The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms

Chapter · September 2019

DOI: 10.1007/978-3-030-30278-8_39

3 authors:

Mohammadreza Fani Sani
RWTH Aachen University
20 PUBLICATIONS **134** CITATIONS
SEE PROFILE

Sebastiaan van Zelst
Fraunhofer Institute for Applied Information Technology FIT
52 PUBLICATIONS **293** CITATIONS
SEE PROFILE

Wil Van der Aalst
RWTH Aachen University
1,274 PUBLICATIONS **73,073** CITATIONS
SEE PROFILE

The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms

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

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

Summary. Process discovery algorithms automatically discover process models on the basis of event data, 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 speed up discovery considerably using ranking-based strategies. Furthermore, results show that biased selection of the process instances compared to random selection of them will result in process models with higher quality.

Key words: Process Mining · Process Discovery· Subset Selection · Event Log Preprocessing · Performance Enhancement

1 Introduction

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]. Currently, the main research focus in 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]. Some process discovery algorithms are impractical in big data settings. 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 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, 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 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 [3] extension of `RapidMiner` [4], 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.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. Section 3 defines preliminary notation. We present different biased sampling strategies in Section 4. The 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 such as the Alpha Miner [5], the ILP Miner [6, 7], and the Inductive Miner [8] first create an abstraction of the event data, e.g., the directly follows graph, and in a second step discover a process model based on it. The performance of all these algorithms depends on different factors such as the number of process instances and the unique number of activities.

Recently, preprocessing of event data has gained attention. In [9, 10], techniques are proposed to increase the quality of discovered process models by cleansing the event data. Also, in [11] and [12] we have shown that by removing/modifying outlier behavior in event logs, process discovery algorithms are able to discover process models with higher quality. Moreover, [13] uses data attributes to filter out noisy behavior. Filtering techniques effectively reduce the size of the event data used by process discovery algorithms. However, sometimes the required time for applying these filtering algorithms is longer than the process discovery time. Also, these techniques have no accurate control on 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 algorithms. Some sampling approaches have been proposed in the field of process mining. In [14], 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 different types of process behavior. Furthermore, [15] recommends a trace-based sampling method specifically for the Heuristic miner [16]. In both of these sampling methods, there is 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. In this paper, we analyze random and biased subset selection methods with which we are able adjust the size of the sampled event data.

3 Preliminaries

In this section, we briefly introduce basic process mining terminology and notations that ease the readability of this paper. Given a set X, a multiset M over X is a function $M\colon X \to \mathbb{N}_{\geq 0}$, i.e., it allows certain elements of X to appear multiple times. $\overline{M} = \{e \in X \mid M(e) > 0\}$ 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 $\sigma\colon \{1, 2, ..., n\} \to X$, 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 concatenation of sequences σ and σ' is written as $\sigma \cdot \sigma'$. We define the frequency of occurrence of σ' in σ by $freq\colon X^* \times X^* \to \mathbb{N}_{\geq 0}$ where $freq(\sigma', \sigma) = |\{1 \leq i \leq |\sigma| - |\sigma'| \mid \sigma'_1 = \sigma_i, ..., \sigma'_{|\sigma'|} = \sigma_{i+|\sigma'|}\}|$. For example, $freq(\langle b\rangle, \langle a, b, b, c, d, e, f, h\rangle) = 2$ and $freq(\langle b, d\rangle, \langle a, b, d, c, e, g\rangle) = 1$.

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 an *event*. A sequence of events for a specific case is referred 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. Let $\mathcal{A}$ be a set of activities. An event log is a multiset of sequences over $\mathcal{A}$, i.e., $L \in \mathcal{M}(\mathcal{A}^*)$. Moreover, we let each $\sigma \in \overline{L}$ describe a *trace-variant* whereas $L(\sigma)$ denote how many traces of the form σ are presented within the event log. S_L is a subset of event log L, if for any $\sigma \in S_L$, $S_L(\sigma) \leq L(\sigma)$. We call S_L as a sampled event log of L.

Different types of behavior abstractions in an event log could be defined. One abstraction is the directly follows relation between activities that can be defined as follows.

