms aim to depict as much as possible behavior in event logs. However, for real event logs, because of the presence of noisy and uncertain behavior, applying these algorithms resulted in inaccurate and incomprehensible, complex process models that are even accurate behavior undetectable in them [18, 36].

To deal with the complexity of process models, in some research, such as [42, 31, 51], clustering methods have been applied to get several sub-models according to clustered sub-logs. For example, [30] and [10] use data attributes and conformance artifacts for this purpose. Using this approach, the quality of the sub-models is better than a single process model discovered on the complete event log. However, getting several process models may be a barrier for decision-makers who need a single overview of each process.

This research studies the effectiveness of applying biased sampling on event data prior to 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. Using the `ProM`-based [48] extension of `RapidMiner` [3], 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 discovery algorithms. Moreover, for some event logs, by sampling, we also improve the quality of discovered process models due to implicit filtering.

This paper extends the work [41]. Here, we explain the instance selection strategies in more detail. The proposed approaches are also applied on many real event logs with state-of-the-art process discovery algorithms, i.e., the Alpha Miner, the Inductive Miner, the Split Miner, and the ILP Miner. We propose to select some process instances of an event

log based on variants or traces and apply process discovery algorithms on the sampled data. Moreover, in this paper, we used more metrics to evaluate the quality and complexity of discovered process models.

The remainder of this paper is structured as follows. In Section2, we discuss related work. Section 3 defines preliminary notation. We present different biased instance selection strategies in Section 4. The experimental evaluation and corresponding results are given in Section 5. Finally, Section 6 concludes the paper and presents some directions for future work.

2. Related Work

Many discovery algorithms, e.g., the Alpha Miner [5], the ILP Miner [53], and the basic Inductive Miner [28] were designed to first build an internal data structure, e.g., the directly follows graph, based on the whole available event data. Subsequently, these algorithms discover a process model directly from this data structure. Other process discovery algorithms, e.g., the Split Miner [7], the Flexible Heuristic Miner [52], the Fuzzy Miner [25], the extended versions of the Inductive Miner[29], and the ILP Miner [54] were designed to be able to filter infrequent or noisy behavior within their internal data structure, prior to discovering a process model. The performance of all these algorithms depends on the number of process instances and the unique number of activities. For some of them, e.g., ILP Miner[53], the number of unique process instances also influences the required time to discover a process model.

Recently, preprocessing of event data has gained attention. In [44, 6], techniques are proposed to increase the quality of discovered process models by cleansing the event data. Some other aim to increase/keep the privacy of users by preprocessing an event log beforehand [40]. Also, in [18], [19], [20] and [21] we have shown that by removing/modifying outlier behavior in event logs, process discovery algorithms are able to discover process models with higher quality. Furthermore, [55] proposes a filtering approach to detect and remove infrequent behavior for event streams. Moreover, [33] uses data attributes to filter out noisy behavior. In [17] an interactive filtering toolkit is provided that let user chooses different filtering methods in combination with several process discovery algorithms. Filtering techniques effectively reduce the size of the event data used by process discovery algorithms. But, to do so, these filtering techniques have non-linear time complexity that does not scale in the context of big data. Meanwhile, sometimes the required time for applying these filtering algorithms is longer than the process discovery time. Also, these techniques have no accurate control over the size of the sampled event log.

Filtering techniques focus on removing infrequent behavior from event data; however, sampling methods aim to reduce the number of process instances and increase the performance of other procedures. Some sampling approaches have been proposed in the process mining field. [26] studies some behavior qualities for sampled event logs. In [22], the authors show that by selecting a few unique process instances of event data, it is possible to approximate the conformance value in a shorter time. In [12], the authors proposed a sampling approach based on the Parikh vector of traces to detect the behavior in the event log. However, we are not able to use this sampling technique for the process discovery purpose, because, the Parikh vector does not store the sequences of activities that are critical for discovering process models. In [8], the authors recommend a random trace-based sam-

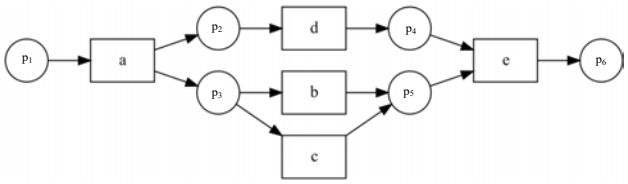

Fig. 1. An example process model with Petri net notation.

pling 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 lead to different types of process behavior. Furthermore, [9] recommends a trace-based sampling method specifically for the Heuristic miner[52]. Both these sampling methods are unable to control 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, all these mentioned sampling methods use random trace-based sampling with a replacement that may lead to pick and analyze a unique process instance several times. Finally, as these methods are unbiased, we have non-deterministic results after each sampling. In this paper, we will offer and analyze random and biased instance selection methods which the size of the sampled event data is adjustable.

3. Preliminaries

In this section, we briefly introduce basic process mining terminology and notations that ease the readability of the paper. There are different notations to depict a process model. In this paper, we used the Petri net [35] notation. Petri net is a directed bipartite graph that can be defined as follows.

Definition 1 (Petri net). *A Petri net is a graph that could be presented as triple (T,P,A) that T is a finite set of transitions, P is a finite set of places and A is a set of arcs that connect places to transitions and transitions to places.* $A \subseteq (T \times P) \cup (P \times T)$.

An example process model with Petri net notation is presented in Figure 1. In this figure, $T = \{a, b, c, d, e\}$, $P = \{p_1, \ldots, p_6\}$, and $A = \{(p_1, a), (a, p_2), \ldots, (e, p_6)\}$.

Given a set X, a multiset M over X is a function $M\colon X \rightarrow \mathbf{N}_{\geq 0}$, i.e. it allows certain elements of X to appear multiple times. We write a multiset as $M = [e_1^{k_1}, e_2^{k_2}, ..., e_n^{k_n}]$, where for $1 \leq i \leq n$ we have $M(e_i) = k_i$ with $k_i \in \mathbf{N}_{>0}$. If $k_i = 1$, we omit its superscript, and if for some $e \in X$ we have $M(e) = 0$, we omit it from the multiset notation. Furthermore, $M = [\,]$ denotes an empty multiset. $\overline{M} = \{e \in X \mid M(e) > 0\}$ is the set of elements present in the multiset. The set of all possible multisets over a set X is written as $\mathcal{M}(X)$.

Let X^* denote the set of all possible sequences over a set X. A finite sequence σ of length n over X is a function $\sigma\colon \{1, 2, ..., n\} \rightarrow X$, 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 ϵ. The

Table 1. Fragment of a fictional event log (each line corresponds to an event).

Case-id	Activity	Resource	Time-stamp
...	...	...	...
1	*register request (a)*	*Karl*	*2017-04-08:08.10*
1	*examine thoroughly (b)*	*Ali*	*2017-04-08:09.17*
2	*register request (a)*	*Karl*	*2017-04-08:10.14*
1	*check resources (c)*	*William*	*2017-04-08:10.23*
1	*check ticket (d)*	*William*	*2017-04-08:10.53*
2	*check resources (b)*	*Ali*	*2017-04-08:11.13*
1	*Send to manager(e)*	*Majid*	*2017-04-08:13.09*
1	*accept request (f)*	*Fatima*	*2017-04-08:16.05*
1	*mail decision(h)*	*Anna*	*2017-04-08:16.18*
...	...	...	...

concatenation of sequences σ and σ' is written as $\sigma \cdot \sigma'$. Function $hd\colon X^* \times \mathbf{N}_{\geq 0} \nrightarrow X^*$, 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 σ. In case $k = 0$ we have $hd(\sigma, 0) = \epsilon$. Symmetrically, $tl\colon X^* \times \mathbf{N}_{\geq 0} \nrightarrow X^*$ 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 σ, with, again, $tl(\sigma, 0) = \epsilon$. Sequence σ' is a subsequence of sequence σ, 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 σ' in σ by $freq\colon X^* \times X^* \to \mathbf{N}_{\geq 0}$ 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. An example event log is presented in Table 1.

Consider the events related to *Case-id* value 1. Karl *registers a request*, after which Ali *examines it thoroughly*. William *checks the ticket* and *checks resources*. Ava *sends the request to the manager* and Fatima *accepts the request*. Finally, Anna *emails the decision* to the client. The example trace is written as $\langle a, b, c, d, e, f, h \rangle$ (using short-hand activity names). In the context of this paper, we formally define event logs as a multiset of sequences of activities.

Definition 2 (Event Log). *Let $\mathcal{A}$ be a set of activities. An event log is a multiset of sequences over $\mathcal{A}$, i.e. $L \in M(\mathcal{A}^*)$.*

For example, $L_1 = [\langle a, b, c, e, g \rangle^6, \langle a, c, b, e, g \rangle^4, \langle a, b, c, e, f \rangle^3, \langle a, c, b, e, f \rangle^2, \langle a, d, e, f \rangle, \langle a, d, e, g \rangle, \langle a, b \rangle, \langle b, d, c, f \rangle \langle a, b, c, e, e, f \rangle]$ is an event log with 20 traces. Observe that each $\sigma \in L$ describes a *trace-variant* whereas $L(\sigma)$ describes how many traces of the form σ are presented within the event log. Therefore, in the above event log, there are 9 trace-variants and $L_1(\langle a, c, b, e, g \rangle) = 4$.

Sampling could be done with/without replacement. In sampling with replacement, it is possible to select the sampled objects more than one time. However, in this work, we use sampling without replacement. In other words, it is not possible to put a process instance more than once in the sampled event log. In the following, we define formally sampled event logs.

Definition 3 (Sampled Event Log). *We define S_L as a trace-based sampled event log of an event log L, if for each $\sigma \in S_L$, $S_L(\sigma) \leq L(\sigma)$. In the same way, S_L is a variant-based sampled event log of L if for all $\sigma \in S_L$, $S_L(\sigma) = 1$.*

In a variant-based sampled event log, the frequency of each selected process instances is equal to 1, however, in a trace-based event log, it is possible to have more than one traces of a unique process instance. In other words, a variant-based sampled event log is a subset of trace-variants in L. For example, $L_2 = [\langle a, b, c, e, g\rangle, \langle a, c, b, e, g\rangle, \langle b, c, e, f\rangle]$ is a variant-based sampled event log of L_1.