Definition 1 (Directly Follows Relation). *Let a and $b \in \mathcal{A}$ be two activities and $\sigma = \langle x_1, .., x_n\rangle$ a trace in the event log. A directly follows (DF) relation from a to b exists in σ, if there is $i \in \{1, .., n-1\}$ such that $x_i = a$ and $x_{i+1} = b$ and is denoted by $a >_\sigma b$.*

We can map an event log to a directed graph whose vertices are activities and edges are directly follows relations and we call it directly follows graph (DFG). So, if there is a $a >_\sigma b$ in the event log, there is also a directed edge from *a* to *b* in the corresponding DFG of this event log.

In [12], it shows that 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 has negative effects on the results of process discovery algorithms.

Definition 2 (Behavioral Context). *We define the set of behavioral contexts present in event log L according to subsequence σ', i.e., $\beta_L \in \mathcal{P}(\mathcal{A}^* \times \mathcal{A}^*)$, as follows:*

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

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. We inspect the probability of contextual sub-patterns, i.e., behavior that is surrounded by the frequent behavioral contexts and denoted by σ' in Equation 1. Thus, we simply compute the empirical conditional probability of a behavioral sequence, surrounded by a certain context.

Definition 3 (Conditional Contextual Probability). *We define the* conditional contextual probability *of σ_s, w.r.t., σ_l and σ_r in event log 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 $\gamma_L : \mathcal{A}^* \times \mathcal{A}^* \times \mathcal{A}^* \rightarrow [0, 1]$, is defined as:*

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

On the basis of these probabilities, we are able to detect unstructured behavior in traces.

4 Subset Selection

In this section, we present different subset selection strategies to improve the discovery procedure's performance. Different behavioral elements of an event log, e.g., events, directly follow relations, traces, and variants can be used for sampling. However, not all of them are useful for the purpose of process discovery. By selecting events, it is possible to consider events from different parts of a process instance that results in imperfect traces that are harmful for all process discovery algorithms. Selecting DF relations is useful for some process discovery algorithms like the Alpha Miner. But, they are insufficient for other process discovery algorithms. Thus, here we make subset of event logs only based on traces and variants. Consequently, these subset selection methods take an event log as an input and return a sampled event log. The schematic of subset selection methods is illustrated in Figure 1.

Note that in *XES* standard [3], variants are not stored in event logs separately. However, there are other structures that we can keep variants and their frequencies as metadata [17] that are more efficient for process discovery algorithm. Here, we consider *XES* standard; however, in sampled event logs, we keep only one trace for each selected variant. Consequently, the frequency of each trace in the sampled event log equals to 1.

In many process discovery algorithms such as the ILP Miner, the family of Alpha miners and the basic Inductive Miner, the frequencies of traces variants (i.e., $L(\sigma)$) have no important effects on discovered process models. Therefore, here we mainly focus on selecting variants; but, all these methods can easily be extended to trace-based subset selection methods. We also used just control-flow related information that is available in all event logs and this is consistent with the way.

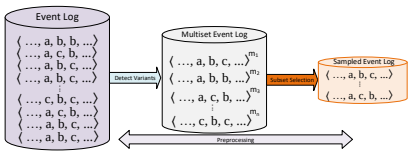

Fig. 1: Schematic overview of event log subset selection

One of the most important characteristics of a sampled event log is the number of its traces, i.e., $|S_L|$. When it is the same as the original event log, there is no reduction in the size. We can set the size of the sampled event log (i.e., the sampling threshold) as $c = \frac{|S_L|}{|\overline{L}|}$ that $0 < c \leq 1$[1]. Moreover, the less required subset selection time is more desirable as it is considered as a preprocessing phase.

We can select traces in an event log randomly or based on some strategies. In the following, we will explain both of these methods.

4.1 Random Sampling

In this method, we randomly select $c \times |L|$ traces in the event log without replacement and return these traces or just unique the trace-variants among them. This method is fast because we do not need to traverse the original event log. However, it is possible that many of sampled traces have similar behavior and we keep just a few unique variants in the sampled event log. The statistical sampling method [14] works based on this approach.