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

Definition 4 (Directly Follows Relation). *Let a and $b \in \mathcal{A}$ be two activities and $\sigma = \langle \sigma_1, .., \sigma_n\rangle$ is a trace in the event log. A directly follows relation from a to b exists in trace σ, 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 $\sigma = \langle a, b, c, e, g\rangle$, we have $c >_\sigma e$, but $d \not>_\sigma a$.

An alternative behavior which has negative affects 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 [20].

Definition 5 (Behavioral Context). *A behavioral context c is a pair of 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:*

$$\beta_L = \{(\sigma_l, \sigma_r) \in \mathcal{A}^* \times \mathcal{A}^* \mid \exists \sigma \in L, \sigma' \in \mathcal{A}^* \, (\sigma_l \cdot \sigma' \cdot \sigma_r \in \sigma)\} \tag{1}$$

For example, in trace $\sigma = \langle a, b, c, e, g\rangle$, $\langle a, b\rangle$ and $\langle e\rangle$ are two subsequences that surround $\langle c\rangle$, hence, the pair $(\langle a, b\rangle, \langle e\rangle)$ is a behavioral context.

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

Definition 6 (Conditional Contextual Probability). *Let $\sigma_s, \sigma_l, \sigma_r \in \mathcal{A}^*$ be three sequences of activities and let $L \in \mathcal{M}(\mathcal{A}^*)$ be an event log. We define the* conditional contextual probability *of σ_s, w.r.t., σ_l and σ_r in L, i.e., representing the sample based estimate of the conditional probability of σ_s being surrounded by σ_l and σ_r in L. Function $\gamma_L \colon \mathcal{A}^* \times \mathcal{A}^* \times \mathcal{A}^* \to [0, 1]$, is based on:*

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

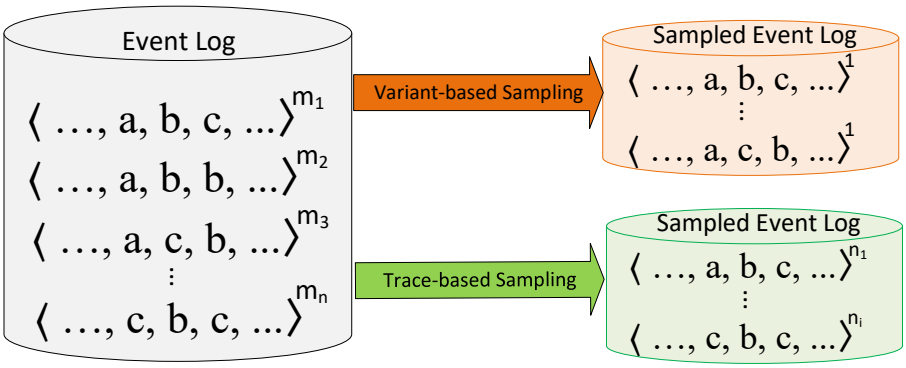

Fig. 2. Schematic overview of instance selection (sampling) methods. We select variants or traces based on different criteria. In variant-based sampling, each variant in a sampled event log represents all the traces with this behavior in the original event log.

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

For example, to compute $P_{L1}(\langle c \rangle \mid \langle a, b \rangle, \langle e \rangle)$, we need to compute in how many traces, subsequence $\langle a, b, c, e \rangle$ is occurred and divide it to the number of traces that contain the pair $(\langle a, b \rangle, \langle e \rangle)$ as a behavioral context that is $\frac{10}{10} = 1$. Based on these probabilities, we are able to detect unstructured behavior in a trace.

4. Sampling Event Data using Instance Selection

In this section, we present different biased instance selection strategies to sampling event logs and consequently increase the discovery procedure's performance. We are able to sample different behavioral elements of an event log, e.g., events, directly follow relations, traces, and variants. By sampling events, it is possible to choose events from different parts of a process instance that is harmful to the process discovery purpose. Sampling directly follows relations is useful for some process discovery algorithms like the Alpha Miner. For example, choosing the most frequent directly follows relations. But, we need to modify these algorithms to accept a set of directly follows relations instead of an event log as an input. However, it may result in process models with some unconnected components. Also, such data structures are not applicable to all process discovery algorithms. Thus, here we just consider trace and variant-based sampling. Therefore, here we just focus on sampling methods that take an event log as an input and return a subset of traces or variants. In other words, the sampling methods select some variants/traces in the input event logs.

The schematic of the instance selection methods is illustrated in Figure 2. In variant-based sampling, the frequency of each sample is 1. In other words, we select only one process instance for each selected variant. However, 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 the ILP Miner, the family of Alpha miners and the Inductive Miner, it is enough to have unique variants to discover a corresponding process model. The frequency of variants is mostly just used for post-processing algorithms like filtering. Therefore, here we mainly focus on variant-based sampling, but, all these methods can easily be extended to trace-based sampling methods. 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 a sampled event log. 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., no reduction in size. We can set the size of sampled event logs (i.e., the sample ratio) as follows.

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

Note that, in the above equation, $0 < c \leq 1$ and denotes that how many percentages of traces (or variants in variant-based sampling) are selected in the sampled event log.

Another 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, it is not required that sampled event logs include all the behavior in the original event log, because many process discovery algorithms are able to generalize the seen behavior. Finally, the last dimension is the sampling time as a preprocessing phase. It is better to sample an event log in a shorter time.

It is possible to sample process instances or variants of an event log randomly or selecting them based on a bias. 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 sampling) or just unique trace-variants among them (i.e., variant sampling). The time complexity of this method is $O(k)$ where k is the number of sampled traces. Therefore, this method is fast because there is no need to traverse the original event log. In this way, we have no control over the number of selected variants. In other words, it is possible that many of the sampled traces have similar behavior and we return just a few unique variants.

Another approach is to first find all the unique variants in an event log, after that, randomly select $c \times |\overline{L}|$ variants from them. The time complexity of traversing an event log is $O(n \times l)$ where $n = |L|$ is the number of traces, and l is the average length of traces in the event log. This approach is a bit slower, but it is able to return more behaviors compared to the previous approach.

4.2. Instance Selection Strategies

In general, traversing an event log is not a very time-consuming phase, and this gives us a motivation that instead of randomly sampling the variants, we are able to use more

advanced strategies (biases) to select variants in the event log. Note that although in XES standard [48], we need to traverse an event log to find the variants, in many other structures variants with some of their properties are stored as meta-data [16].

In biased sampling methods, we first traverse the event log to find unique trace-variants in it, then rank them based on different strategies. Afterward, we return the top $c \times |\overline{L}|$ variants with the highest rank in the sampled event log. For the trace-based sampling methods, we return all process instances that correspond to each trace-variant (with considering the c value). We are able to use different ranking strategies for selecting the trace-variants that will be discussed in follows.

Frequency-based Sampling

The first ranking strategy is sampling variants based on their frequencies. This sampling method gives more priority to a trace-variant that has a higher occurrence frequency in the event log. Thus, 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. For example, for L_1 if we want to select only one variant based on their frequencies, $\langle a, b, c, e, g\rangle$ will be selected. The time complexity of this strategy is $O(n{\times}l + r)$ where $n{\times}l$ corresponds to traversing the event log and r represents the required time that for ranking variants $\sigma{\in}\bar{L}$.

The advantage of this strategy is that we are able to have 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, there are lots of variants with very low frequencies. Sometimes, the majority of process instances have a unique trace-variant. Thus, differentiating between them will be challenging. Therefore, a drawback of this strategy is that the sampled event log may not contain many behaviors in the original event log. The trace-based version of this instance selection bias is existed in many process mining tools. They used this selection bias to reduce the complexity of the resulted process model and show the main stream behavior of the process model.

Length-based Sampling

We are able to select process instances or trace-variants based on their length, i.e., the number of activities in the trace or $|\sigma|$. If we want to keep more behaviors in our sampled event log, we need to select longer traces first. However, if we are interested in retaining the main-stream behaviors of the event log, it is usually better to choose shorter variants. For the running example, we will chose $\langle a, b, c, e, e, f\rangle$ and $\langle a, b\rangle$ first if we use longer and shorter strategies respectively.

Selection based on 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 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 σ has $a >_\sigma a$ one time or more. On the other hand, with selection based on *shorter* strategy, if there are incomplete traces in an event log, we probably will have them in the sampled event log. Similar to the frequency-based strategy, the time complexity of this approach is $O(n{\times}l + r)$.

Similarity-based Selection

If we aim to sample variants that contain general behavior of the whole event log, we need to use the similarity-based sampling methods. In 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 $B_i = (a, b)$ (that $a, b \in \mathcal{A}$) according to the following formula.

$$Prob(B_i) = \frac{|\{\sigma \in \overline{L} : a >_\sigma b\}|}{|\overline{L}|} \tag{4}$$

Therefore, 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 $0 \geq T_P \geq 1$), we expect that sampled variants also should contain it. Thus, any variant contains such a high probable behavior (that here is a directly follows relations), will give a $+1$ to its rank. Otherwise, if a variant does not contain a probable behavior, we decrease its rank by -1. For example, (e, f) is a high probable relation with $Prob(e, f) = \frac{5}{9}$. Note that by using a higher T_P value, we will have fewer high probable and low probable relations; therefore, many variants will have similar ranks.

Contrariwise, if a variant contains a low probable behavior (i.e., $Prob_{B_i} \leq 1 - T_P$), we decrease its rank by -1. In other words, we are searching for variants that have much high probable behaviors and have less low probable behaviors. For example, $Prob(e, e) = \frac{1}{9}$ is a low probable relation. Note that, it is possible that some behaviors be neither high probable nor low probable that we do nothing for such behaviors. Afterward, we sort the variants based on their ranks and return the $c \times |\overline{L}|$ ones with the highest rank. For example, $\langle a, d, e, f \rangle$ is the first variant that will be chosen using this strategy for log L_1.

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. Another disadvantage of this strategy is that the length of variants affects the final rank. It is possible that two variants have a similar set of directly follows relations, but based on the frequency of these directly follows, they will have different ranks. The time complexity of this strategy is $O(n \times l + d + r)$ where d corresponds to finding probabilities of directly follows relations and in the worst case, its complexity is $O(|A|^2 \times \bar{L})$.