As an alternative method, we can first find all the unique variants in an event log; afterward, randomly select $c \times |\overline{L}|$ variants from them. This approach is a bit slower; however, it is able to return much behavior compared to the previous approach.

4.2 Biased Sampling Strategies

In general, traversing a big event log is not too time-consuming compared to the time of process discovery. Therefore, as it shown in Figure 1, instead of randomly selecting the variants, we are able to first find all variants in an event log and use more advanced strategies (biases) to select them. In this type of approaches, we first rank all variants of an event log based on different strategies. Afterward, we return the top $c \times |\overline{L}|$ variants with the highest rank in the sampled event log. We are able to use different ranking strategies that will be discussed as follows. These ranking strategies have different preprocessing time and result in different sampled event logs.

As we select variants, the frequency of behavior in the sampled event log will be unusable. To consider these frequencies, we can benefit from other event log standards that are able to keep frequencies of variants like [17] or instead of returning $c \times |\overline{L}|$ variants, we should return $c \times |L|$ traces that correspond to these high ranked variants.

Frequency-based Selection: The first ranking strategy is selecting variants based on their frequencies in the original event log. This ranking strategy gives higher priority to a variant that has a higher occurrence frequency in the event log. So, 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. The trace-based version of this strategy is already presented in many process mining tools that helps users to keep the top most frequent behavior of the event log. However, in some event logs, the majority of process instances have a unique trace-variant which makes this subset selection method unusable.

[1] Here, we select only one trace for each variant.

Length-based Selection: We are able to rank variants based their length (i.e., $|\sigma|$). So, in this strategy, we sort variants based on their length and choose the longest or the shortest ones first. By using the *longer* strategy, we keep much behavior in our sampled event log and at the same time leave out many of incomplete traces, that may improve 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 choose many infrequent variants with the same behavior for process discovery algorithms. On the other hand, by applying *shorter* strategy, there will be less behavior in the sampled event log; but, it is possible to keep many incomplete traces that leads to an unsuitable process model.

Similarity-based Sampling: If we are interested in retaining the main-stream behaviors of the event log, we need to rank variants based on the similarity of them to each other. In this approach, we first find common behavior of the event log. For this purpose, we can use different types of behavior; however, the simplest and the most acceptable type of behavior for process discovery is the DF relation. Thus, we compute the occurrence probability of each directly follows relation (a, b) (that $a, b \in \mathcal{A}$) according to the following equation:

$$Prob(a, b) = \frac{|\sigma \in \overline{L} | a >_\sigma b|}{|\overline{L}|}. \tag{3}$$

So, we compute the occurrence probability of all of the DF relations in a variant. If $Prob(a, b)$ is high enough (i.e., be higher than a defined threshold T_P), we expect that sampled variants should contain it. So, any variant that contains such a high probable behavior, will give a $+1$ to its rank. Otherwise, if a variant does not contain a probable behavior, we decrease its rank by -1. Contrariwise, if a variant contains a low probable behavior (i.e., $Prob(a, b) \leq 1 - T_P$), we decrease its rank by -1. To normalize the ranking values, we divide them by the variant length. By using this ranking strategy, we are looking for variants with much of high probable behavior and less of low probable ones. Note that it is possible that some DF relations be neither high probable nor low probable that we do not consider them in the ranking procedure. Finally, we sort the variants based on their ranks and return the $c \times |\overline{L}|$ ones with the highest rank.

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 the similarity score of all variants. Especially, if we use more advanced behavioral data structures such as eventually follows instead of DF relations, this computation can be a limitation for this ranking strategy.

Structure-based Selection: In this subset selection method, we consider the presence of unstructured behavior (i.e., based on Definition 3) in each variant. In this regard, we first compute the occurrence probability of each sub-patten σ' among its surrounding contextual contexts $\beta_L(\sigma')$ (i.e., $\gamma_L(\sigma_s, \sigma_l, \sigma_r)$). If this probability is below the given threshold, i.e., T_S, we consider it as unstructured behavior. We expect that unstructured subsequences have problematic effects on process discovery algorithms and make discovered process models inaccurate and complex [12]. Thus, for each unstructured behavior in a variant, we give a penalty to it and decrease its rank by -1. Consequently, a variant with unstructured behavior receives more penalties and it is not appealing to be placed in the sampled event log.

The main reason to use this ranking strategy is that it results in the main skeleton of the process models. It is designed to reduce improbable parallel behavior and having simpler process models. However, this subset selection strategy requires longer time to rank variants in event logs.

5 Evaluation

In this section, we aim to find out the effects of subset selection methods on the performance of process discovery algorithms. Moreover, we will analyze the quality of process models that are discovered via sampled event logs.

To apply the proposed subset selection methods, we implemented the *Sample Variant* plug-in in `ProM` framework[2]. In this implementation, we used static thresholds for both similarity and structure-based ranking strategies. Using this plug-in, the end user is able to specify her desired percentage of sampling variants/traces and the ranking strategy. It takes an event log as an input and returns the top fraction of its variants/traces. In addition, 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` which extends `RapidMiner` with process analysis capabilities. In our experiment, we also used the statistical sampling method [14]; however, as we consider only work-flow information, its relaxation parameter is ignored.

Information about real event logs that are used in the evaluation is given in Table 1[3]. To differentiate between the activities that are occurred at the starting and ending parts of traces, we added artificial *Start* and *End* activities to all of the traces. For process discovery, we used the Inductive Miner [18], the ILP Miner [7], and the Split Miner [19]. On the sampled event logs, we applied process discovery algorithms just without their built-in filtering mechanisms.

Table 1: Details of real event logs

Event Log	Activities#	Traces#	Variants#	DF#
BPIC-2012	26	13087	4336	138
BPIC-2017	28	31509	1593	178
BPIC-2019	44	251734	11973	538
Hospital	20	100000	1020	143
Road	13	150370	231	70
Sepsis	18	1050	846	115

We investigate the probable improvement of subset selection methods on the performance of process discovery algorithms. To measure this improvement, we apply the following equation:

$$DiscoveryTimeImprovement = \frac{DiscoveryTime(WholeLog)}{DiscoveryTime(SampledLog) + SamplingTime(WholeLog)}. \quad (4)$$

Figure 2 shows improvements in the performance of process discovery algorithms when we select subset of event logs. Here, for ranking-based strategies, we used sampling threshold (i.e., c) equals to 0.1. Each experiment was repeated four times (because the discovery and sampling times are not deterministic) and the average values are shown. For some event logs, the improvement is more than 100 by sampling event logs. It is shown that the improvement is significantly higher when the statistical sampling

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

[3] `https://data.4tu.nl/repository/collection:event_logs_real`

method is used because it does not need to traverse the input event log. However, for *Sepsis* that has few traces, ranking based subset selection methods are faster. Note that the structure-based strategy sometimes has no improvement because it requires higher sampling time. The sampling time of different methods are depicted in Figure 4. As we expected, the statistical sampling method is much faster than other subset selection methods and the structure-based is the slowest one.

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 subset selection, it is possible that some of infrequent activities are not placed in the sampled event log. Moreover, by subset selection we reduce the size of event logs (i.e., $|S_L| \leq |L|$) and also possible behavior in the event log. Figure 3 shows the process discovery time of sampled event logs with different sampling thresholds when we used the ILP miner. When we used the sampling threshold equals to 0.99, we will have almost all behavior of the original event log in the sampled event log; however, the number of traces in the sampled event log is significantly lower than the original event log. Results show that for many event logs, the main reason of the improvement in performance of the process discovery is gained by reducing the number of variants. However, for *Road* and *Hospital* event logs that there are high frequent variants, reducing the number of traces has higher impact on the performance of the process discovery algorithms.

As explained, the amount of behavior in the event log has an important role on the performance of process discovery algorithms. The remained percentage of DF relations in the sampled event logs for different subset selection methods are given in Figure 5. We can see that for most of the event logs the similar and structure based methods keep fewer DF relations. However, according to their ranking policy, they keep the most common DF relations among the original event log.