Structured-based Sampling

In this sampling method, we consider the presence of unstructured behavior (i.e., based on Definition 6) in each variant. In this regard, we first compute the occurrence probability of each sub-pattern 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., $0 \geq T_S \geq 1$, we call it an odd structure. We expect that unstructured subsequences are the most problematic behavior in event logs that make discovered process models inaccurate and complex [20]. Thus, for each unstructured behavior in a trace-variant, we give a penalty to it and decrease its rank by -1. For example, $P_{L1}(\langle d \rangle | \langle b \rangle, \langle c \rangle)$ is very low and $\langle b, d, c, f \rangle$ will give a penalty for it.

Consequently, a variant with higher odd structures receives more penalties, and it is not appealing to be placed in the sampled event log.

By choosing a high T_S value, most of behaviors will be considered as odd structures. In contrast, a very low T_S will result in few unstructured behaviors that leads to have many variants with the same rank. Note that, by increasing the length of behavioral context, we will detect more important nonstructural behaviors. Similarly, by decreasing the T_S value, the detected behavioral patterns will be more problematic.

Similar to similarity-based selection strategy, in this strategy, the length of variants will affects their rank. Using this strategy, we select the most well-structured trace-variants in the event log, that is expected to have process models with high precision value. However, it is possible to select trace-variants that not frequent in the input event log. The time complexity of this strategy is $O(n \times l + c + r)$ where c corresponds to conditional contextual probabilities and in the worst case, its complexity is $O(|A|^m \times \bar{L})$ where m is the maximum length of considered contextual behavior and the substring. Therefore, this strategy is potentially the slowest strategy for instance selection.

Hybrid Methods

In a hybrid strategy, we are able to combine two or three of other sampling strategies. In this way, we expect to have benefits of different methods. Here, we combine the frequency-based and similarity-based methods. In this regard, when we compute the probability of each directly follows relation, the frequency of it in the whole event log will be used according to the following formula.

$$Prob(a, b) = \frac{\Sigma_{\sigma \in \bar{L}_{a>b}} (L(\sigma))}{|L|} \tag{5}$$

In the above equation, $\bar{L}_{a>b} = \{\sigma \in \bar{L} : \ a >_\sigma b\}$. Note that, other combinations are also possible.

Different ranking strategies require different sampling time and result in different sampled event logs according to their completeness. In the next section, we will analyze how these selection strategies affect the performance and quality of process discovery algorithms on real event logs.

5. Evaluation

In this section, by doing experiments, we aim to answer the following research questions:

- **Q1**) Are the proposed instance selection strategies able to improve the performance of process discovery?
- **Q2**) What are the effects of instance selection on the quality of discovered process models?

In this regard, we first explain the implementation details of the proposed method. Afterward, the experimental settings for the experiments will be discussed, and finally, the results will be explained.

Table 2. Some information of real event logs that are used in the experiment.

Event Log	Activities#	Traces#	Variants#	Directly Follows Relations#
BPIC_2012 [45]	23	13087	4336	138
BPIC_2013 [49]	4	7554	1511	11
BPIC_2017_All [46]	26	31509	1593	178
BPIC_2017_Offer [46]	8	42995	169	14
BPIC_2018_Control [15]	7	43808	59	12
BPIC_2018_Inspection [15]	15	5485	3190	67
BPIC_2018_Reference [15]	6	43802	515	15
BPIC_2019 [47]	44	251734	11973	538
$Hospital$ [32]	18	100000	1020	143
$Road$ [14]	11	150370	231	70
$Sepsis$ [33]	16	1050	846	115

5.1. Implementation

To apply the proposed instance selection strategies, we implemented the *Sample Variant* plug-in in `ProM` framework[3]. ProM is an academic platform with several plug-ins that cover wide domains of process mining areas [4]. In this implementation, we used static thresholds for both similarity and structure-based ranking strategies. For similarity-based selection, we just considered the directly follows relation as explained in the previous section. Using this plug-in, the end-user is able to specify her desired percentage of sampling variants/traces and the instance selection strategy. It takes an event log as an input and returns the top fraction of its variants/traces. A screen-shot of the settings of the instance selection plug-in in ProM is presented in Figure 3. By adjusting these settings, users will be able to have different sampled event logs.

In this implementation, we used T_P=0.8 and T_S=0.2. Moreover, for behavioral contexts' subsequences, we consider the maximum length equals to 2.

Furthermore, to apply our proposed method on various event logs and use different process discovery algorithms with their different parameters, we ported the *Sample Variant* plug-in to `RapidProM` [3] which extends `RapidMiner` with process analysis capabilities. In our experiments, we also used the statistical sampling method that is presented in [8]; however, as we consider only work-flow information, its relaxation parameter is ignored.

5.2. Experimental Setup

For analyzing the effect of sampling, we applied the proposed method on several real event logs. Some information about these event logs is given in Table 2. We added artificial *start* and *end* activities to the whole process instances of all event logs. For process discovery, we used the Alpha Miner [5], the Inductive Miner (IM) [28], the Inductive Miner with its infrequent behavior filtering mechanism (IMi) [29], the ILP Miner [43], and the Split Miner [7]. For computing the quality of discovered process models, the original event logs were used.

[3] Sample Variant plug-in in: svn.win.tue.nl/repos/prom/Packages/LogFiltering.

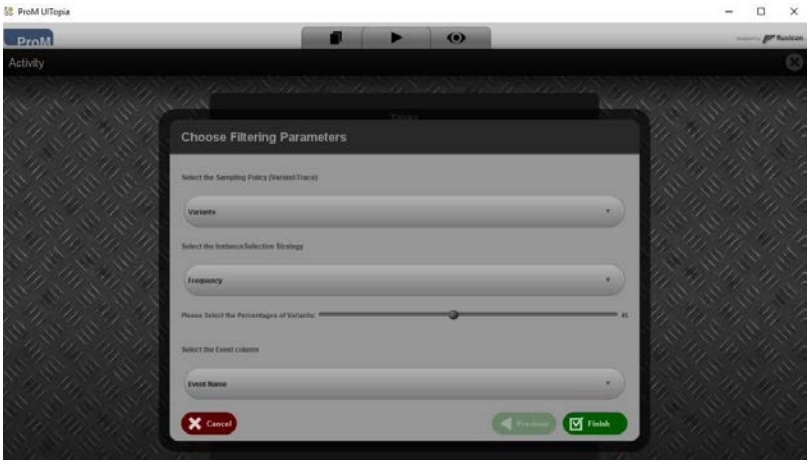

Fig. 3. A screen-shot of the implemented instance selection plug-in in ProM platform. It receives an event log and return a sample event log based on the adjusted settings.

We sampled event logs with different variant and trace-based instance selection strategies, and c with different values that are $[1, 2, 3, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100]$ which c presents how many percentages of traces or variants of the event log is selected to be placed in the sampled event log. Therefore, if we use the trace-based sampling policy with $c = 100$, the sampled event log equates to the original event log. As sampling time and discovery time are nondeterministic, each experiment was repeated four times.

5.3. Experimental Result

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

Performance Analysis

To measure the improvement in the performance, we consider both the discovery time and sampling time of event logs using the following formulas:

$$DiscoveryTimeImprovement = \frac{DiscoveryTime_{WholeLog}}{DiscoveryTime_{SampledLog}} \tag{6}$$

$$TotalTimeImprovement = \frac{DiscoveryTime_{WholeLog}}{DiscoveryTime_{SampledLog} + SamplingTime} \tag{7}$$

In discovery time improvement, we measure how discovering a process model is faster on a sampled event log compared to the original event log. However, in total time improvement, the sampling time is also considered. Therefore, it measures how sampling an event log and discovering a process model of it, is faster compared to discover a process model from the original event log. An improvement value shows how many times a process model will be discovered faster by using the sampled event log.

Table 3. Process discovery performance improvement for different process discovery algorithms using variant and trace-based instance selection methods when $c = 10$.

Discovery Algorithm	Event Log	Normal Discovery Time (ms)	Discovery Improvement			Discovery Algorithm	Normal Discovery Time (ms)	Discovery Improvement		
			Statistical	Traces	Variants			Statistical	Traces	Variants
Alpha Miner	BPIC-2012	154	4	7	50	Inductive Miner (IMi)	3,571	4	7	18
	BPIC-2013	94	6	8	17		890	9	11	48
	BPIC-2017-All	1,349	7	9	38		70,652	11	27	50
	BPIC-2017-Offer	303	38	14	758		2,108	44	14	727
	BPIC-2018-Control	234	585	9	41		1,707	43	9	557
	BPIC-2018-Inspection	252	2	10	45		4,522	2	16	39
	BPIC-2018-Reference	186	24	9	61		1,511	28	7	184
	BPIC-2019	1,817	14	6	25		62,021	16	10	42
	Hospital	687	14	8	224		17,366	35	29	1,192
	Road	660	42	6	1,650		6,988	68	11	12,332
	Sepsis	20	1	7	51		317	1	51	48
ILP Miner	BPIC-2012	2,4875	4	6	9	Split Miner	903	3	5	8
	BPIC-2013	1,108	4	4	7		537	6	8	25
	BPIC-2017-All	1,088,584	30	14	41		11,479	9	13	31
	BPIC-2017-Offer	2,279	7	6	12		1,599	25	9	147
	BPIC-2018-Control	2,511	11	6	19		1,964	36	8	353
	BPIC-2018-Inspection	7,746	1	6	5		1,755	2	7	17
	BPIC-2018-Reference	2,390	9	5	14		1735	27	7	294
	BPIC-2019	95,697	5	3	7		19,520	15	10	41
	Hospital	10,995	5	7	8		3,388	13	8	64
	Road	6,144	10	5	10		3601	46	8	171
	Sepsis	7,464	1	5	4		180	1	4	3

Table 3 and Figure 4 show the improvement when we sample event logs. The y-axis in these figures represents the average performance improvements using a logarithmic scale. Values below 1 show that there is no improvement in performance. Here, we consider $c = 10$ and all instance selection strategies. *Statistical* corresponds to the sampling method that is proposed in [8] (with its default setting). It is clear that by reducing the size of an event log, the required process discovery time is reduced. Therefore, the $DiscoveryTimeImprovement$ for variant-based selection strategies is significantly higher than the trace-based policy. Note that all process discovery algorithms traverse the event log at least once that requires $O(n \times l)$. Therefore, $n = |S_L|$ for variant-based sampled event log is usually extremely lower than the trace-based one. For some event logs, a process discovery algorithm is more than 1000 times faster on the sampled event log using variant-based sampling compared to using the whole event logs. 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. We see that the Statistical method is not able to improve the total discovery time of *Sepsis* event log for any process discovery algorithm. As it is shown in Table 2, almost all of trace-variants in this event log are unique that cause this sampling method selects almost all the traces in the event log. For the Alpha miner, we usually do not have improvement. This process discovery algorithm is simple, and its main time-consuming task is to find out the directly follows relations in the event log. However, for advanced process discovery algorithms, e.g., the Inductive Miner, the total improvement is very high. Comparing these two figures, we find that the sampling time for the statistical sampling method is much faster than our techniques, but the discovery time of sampled event logs is less using the proposed instance selection strategies.