In the previous experiment, the sampling threshold for ranking-based subset selection methods equals to 0.1. Note that there is no such control on 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, $L_1 = [\langle a, b, c\rangle, \langle a, c, b\rangle]$ and $L_2 = [\langle a, c, b, c\rangle, \langle a, b\rangle]$ have the same sets of DF relations; however, their process models are different. Figure 6 indicates the average percentage of remaining variants in sampled event logs using the statistical sampling method. It shows that, this method is able to just keep few percentage of variants of an event log. For ranking-based strategies, the number of remaining variants can be adjusted by c.

We also compared the average of preprocessing time for trace filtering methods and the similarity-based subset selection method. For this experiment we filter event logs with six different filtering settings and iterate the experiments for four times. Also, we used the similarity-based strategy with the threshold in $[0.01, 0.05, 0.1]$. It is clear that the subset selection method preprocessed the event logs faster. Also, with the trace filtering methods we do not have control over the size of the filtered event logs.

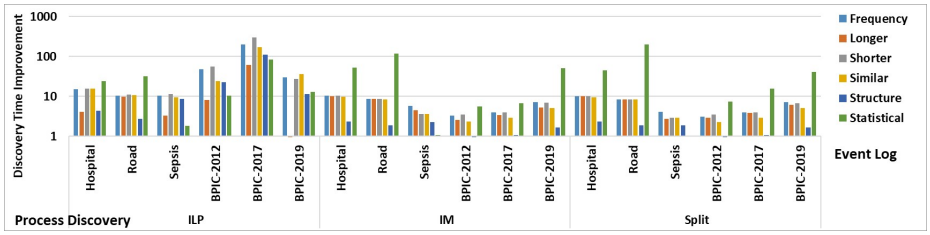

Fig. 2: Discovery time improvement for discovering process using subset selection.

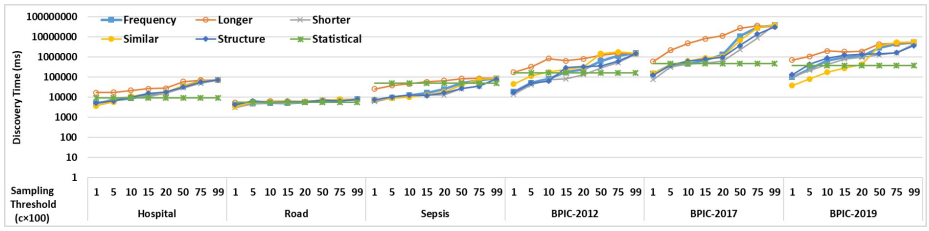

Fig. 3: The average of discovery time of the ILP miner for different subset selection methods and different sampling thresholds.

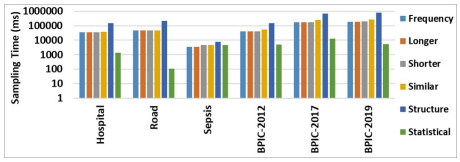

Fig. 4: Sampling time of different subset selection methods.

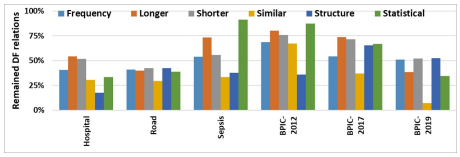

Fig. 5: Remained percentage of DF relations in the sampled event logs.

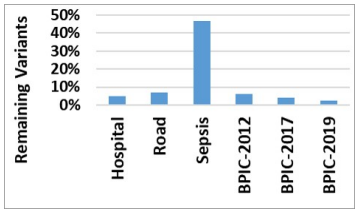

Fig. 6: Average of remained variants in sampled event logs using the statistical sampling [14].

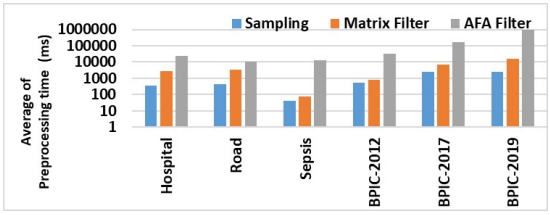

Fig. 7: The average of preprocessing time when we used similarity-based strategy and two state of the arts trace filtering methods [11, 20].

To analyze the quality of process models that are discovered from sampled event logs we can use *fitness* and *precision*. 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 presented in the event log. A low precision value means that the process model allows for much behavior compared to the event log. There is a trade-off between these measures [21], sometimes,

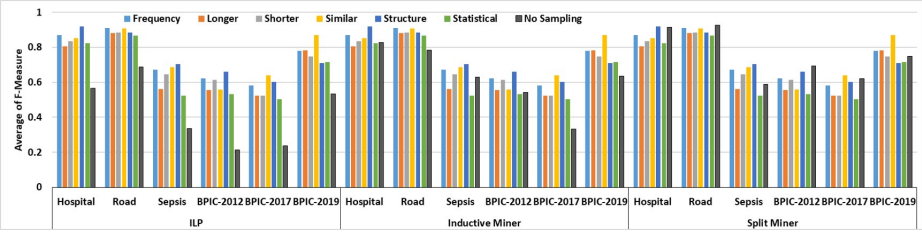

Fig. 8: Comparing the F-Measure of discovered process models with different subset selection methods. For ranking-based strategies, we used the sampling threshold in [0.01, 0.05, 0.1] and the average value of F-Measure is presented.

putting aside a small amount of behavior causes a slight decrease in the fitness value, whereas the precision value increases dramatically. Therefore, we use the F-Measure metric that combines both of them according to the following formula:

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Fitness}}{\text{Precision} + \text{Fitness}}. \tag{5}$$

Figure 8 compares the quality of best process models that are discovered with/without subset selection. 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 ranking based subset selection methods were used, we applied the sampling thresholds [0.01, 0.05, 0.1], 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. According to the results, for the ILP and the Inductive miner, we always have an improvement when we use subset selection. However, for some event logs, the Split miner can discover process models with higher quality via the original event logs. Moreover, the statistical sampling method that randomly selects process instances results in process models with less quality according to the F-Measure compared to ranking based subset selection methods. Among ranking strategies, the structure and similarity-based ones result in better process models for most of the event logs. For some event logs like *BPIC-2019*, the similarity-based ranking strategy is the best choice for all of the process discovery algorithms. For *Hospital* event log, the *structure*-based ranking strategy results in process models with the highest F-Measure. As there are frequent variants in *Road*, the best subset selection method for this event log is the *frequency* one. Results show that length-based methods are not suitable to sample event logs if the purpose is to have process models with high quality.

In this experiment, we just used the basic versions of the process discovery algorithms. However, in practice users can apply their embedded filtering and select the best discovered process model. In the next experiment, we applied the inductive miner and the Split miner with 100 different filtering thresholds and find best process models according to the F-Measure values. Figure 9 compares the best F-Measure value that discovered via ranking-based subset selection methods compared to the case that we used embedded filtering mechanisms of process discovery algorithms on original event data. As mentioned in subsection 4.2, by applying subset selection methods, we will lose the frequency of variants. As a result, some embedded filtering mechanisms

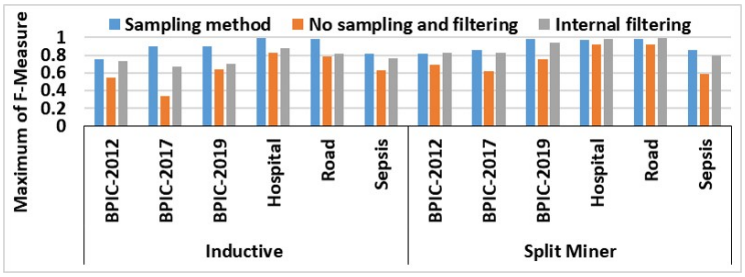

Fig. 9: The best F-Measure of discovered process models when using sampled event logs compared to applying embedded filtering mechanisms of discovery algorithms.

in process discovery algorithms become unusable. However, the results of this experiment show that we may discover process models with high quality from sampled event logs, even without these filtering methods. If for any reason we need to use the frequency of variants, it is recommended to apply trace-based subset selection methods or other standards to store event logs like [17]. Also, all of the proposed subset selection methods needs to load the original event log (like the statistical sampling) that is a limitation for dealing big event logs when memory is a constraint. Moreover, much of preprocessing time in ranking-based strategies consumed to traverse the event log and find possible variants. This information is already recorded in some event log standards like MXML [17]. Using these standards leads to decrease the preprocessing time of the proposed approaches.