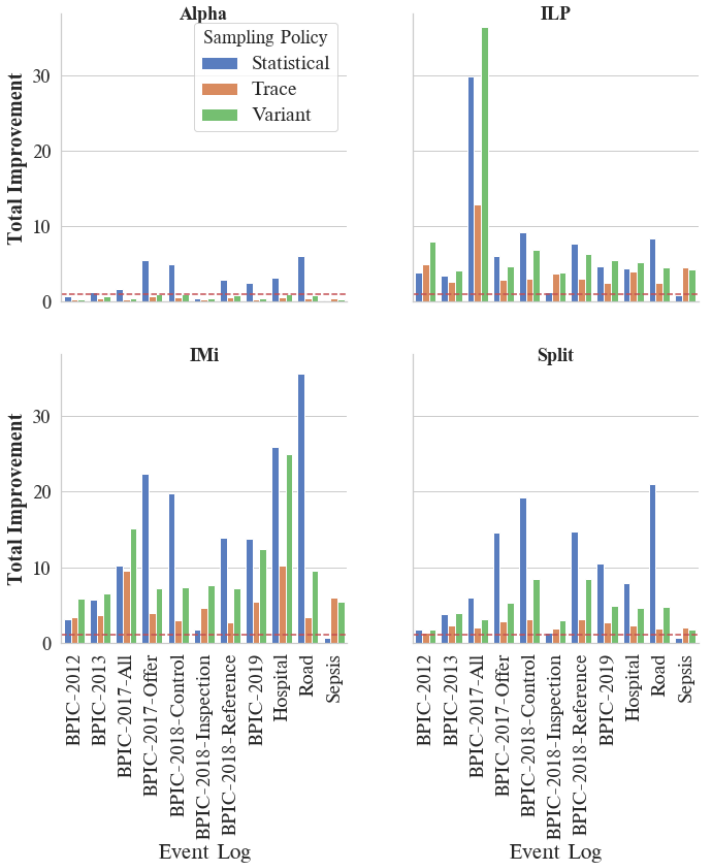

Fig. 4. Total time improvement for discovering process using sampling methods.

Figure 5 shows improvements in the performance of process discovery algorithms when we select some instances of event logs. Here, for instance selection strategies, we used the sampling threshold (i.e., c) equals to 0.1 and the variant-based policy. It shows that the improvement is usually lower when the statistical sampling method even it does not need to traverse the input event log. As it is mentioned before, for *Sepsis* event log, this method has no improvement. Among instance selection methods, *Longer* strategy usually results in lower improvement. The computation time of process discovery algorithms depends on different parameters e.g., the number of activities (i.e., $|A|$), the number of variants (i.e., $|\bar{L}|$) and sometimes the prefix-closure of the given event log. Using instance selection strategies, we are able to reduce all the above parameters and consequently decrease the required time for process discovery.

The sampling time of different methods is depicted in Figure 6[4]. As we expected, except for the *Sepsis* event log, the statistical sampling method is much faster than instance

[4] We do not show the results for all event logs.

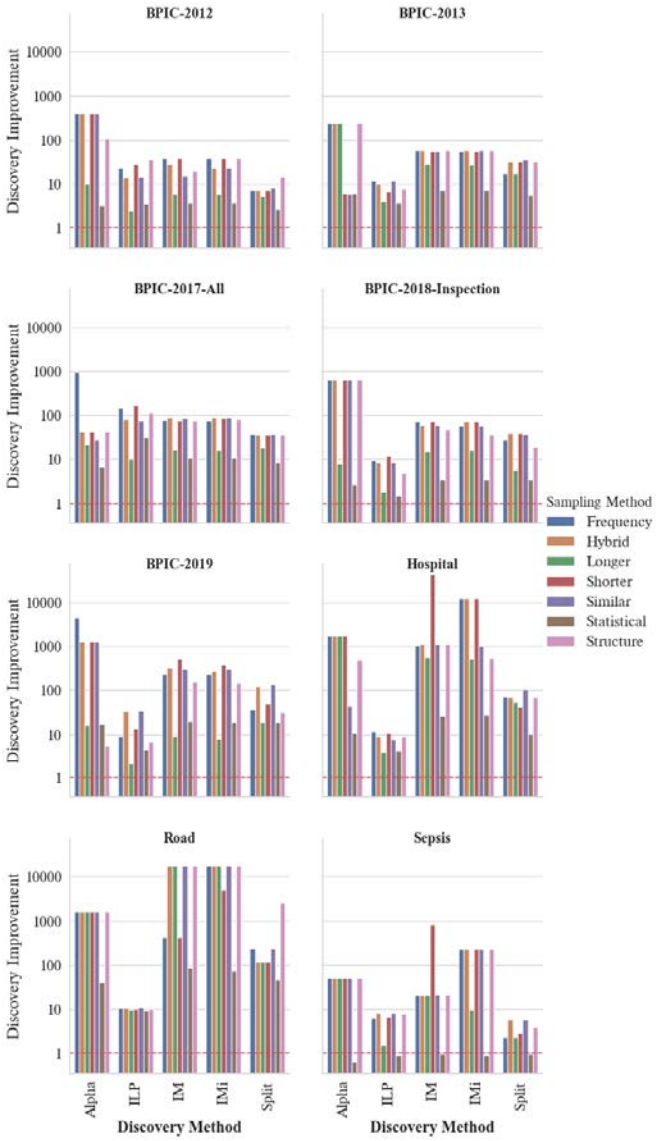

Fig. 5. Discovery time improvement for discovering process using instance selection.

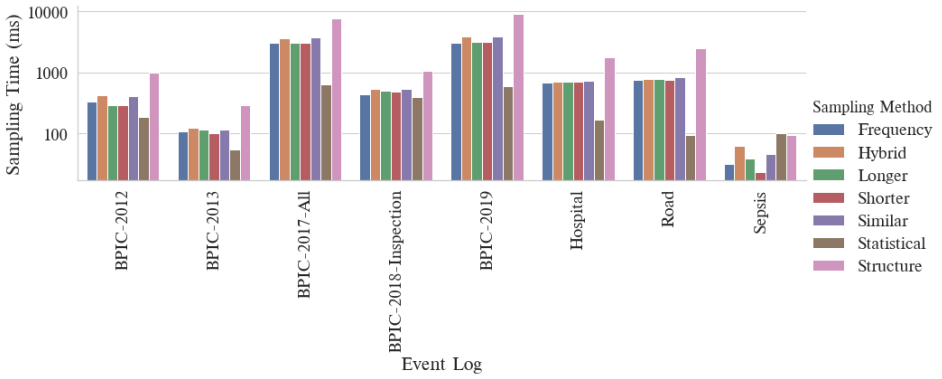

Fig. 6. Sampling time of different instance selection strategies.

selection strategies as it does not need to traverse the event log. Among the instance selection strategies, the structure-based is the slowest one as it needs to consider more complex behavioral structures. Frequency and length based strategies are required almost the same time to sampled event logs. As it is mentioned, in case that we use MXML or any other structure that contains meta-data of variants, the sampling time for these strategies will be much less.

As event filtering algorithms are able to reduce the variability of event logs, we are able to compare them with the instance selection methods. In this regard, we filter event logs using Anomaly Free Automaton (AFA) [13] and Matrix Filter (MF) [18] using their default settings. For sampling, we used the variant-based sampling using $c = 10$ and the similarity based strategy. Here, for process discovery, we used the IMi algorithm with its default setting. The results of this experiment is shown in Table 4. We could not have the filtered event log for *BPIC-2018-Inspection* using the AFA method. Results show that the preprocessing time for the instance selection method is less than other filtering methods. For most of the event logs, process discovery on the preprocessed event log using the proposed approach is faster mostly because it returns the variant-based sampled event logs. The discovery time for some filtered event logs is significantly low that is because of removing lots of behavior during filtering.

We aim to analyze what is the origin of the performance improvement. The improvement in performance of process discovery algorithms may be driven by reducing (1) the number of activities, (2) the number of traces, or (3) the amount of unique behavior (e.g., DF relations or variants). By event log instance selection, it is possible that some of the infrequent activities are not placed in the sampled event log. Moreover, by sampling we reduce the size of event logs (i.e., $|S_L| \leq |L|$), specifically when we apply the variant-based strategies. Finally, it is also possible that we reduce the number of unique behavior in the event log.

Figure 7 shows the process discovery time of sampled event logs with different instance selection thresholds when we apply the ILP miner. Here, to focus on the reason for improvements, we used variant-based strategies. When we used the sampling size equals to 100, we will have all the behaviors of the original event log in the sampled event log;

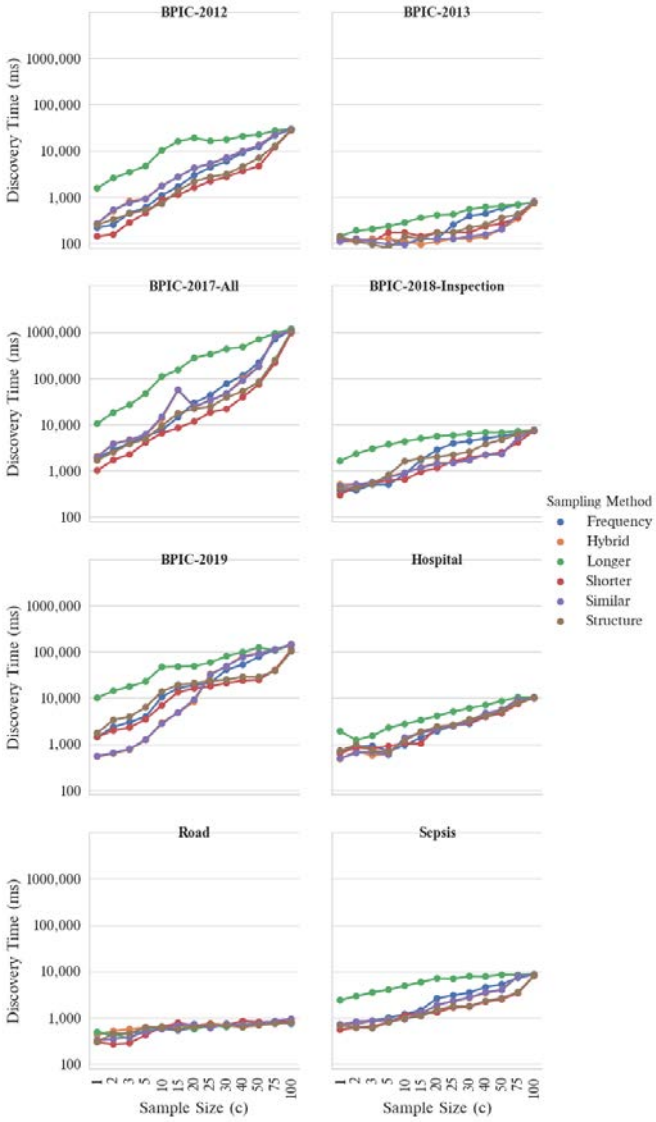

Fig. 7. The average of discovery time of the ILP miner for different instance selection methods with different sampling size (i.e., *c*).

Table 4. Comparing the instance selection and filtering methods based on the required time of them for preprocessing and process discovery.

	Preprocessing Time			Discovery time			Total Discovery Time		
Event Log	AFA	MF	Sampling	AFA	MF	Sampling	AFA	MF	Sampling
BPIC-2012	26,705	593	749	379	19978	140	27084	20571	889
BPIC-2013	3246	312	187	435	749	16	3681	1061	203
BPIC-2017-All	57927	3188	5212	2	1034	859	57929	4222	6071
BPIC-2018-Inspection	$\sim$	1192	649	$\sim$	4	170	$\sim$	1196	819
BPIC-2019	804913	6663	4403	7115	312	219	812028	6975	4622
Hospital	27442	2122	828	2014	19767	15	29456	21889	843
Road	13229	2763	937	3418	4478	1	16647	7241	938
Sepsis	12028	62	47	2	4384	1	12030	4446	48

however, the number of traces in the sampled event log is significantly lower than the original event log. The greatest improvement is usually gained when we use the similarity or shorter strategies for this process discovery algorithm. Results show that for many event logs, by increasing the percentage of selected variants, how will reduce the improvement. Therefore, for many event logs, the main reason for the improvement in performance of the process discovery is gained by reducing the number of variants. However, in *Road* and *Hospital* event logs that there are high-frequent variants, reducing the number of traces has a higher impact on the performance of the process discovery algorithms.

As it is explained, the amount of behavior in the event log also has an important role in the performance of process discovery algorithms. The remaining percentage of *DF* relations in the sampled event logs for different instance selection strategies are given in Figure 8 when we increased the size of the selected percentage of variants (i.e., c). To compute the remained DF relations, we divide the number of existing relations in the sampled event log to all DF relations in the original event log. We see that for most of the event logs, the similar and structure-based selection strategies keep fewer DF relations. However, according to their ranking methods, they keep the most common DF relations among the original event log. Moreover, it is shown that for many event logs, with selecting only 50% of the variants, we are able to keep more than 80% of the DF relations in the sampled event log when the frequency or shorter strategies are used. Unlike our expectations, by using the shorter strategy for selecting variants, we will have high percentages of Df relations in the sampled event log. It happens because using this strategy, lots of infrequent trace-variants that may be incomplete, are also selected to be kept in sample event logs.

Note that there is no such control like the c value for the statistical sampling method and it aims to keep as much as DF relations in sampled event logs. However, for most of process discovery algorithms variants are more important compared to only DF relations. Even the basic Inductive miner that uses DF relations may result in different process models for event logs that have identical sets of DF relations. For example, $L_2 = [\langle a, b, c\rangle, \langle a, c, b\rangle]$ and $L_3 = [\langle a, c, b, c\rangle, \langle a, b\rangle]$ have the same sets of DF relations; however, their process models are different. Figure 9 indicates that the average percentage of the remaining variants in sampled event logs using the statistical sampling method [8].

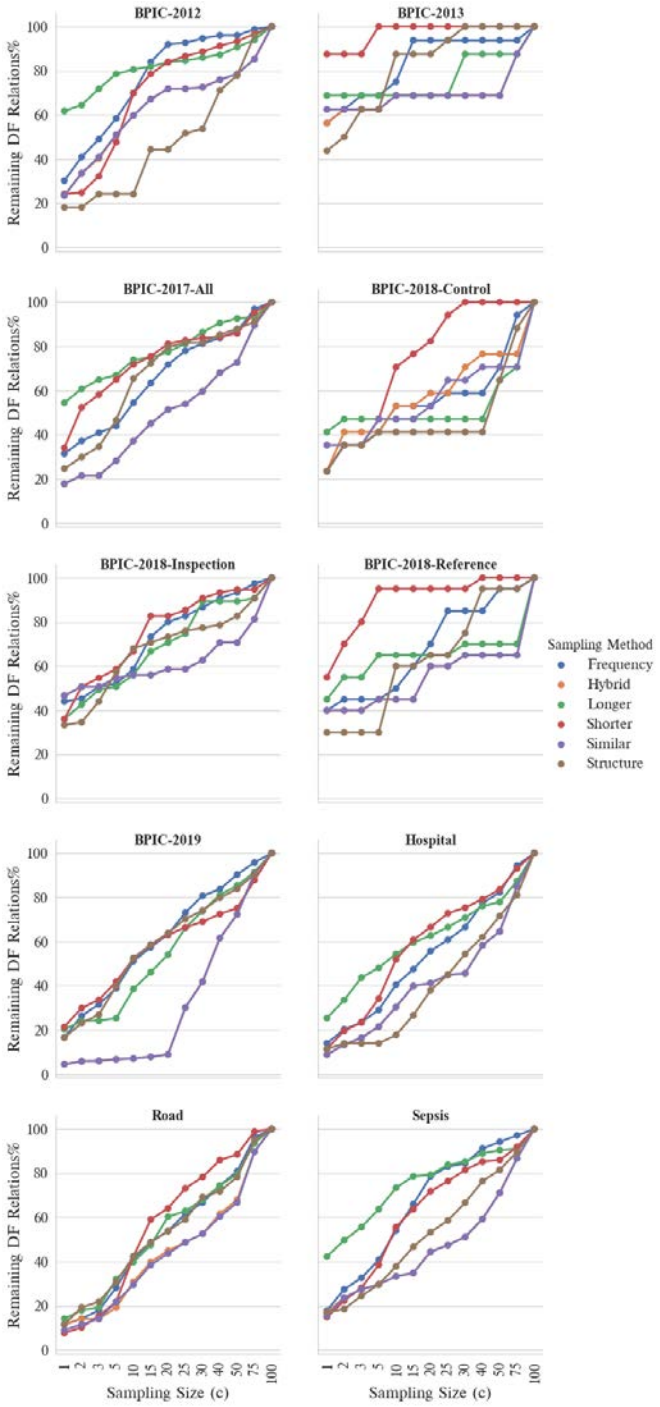

Fig. 8. Remained percentage of DF relations in the sampled event logs using instance selection strategies with different sampling size.

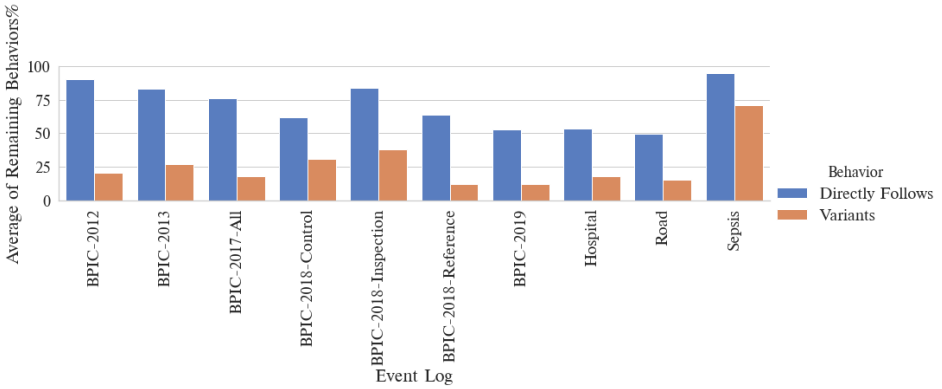

Fig. 9. Average of remained variants and directly follows relations in sampled event logs using the statistical sampling method [8].

It shows this method is able to keep a few percentages of variants of an event log (using the default setting).

According to these experiments, we find that using instance selection improves the performance of process discovery algorithms, specifically if we apply the variant-based policy. Moreover, we show that instance selection strategies that only uses the variant meta-data, e.g., frequency or length-based methods are faster than other ones. The instance selection strategies improve the performance of process discovery algorithms by reducing the number of process instances, variants and behaviors in sampled event logs. Note that, many times we consider sampling as a preprocessing phase, and we apply different process mining algorithms, e.g., process discovery several times on the preprocessed event log. As explained in Section 2, there are some preprocessing algorithms that by removing traces with outlier behaviors will reduce the size of the event log. To compare the performance of instance selection strategies with other related preprocessing algorithms, in Figure 10, we examined the average of preprocessing time of three trace filtering methods and instance selection methods. For some event log, we are not able to have the preprocessed event logs. For this experiment, we filter event logs with four different filtering settings and iterate the experiments for four times. The result shows that the instance selection method preprocessed the event logs much faster. Note that, in these filtering methods, we do not have accurate control over the size of the preprocessed event logs.