Note that we did not use filtering mechanisms of the process discovery algorithm for sampled event logs. Results show that subset selection methods can increase the quality of discovered process models; specifically, if we use the Inductive Miner. It shows the weakness of process discovery algorithms in dealing with infrequent behavior [11].

6 Conclusion

In this paper, we proposed some subset selection strategies to increase the performance of process discovery procedure. We recommend to apply process discovery algorithms on sampled event logs when dealing with large data sets. We implemented different ranking strategies in ProM and ported this functionality into RapidProM and applied it on some real event logs using different process discovery algorithms.

Experimental results show that event log subset selection methods decrease the required time used by state-of-the-art process discovery algorithms. We found that ranking-based strategies mostly increase the performance of process discovery by reducing the amount of behavior and the number of traces. Therefore, by applying these methods, we are able to discover an acceptable approximation of the final process model in a shorter time. Moreover, results show that for some event logs, subset selection methods can improve the quality of discovered process models according to the F-Measure metric. Results show that to have higher F-Measure value, it is better to use the structure and similarity-based strategies. However, by using random trace sampling methods, e.g., the statistical method, we can discover process models in a shorter time.

As future work, we aim to find out what the best subset selection method is due to the available time and event data.

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. (2011) 169–194
3. Verbeek, H., Buijs, J.C., Van Dongen, B.F., Van Der Aalst, W.M.: Xes, xesame, and prom 6. In: CAISE, Springer (2010) 60–75
4. 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)
5. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow Mining: Discovering Process Models From Event Logs. IEEE Trans. Knowl. Data Eng. **16**(9) (2004) 1128–1142
6. van der Werf, J., van Dongen, B., Hurkens, C., Serebrenik, A.: Process Discovery using Integer Linear Programming. Fundam. Inform. **94**(3-4) (2009) 387–412
7. 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)
8. Leemans, S.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Application and Theory of Petri Nets and Concurrency. Springer Berlin Heidelberg (2013) 311–329
9. Suriadi, S., Andrews, R., ter Hofstede, A., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems **64** (2017) 132–150
10. Andrews, R., Suriadi, S., Ouyang, C., Poppe, E.: Towards Event Log Querying for Data Quality: Let's Start with Detecting Log Imperfections. (2018)
11. 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: Business Process Management BPM Workshops, Barcelona, Spain. (2017) 216–229
12. Fani Sani, M., van Zelst, S., van der Aalst, W.M.P.: Repairing Outlier Behaviour in Event Logs. In: Business Information Systems, Springer (2018) 115–131
13. 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
14. Bauer, M., Senderovich, A., Gal, A., Grunske, L., Weidlich, M.: How much event data is enough? a statistical framework for process discovery. In: International Conference on Advanced Information Systems Engineering, Springer (2018) 239–256
15. Berti, A.: Statistical sampling in process mining discovery. In: The 9th International Conference on Information, Process, and Knowledge Management. (2017) 41–43
16. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible Heuristics Miner (FHM). In: CIDM. (2011)
17. van Dongen, B.F., van der Aalst, W.M.P.: A meta model for process mining data.
18. Leemans, S.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In: BPI. (2014) 66–78
19. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: Automated Discovery of Accurate and Simple Business Process Models from Event Logs. Knowledge and Information Systems (2019) 1–34
20. Conforti, R., La Rosa, M., ter Hofstede, A.: Filtering Out Infrequent Behavior from Business Process Event Logs. IEEE Trans. Knowl. Data Eng. **29**(2) (2017) 300–314
21. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A Robust F-measure for Evaluating Discovered Process Models. In: Proceedings of the CIDM. (2011) 148–155