Quality Analysis

There are some research has been done on measuring the quality of business process models [34, 23, 37]. Here, to analyze the quality of process models that are discovered on sampled event logs, we use *fitness* and *precision* metrics. Fitness measures how much behavior in the event log is also described by the process model. Thus, a fitness value equal to 1, indicates that all behavior of the event log is described by the process model. Precision measures how much of behavior, that is described by the process model, is also

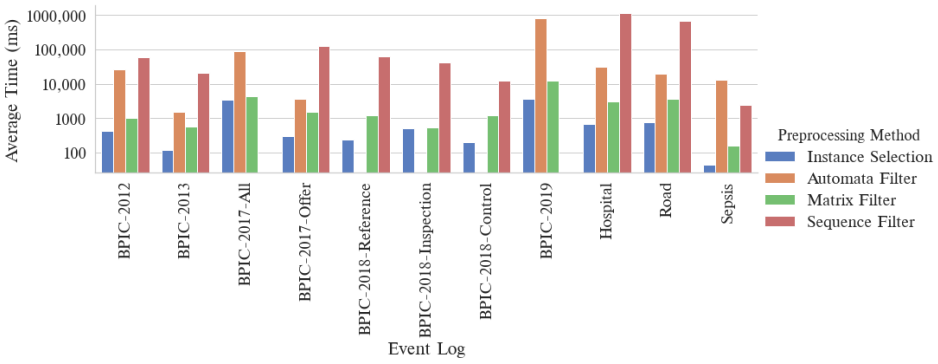

Fig. 10. The average of preprocessing time when we used instance selection strategies and three state-of-the-arts trace filtering methods [13, 18, 19].

presented in the event log. A low precision value means that the process model allows for much behavior compared to the given event log. There is a trade-off between these measures [50], sometimes, putting aside a small amount of behavior causes a slight decrease in the fitness value, whereas the precision value increases significantly. Therefore, we use the F-Measure metric that combines both of them with the same weight according to the following formula[11]:

$$\textit{F-Measure} = \frac{2 \times \textit{Precision} \times \textit{Fitness}}{\textit{Precision} + \textit{Fitness}}. \tag{8}$$

We did not consider the results of the Alpha miner algorithm as it may discover unsound process models, and consequently, it is not possible to compute the fitness value of them. Figure 11 compares the quality of best process models that are discovered with/without the instance selection method. We used sampled event logs just for discovery purpose, and the original event logs were used for computing F-Measure values. For the cases that the instance selection methods were used, we applied the sampled size in [1, 2, 3, 5, 10], and the average of F-Measure values is shown. For the statistical sampling method, we iterate the experiment four times, and again the average of F-Measure values are considered. For the *IMi* method, we used nine different filtering thresholds, but for other discovery methods, just the default setting was used. According to results that are presented in Figure 11, for the ILP, we always have an improvement when we use the instance selection method as a preprocessing step. However, the Split miner can usually discover process models with higher quality via the original event logs or using the statistical sampling method. Moreover, the statistical sampling method that randomly selects process instances results in process models with similar quality according to the F-Measure compared to original event logs. Note that instance selection strategies usually improve the quality when the size of the event log is considerable, e.g., *BPIC-2017-All* and *BPIC-2019*. In cases that the instance selection method could not improve the quality of process models, its quality is not much worse than using original event logs.

In Figure 12, the maximum F-Measure value of each instance selection strategy is shown using the previous setting. It is shown that for some event logs that there are

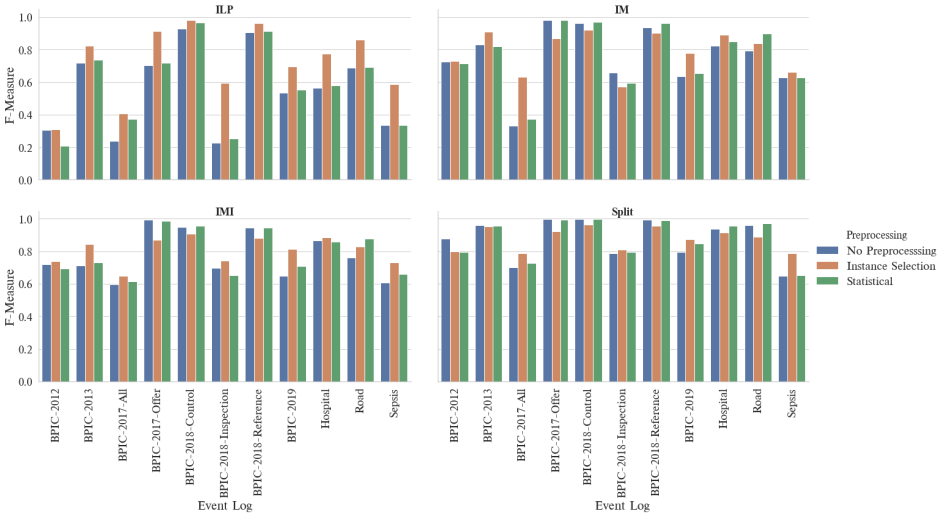

Fig. 11. Comparing the average F-Measure values of discovered process models with different subset selection methods.

high-frequent trace-variants, e.g., *Road* the frequency strategy outperforms other methods. Among ranking strategies, the longer strategy results in worse process models for most of the event logs.

For some event logs like *BPIC-2017-All*, the similarity-based selection strategy is the best choice for all of the process discovery algorithms. For the *Hospital* event log, the structure strategy usually results in process models with the highest F-Measure. Note that here, we show the maximum value of F-Measure, when we used different size of sampled event logs. Finding the best size and selection strategy is a challenging task, but as shown in this figure, we usually expect to have higher quality compared the original event log (i.e., "*NO Preprocessing*").

To understand a process model, the simplicity of it plays an important role. To measure the simplicity of a process model, we consider three metrics that measure the complexity of it. *Size* of process models is a combination of the number of transitions, places and arcs that connected them. Note that we used the Petri net notation for process models [35]. Another metric is the Cardoso metric [27] that measures the complexity of a process model by its complex structures, i.e., *And*, *Or*, and *Xor* components. Containing more complex structures requires much times to understanding behaviors, and consequently, it increases the complexity of the process model. There is also another metric called *Structuredness* that measures how different structural components in a Petri net is wrapped to each other [27]. It is expected that a Petri net with having more wrapped components is more difficult and requires much time to be understand. For all of these three measures, a lower value means less complexity and consequently a simpler process model.

In Table 5, we compared the complexity of process models when different preprocessing methods are applied. Here we used the same setting as explained in the previous experiment. For both the statistical and instance selection methods, we present the aver-

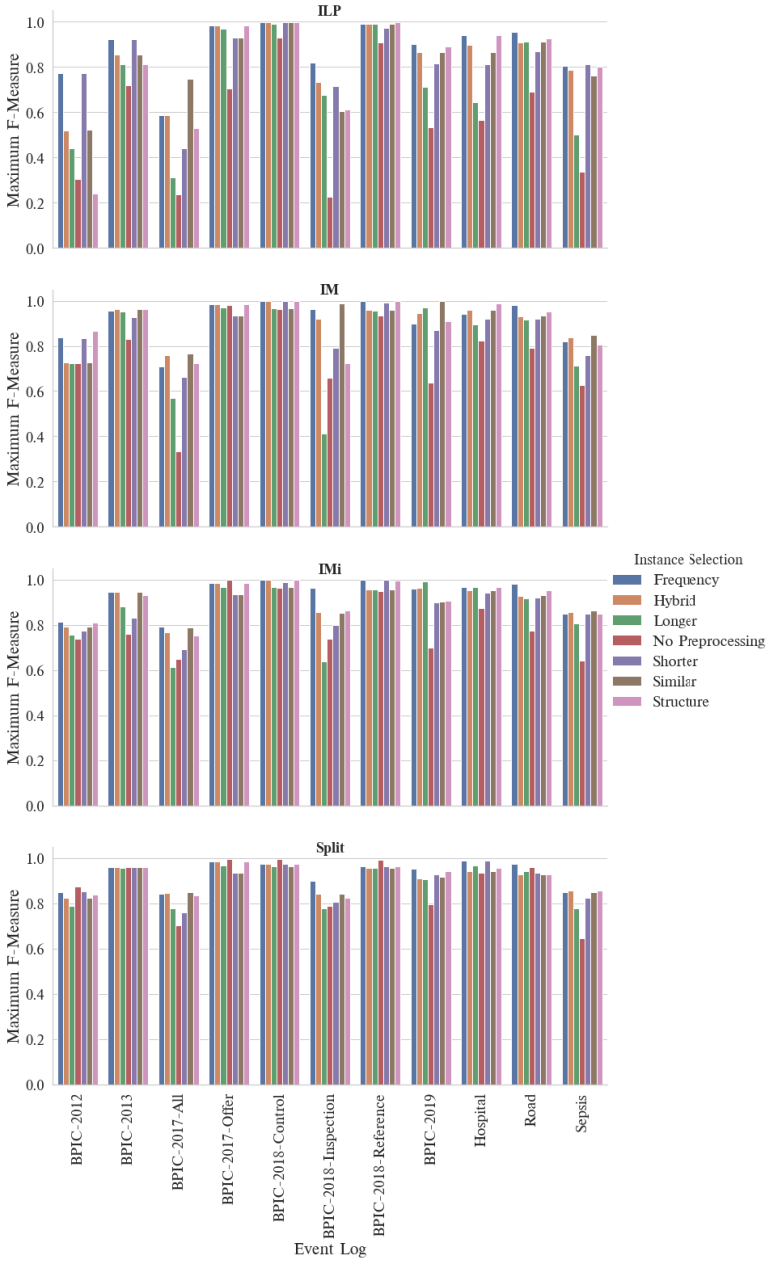

Fig. 12. The maximum F-Measure of discovered process models using different instance selection strategies.

Table 5. The average complexity metrics values of discovered process models using different process discovery algorithms. The size is representing the number of arcs, places, and transitions. The null values correspond to unsound process models.

Alpha Miner										ILP Miner								
Preprocessing	Instance Selection			No Preprocessing			Instance Selection			Instance Selection			No Preprocessing			Instance Selection		
Event Log	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness
BPIC-2012	20 X 10 X 18	10	null	42 X 16 X 25	21	null	29 X 12 X 25	13	null	257 X 23 X 20	112	76948	428 X 28 X 27	134	50490	567 X 31 X 27	189	125878
BPIC-2013	2 X 2 X 5	1	null	2 X 2 X 6	1	null	2 X 2 X 6	1	null	35 X 6 X 7	13	866	44 X 10 X 8	22	1080	49 X 6 X 8	15	1393
BPIC-2017-All	22 X 9 X 23	10	null	48 X 14 X 28	21	null	41 X 12 X 27	14	null	416 X 33 X 25	200	88513	716 X 44 X 30	290	96450	823 X 44 X 29	363	127961
BPIC-2017-Offer	9 X 5 X 6	4	null	34 X 12 X 10	12	3900	23 X 8 X 10	8	null	18 X 7 X 8	9	117	58 X 13 X 12	29	2040	82 X 11 X 12	31	4217
BPIC-2018-Control	7 X 4 X 6	3	null	14 X 8 X 9	7	null	8 X 5 X 8	4	null	24 X 7 X 8	12	434	52 X 13 X 11	26	3190	40 X 9 X 10	18	1128
BPIC-2018-Inspection	15 X 7 X 13	7	null	29 X 11 X 17	12	null	20 X 8 X 17	9	null	134 X 16 X 15	64	27353	324 X 26 X 19	142	53200	356 X 27 X 19	151	68179
BPIC-2018-Reference	8 X 5 X 6	4	0	12 X 7 X 8	6	null	9 X 6 X 8	5	null	22 X 8 X 8	11	336	36 X 11 X 10	18	800	33 X 10 X 10	15	921
BPIC-2019	108 X 25 X 24	39	887	153 X 32 X 44	42	null	102 X 26 X 41	37	null	593 X 31 X 26	224	237240	1550 X 61 X 46	561	332120	1438 X 56 X 43	578	458461
Hospital	20 X 8 X 12	9	929	4 X 3 X 20	2	null	8 X 4 X 19	4	null	158 X 14 X 14	54	13696	440 X 29 X 22	120	42900	401 X 24 X 21	116	37589
Road	22 X 10 X 9	10	171	39 X 18 X 13	18	null	51 X 20 X 13	22	null	45 X 11 X 11	22	1892	150 X 16 X 15	67	9000	121 X 16 X 15	52	12263
Sepsis	33 X 13 X 12	15	945	66 X 19 X 18	22	null	66 X 19 X 18	22	null	121 X 16 X 14	49	12811	470 X 31 X 20	209	42000	470 X 31 X 20	209	42000
IMi										Split Miner								
Preprocessing	Instance Selection			No Preprocessing			Instance Selection			Instance Selection			No Preprocessing			Instance Selection		
Event Log	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness	Size	Cardoso	Structuredness
BPIC-2012	71 X 27 X 32	32	21068	98 X 26 X 47	34	274	109 X 34 X 50	41	411	150 X 40 X 75	75	1113.1	352 X 69 X 176	176	4576	331 X 67 X 166	166	3294.5
BPIC-2013	14 X 7 X 7	7	32	30 X 13 X 13	15	270	32 X 13 X 14	16	533	40 X 14 X 20	20	48.5	54 X 18 X 27	26	162	61 X 20 X 30	29.6	91.25
BPIC-2017-All	75 X 24 X 36	33	3032	102 X 30 X 49	39	8956	102 X 31 X 49	41	4546	188 X 50 X 94	93	986.92	460 X 82 X 230	230	3680	413 X 81 X 207	207	2793.3
BPIC-2017-Offer	13 X 7 X 6	6	9	22 X 8 X 11	7	15	22 X 8 X 11	7	15	21 X 9 X 11	10	24.146	46 X 13 X 23	22	92	47 X 14 X 24	22.6	87.25
BPIC-2018-Control	15 X 7 X 7	7	19	34 X 13 X 16	14	166	28 X 11 X 13	12	102	24 X 10 X 12	12	30.292	46 X 16 X 23	22	92	45 X 16 X 22	21.9	82
BPIC-2018-Inspection	48 X 20 X 22	23	673	92 X 34 X 40	41	924	85 X 30 X 38	38	1404	85 X 26 X 43	42	228.42	174 X 41 X 87	87	870	179 X 43 X 89	89.4	709.25
BPIC-2018-Reference	18 X 8 X 9	8	30	28 X 11 X 13	12	162	27 X 10 X 13	12	33	24 X 10 X 12	12	30.708	60 X 20 X 30	30	180	43 X 15 X 22	21.1	106.5
BPIC-2019	81 X 24 X 38	29	2327	114 X 21 X 56	27	2936	115 X 26 X 55	34	1775	315 X 58 X 158	155	5027.9	1128 X 147 X 564	562	37290	792 X 120 X 396	392	17260
Hospital	36 X 12 X 18	14	116	76 X 22 X 36	24	350	73 X 22 X 35	27	293	81 X 24 X 41	40	446.75	400 X 77 X 200	199	5600	273 X 62 X 136	136	2141.5
Road	27 X 11 X 13	11	51	62 X 23 X 26	24	193	53 X 18 X 25	21	179	35 X 12 X 17	17	75.771	146 X 30 X 73	73	584	101 X 27 X 50	50.4	278
Sepsis	44 X 17 X 20	18	918	82 X 26 X 37	28	199	82 X 26 X 37	28	199	85 X 24 X 43	42	432.29	282 X 49 X 141	141	3666	280 X 48 X 140	140	3362.5

age values. In this table, *size* represents the number of elements in the discovered Petri net as $|A| \times |P| \times |T|$. Results show that using instance selection methods we will decrease the size of elements in discovered process models. Moreover, according to Cardoso values, the number of the complex component will be reduced when we use both preprocessing methods. However, using the instance selection method, the decrement is higher. In this table, there are some null values in the Structuredness field that indicate the corresponding process models are unsound. Therefore, as it is expected, the Alpha miner is not able to guarantee the soundness of discovered process models. Results illustrate that using instance selection strategies, for most of the cases, we are able to decrease the structuredness value or have simpler structures. In general, the reduction in complexity of discovered process models is higher for the Split miner that discovers more complex process models.

The average Cardoso values of different instance selection strategies are presented in Figure 13. It is shown that for most of the cases, all the instance selection strategies are having lower Cardoso values compared to the statistical sampling method. Among the instance selection methods, there is no method that dominantly outperforms others, however, the shorter strategy performs better than others. Except for the Alpha miner, the longer strategy usually results in process models with higher Cardoso value.

6. Conclusion

In this paper, we proposed some instance selection strategies to increase the performance of process discovery algorithms. We recommend applying process discovery algorithms on sampled event logs instead of whole event logs when dealing with large data sets. We

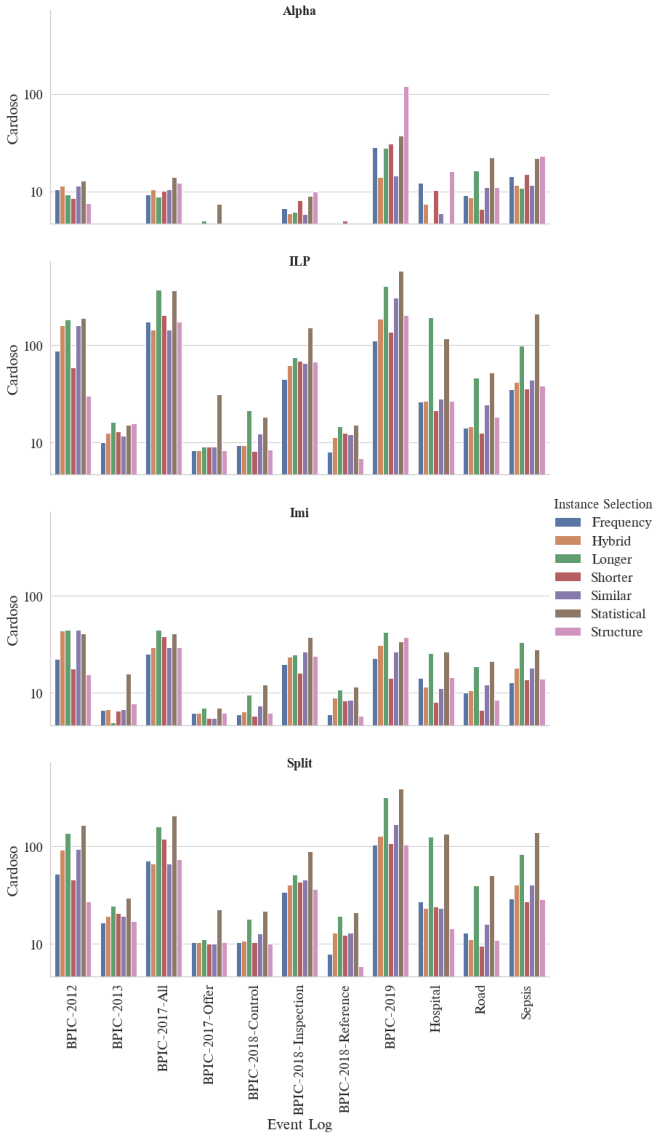

Fig. 13. The average Cardoso values of discovered process models using different preprocessing methods.

implemented different instance selection strategies in the ProM platform and also ported them into RapidProM.

To evaluate the proposed method, we applied it to several real publicly available event logs with different state-of-the-art process discovery algorithms. Experimental results show that by instance selection strategies, we are able to decrease the required time used by the process discovery algorithms. We found that instance selection strategies mostly increase the performance of process discovery by reducing the amount of behavior and the number of traces. Therefore, by using the sampled event logs, we are able to discover an acceptable approximation of the final process model in a shorter time. Using the sampled event log also results in having simpler process models according to complexity metrics. Moreover, the experiments indicate that for some event logs, instances selection strategies improve the quality of discovered process models according to the F-Measure metric for most of process discovery algorithms. The performance and F-Measure improvements for split miner are less than other process discovery algorithms. However, the instance selection strategies improve the complexity of this algorithm more than other methods. We found that the frequency, similarity and structured-based selection strategies result in process models with higher quality.

As future work, we aim to find out for different event logs and process discovery algorithms, which sampling strategy returns the best process model in less time. It is valuable to analyze the monotonicity of instance selection strategies and if possible guarantee specific requirements such as fitness or F-measure with the minimums size of the sampled event data.

Acknowledgments. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer Berlin Heidelberg (2016)
2. van der Aalst, W.M.P., et all: Process mining manifesto. In: Business Process Management BPM Workshops, Clermont-Ferrand, France. pp. 169–194 (2011)
3. van der Aalst, W.M.P., Bolt, A., van Zelst, S.: RapidProM: Mine Your Processes and Not Just Your Data. CoRR abs/1703.03740 (2017)
4. van der Aalst, W.M.P., van Dongen, B., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: ProM: The Process Mining Toolkit. BPM (Demos) 489(31) (2009)
5. van der Aalst, W.M., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004), https://doi.org/10.1109/TKDE.2004.47
6. Andrews, R., Suriadi, S., Ouyang, C., Poppe, E.: Towards event log querying for data quality - let's start with detecting log imperfections 11229, 116–134 (2018), https://doi.org/10.1007/978-3-030-02610-3_7
7. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowl. Inf. Syst. 59(2), 251–284 (2019), https://doi.org/10.1007/s10115-018-1214-x
8. Bauer, M., Senderovich, A., Gal, A., Grunske, L., Weidlich, M.: How much event data is enough? A statistical framework for process discovery. In: Krogstie, J., Reijers, H.A. (eds.) Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018,

Tallinn, Estonia, June 11-15, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10816, pp. 239–256. Springer (2018), https://doi.org/10.1007/978-3-319-91563-0_15
9. Berti, A.: Statistical sampling in process mining discovery. In: The 9th International Conference on Information, Process, and Knowledge Management. pp. 41–43 (2017)
10. Boltenhagen, M., Chatain, T., Carmona, J.: Generalized alignment-based trace clustering of process behavior. In: Donatelli, S., Haar, S. (eds.) Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11522, pp. 237–257. Springer (2019), https://doi.org/10.1007/978-3-030-21571-2_14
11. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman, R., Panetto, H., Dillon, T.S., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I. Lecture Notes in Computer Science, vol. 7565, pp. 305–322. Springer (2012), https://doi.org/10.1007/978-3-642-33606-5_19
12. Carmona, J., Cortadella, J.: Process mining meets abstract interpretation. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I. Lecture Notes in Computer Science, vol. 6321, pp. 184–199. Springer (2010), https://doi.org/10.1007/978-3-642-15880-3_18
13. Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Trans. Knowl. Data Eng. 29(2), 300–314 (2017), https://doi.org/10.1109/TKDE.2016.2614680
14. De Leoni, M., Mannhardt, F.: Road traffic fine management process (2015), https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5
15. van Dongen, B., Borchert, F.: BPIC 2018. Eindhoven University of Technology (2018), https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972
16. van Dongen, B.F., van der Aalst, W.M.P.: A meta model for process mining data 160 (2005), http://ceur-ws.org/Vol-160/paper11.pdf
17. Fani Sani, M., Berti, A., van Zelst, S.J., van der Aalst, W.M.P.: Filtering toolkit: Interactively filter event logs to improve the quality of discovered models 2420, 134–138 (2019), http://ceur-ws.org/Vol-2420/paperDT4.pdf
18. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Teniente, E., Weidlich, M. (eds.) Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers. Lecture Notes in Business Information Processing, vol. 308, pp. 216–229. Springer (2017), https://doi.org/10.1007/978-3-319-74030-0_16
19. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Applying sequence mining for outlier detection in process mining. In: Panetto, H., Debruyne, C., Proper, H.A., Ardagna, C.A., Roman, D., Meersman, R. (eds.) 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. Lecture Notes in Computer Science, vol. 11230, pp. 98–116. Springer (2018), https://doi.org/10.1007/978-3-030-02671-4_6
20. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: 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. Lecture Notes in Business Information Processing, vol. 320, pp. 115–131. Springer (2018), https://doi.org/10.1007/978-3-319-93931-5_9
21. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Repairing outlier behaviour in event logs using contextual behaviour. vol. 14, pp. 5:1–5:24 (2018), https://doi.org/10.18417/emisa.14.5

22. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Conformance checking approximation using subset selection and edit distance. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12127, pp. 234–251. Springer (2020), https://doi.org/10.1007/978-3-030-49435-3_15
23. Fernández-Cerero, D., Varela-Vaca, Á.J., Fernández-Montes, A., López, M.T.G., Alvárez-Bermejo, J.A.: Measuring data-centre workflows complexity through process mining: the google cluster case. J. Supercomput. 76(4), 2449–2478 (2020), https://doi.org/10.1007/s11227-019-02996-2
24. Gao, J., van Zelst, S.J., Lu, X., van der Aalst, W.M.P.: Automated robotic process automation: A self-learning approach. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11877, pp. 95–112. Springer (2019), https://doi.org/10.1007/978-3-030-33246-4_6
25. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4714, pp. 328–343. Springer (2007), https://doi.org/10.1007/978-3-540-75183-0_24
26. Knols, B., van der Werf, J.M.E.M.: Measuring the behavioral quality of log sampling. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019. pp. 97–104. IEEE (2019), https://doi.org/10.1109/ICPM.2019.00024
27. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for workflow nets. Inf. Softw. Technol. 51(3), 610–626 (2009), https://doi.org/10.1016/j.infsof.2008.08.005
28. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - A constructive approach. In: Colom, J.M., Desel, J. (eds.) Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings, Lecture Notes in Computer Science, vol. 7927, pp. 311–329. Springer (2013), https://doi.org/10.1007/978-3-642-38697-8_17
29. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: 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, Lecture Notes in Business Information Processing, vol. 171, pp. 66–78. Springer (2013), https://doi.org/10.1007/978-3-319-06257-0_6
30. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. 56, 235–257 (2016), https://doi.org/10.1016/j.is.2015.07.003
31. Luengo, D., Sepúlveda, M.: Applying clustering in process mining to find different versions of a business process that changes over time. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I. Lecture Notes in Business Information Processing, vol. 99, pp. 153–158. Springer (2011), https://doi.org/10.1007/978-3-642-28108-2_15
32. Mannhardt, F.: Hospital billing-event log. Eindhoven University of Technology. Dataset pp. 326–347 (2017)
33. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Data-driven process discovery - revealing conditional infrequent behavior from event logs. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10253, pp. 545–560. Springer (2017), https://doi.org/10.1007/978-3-319-59536-8_34

34. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Lecture Notes in Business Information Processing, vol. 6. Springer (2008), https://doi.org/10.1007/978-3-540-89224-3
35. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
36. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.P.: Discovering process models from uncertain event data. In: Francescomarino, C.D., Dijkman, R.M., Zdun, U. (eds.) Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 362, pp. 238–249. Springer (2019), https://doi.org/10.1007/978-3-030-37453-2_20
37. Pérez-Castillo, R., Fernández-Ropero, M., Piattini, M.: Business process model refactoring applying IBUPROFEN. an industrial evaluation. J. Syst. Softw. 147, 86–103 (2019), https://doi.org/10.1016/j.jss.2018.10.012
38. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Scenario-based prediction of business processes using system dynamics. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11877, pp. 422–439. Springer (2019), https://doi.org/10.1007/978-3-030-33246-4_27
39. Qafari, M.S., van der Aalst, W.M.P.: Fairness-aware process mining. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11877, pp. 182–192. Springer (2019), https://doi.org/10.1007/978-3-030-33246-4_11
40. Rafiei, M., van der Aalst, W.M.P.: Mining roles from event logs while preserving privacy. In: Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers. pp. 676–689 (2019), https://doi.org/10.1007/978-3-030-37453-2_54
41. Sani, M.F., van Zelst, S.J., van der Aalst, W.M.P.: The impact of event log subset selection on the performance of process discovery algorithms. In: Welzer, T., Eder, J., Podgorelec, V., Wrembel, R., Ivanovic, M., Gamper, J., Morzy, M., Tzouramanis, T., Darmont, J., Latific, A.K. (eds.) 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. Communications in Computer and Information Science, vol. 1064, pp. 391–404. Springer (2019), https://doi.org/10.1007/978-3-030-30278-8_39
42. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops, BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers. Lecture Notes in Business Information Processing, vol. 17, pp. 109–120. Springer (2008), https://doi.org/10.1007/978-3-642-00328-8_11
43. van Zelst, S., van Dongen, B., van der Aalst, W.M.P., Verbeek, H.M.W.: Discovering Workflow Nets Using Integer Linear Programming. Computing (Nov 2017), https://doi.org/10.1007/s00607-017-0582-5
44. Suriadi, S., Andrews, R., ter Hofstede, A.H.M., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Inf. Syst. 64, 132–150 (2017), https://doi.org/10.1016/j.is.2016.07.011
45. van Dongen, B.F.: BPIC 2012. Eindhoven University of Technology (2012), https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
46. van Dongen, B.F.: BPIC 2017. Eindhoven University of Technology (2017), https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

47. van Dongen, B.F.: BPIC 2019. Eindhoven University of Technology (2019), https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1
48. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: 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. Lecture Notes in Business Information Processing, vol. 72, pp. 60–75. Springer (2010), https://doi.org/10.1007/978-3-642-17722-4_5
49. Ward Steeman: BPIC 2013. Eindhoven University of Technology (2013), https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07
50. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: 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. pp. 148–155. IEEE (2011), https://doi.org/10.1109/CIDM.2011.5949428
51. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. 25(12), 2708–2720 (2013), https://doi.org/10.1109/TKDE.2013.64
52. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). 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. pp. 310–317. IEEE (2011), https://doi.org/10.1109/CIDM.2011.5949453
53. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. Fundam. Inform. 94(3-4), 387–412 (2009), https://doi.org/10.3233/FI-2009-136
54. van Zelst, S.J., van Dongen, B.F., van der Aalst, W.M.P.: Avoiding over-fitting in ilp-based process discovery. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings, Lecture Notes in Computer Science, vol. 9253, pp. 163–171. Springer (2015), https://doi.org/10.1007/978-3-319-23063-4_10
55. van Zelst, S.J., Fani Sani, M., Ostovar, A., Conforti, R., Rosa, M.L.: Detection and removal of infrequent behavior from event streams of business processes. Inf. Syst. 90, 101451 (2020), https://doi.org/10.1016/j.is.2019.101451