Representative Sampling in Process Mining: Two Novel Sampling Algorithms for Event Logs

Frederik Fonger[1][0009−0000−8445−8104], Niclas Nebelung[1][0009−0005−1945−9051], Arvid Lepsien[1][0000−0002−8105−382X], Milda Aleknonytė-Resch[1][0000−0003−0472−1262], and Agnes Koschmider[1,2][0000−0001−8206−7636]

[1] Department of Computer Science, Kiel University, Kiel, Germany `{ffo, ale, mar}@informatik.uni-kiel.de`
[2] Chair of Business Informatics and Process Analytics, University of Bayreuth, Bayreuth, Germany `agnes.koschmider@uni-bayreuth.de`

Abstract. Process mining allows the discovery of business processes from an event log. However, event logs are rapidly increasing in size and process mining algorithms struggle with the computational load when efficient processing is required. This calls for methods that decrease the event log size while still preserving the representativeness of the event log. This paper presents two new algorithms for sampling event logs. The first algorithm called `RemainderPlus` chooses traces from an event log above a threshold and subsequently selects traces with underrepresented Directly Follows Relations. The second sampling algorithm called `AllBehavior` selects samples that have a high intersection of Directly Follows Relations with the original event log. Usually, `AllBehavior` is complemented with `RemainderPlus` for a more accurate sample representation. They perform well for conformance checking and excel in certain scenarios for process discovery. Thus, both algorithms outperform existing sampling algorithms.

Keywords: event log · sampling · algorithm · process mining · process analysis

1 Introduction

Process mining is a form of process analytics with a focus on discovering business processes from an event log, measuring the conformance between the as-is and to-be process behavior and for predictive monitoring. However, the increased volume of event log data challenges computational resources of process discovery algorithms and conformance checking [1] [5]. Therefore, techniques which reduce the size of the event log to be processed while maintaining event log usefulness are necessary.

To overcome this challenge, a few sampling algorithms have been suggested [4, 13] aiming to find a representative subset of an event log. The naive approach to randomly select a subset of an event log is not an efficient solution, although some sampling algorithms exist following this approach [11,18]. These algorithms particularly struggle with smaller sample sizes. Rather, algorithms which choose a subset of an event log that still maintains utility in terms of the quality dimensions of process mining are required. In fact, this is still an open challenge of existing sampling algorithms, which warrants further research. Therefore, this paper suggest two new sampling algorithms for event logs called `RemainderPlus` and `AllBehavior`.

Fig. 1 shows how the `RemainderPlus` algorithm works. The algorithm initially extracts the trace variants of an event log and sorts them by occurrence. Then, traces are selected to be included in the sample by the following two steps. In the first step, the occurrence of each trace variant is counted and multiplied by the sample ratio to determine the trace frequency for the sample. In order not to exceed the sample size, the calculated occurrence of traces is rounded down. Afterwards, the traces are added to the sample according to the frequency of their occurrence. In the second step, a score is calculated to determine the number of underrepresented and overrepresented Directly Follows Pairs (DFPs). This means that the trace variants are ranked according to the frequency of their occurrence. The top-ranked variant is added to the sample and the ranking is reshuffled. This is repeated until the target sample size is reached. The `AllBehavior` algorithm aims to find an event log sample that has a high intersection of Directly Follows Relations with the original event log. Therefore, the `AllBehavior` algorithm ranks the traces by its unsampled DFPs and adds them to the sample. If there are no unsampled Directly Follows Relations left, the `AllBehavior` algorithm terminates and the `RemainderPlus` algorithms is used to complete the sample.

We evaluated both sampling algorithms on seven event logs (e.g., BPIC 2012 [6] and the Sepsis dataset [14], etc.). We also compared the evaluation results with existing approaches. The evaluation results show superiority in terms of quality measures on the sample level, like mean absolute error and coverage. Furthermore, both sampling algorithms perform well for conformance checking and process discovery, as the results closely match those derived from the original log.

The remainder of the paper is structured as follows. Section 2 summarizes the terms and notations to which we refer throughout the paper. Section 3 presents the sampling algorithms `RemainderPlus` and `AllBehavior`. The evaluation results are described and discussed in Section 4. Related works are summarized in Section 5. The paper ends in Section 6 with an outlook and future research directions.

2 Preliminaries

This section first introduces the basic notations to which we will refer throughout the paper. Subsequently, quality measures for evaluating event log samples are summarized.

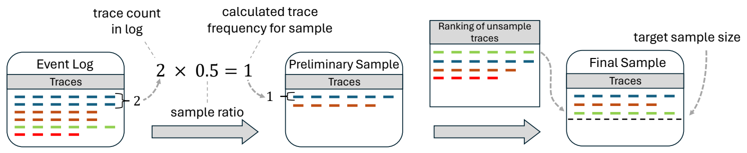

Fig. 1. A simplified representation of the RemainderPlus sampling algorithm applied on an event log.

2.1 Definitions

The basic notations are based on the definitions presented in [11]. Let X be a set. A *multiset* is a function $M : X \to \mathbb{N}_0$, where for $x \in X$, $M(x)$ denotes the number of occurrences of x in M. Multisets can also be denoted with square brackets, e.g., $M = [e_1^{k_1}, e_2^{k_2}, \ldots, e_n^{k_n}]$ where $k_i = M(e_i)$ for $1 \le i \le n$. $\mathbb{B}(X)$ is the set of all multisets over X. For any $x \in X$, inclusion is defined as $x \in M \Leftrightarrow M(x) > 0$. Another multiset S is a subset of M if $\forall_{x \in X} S(x) \le M(x)$. $|M| = \sum_{x \in X} M(x)$ is the cardinality of a multiset. The union of two multisets is defined as $\forall_{x \in X} (M \uplus S)(x) = M(x) + S(x)$. A (finite) *sequence* over some set X is a function $\sigma : \{1, \ldots, n\} \to X$ with $\sigma(i) = x_i$ for $1 \le i \le n$. The sequence has length $|\sigma| = n$, the set of all possible sequences over X is denoted as X^*, and two sequences $\sigma, \rho \in X^*$ are equivalent if $|\sigma| = |\rho| \ \wedge \ \forall_{1 \le i \le |\sigma|} \ \sigma(i) = \rho(i)$. The two sampling algorithms introduced in this paper are intended for control-flow aspects of business processes where additional attributes, e.g., the duration of executed activities are not considered. Next, an event log is defined as follows:

Definition 1 (Event log, sample). *The universe of activities is denoted as* $\mathcal{A}$. *Let* $A \subseteq \mathcal{A}$ *be a non-empty set of activities. Then, an event log is a multiset of sequences of activities* $L \in \mathbb{B}(A^*)$. *In an event log, a sequence of activities* $\sigma \in L$ *is called a trace. An occurrence of an activity in an event log is called an event. A subset* $S_L \subseteq L$ *is called a sample of* L*, with sample ratio* $r_L(S_L) = \frac{|S_L|}{|L|}$.

A *sampling algorithm* is any algorithm that takes a target sample ratio c and and event log L as input a produces a sample $S_L \subseteq L$. The actual sample ratio $r_L(S_L)$ should match the target sample ratio c closely. A small error range might be expected since traces cannot be partially included in a sample. Finally, the Directly Follows Relation, which is used as a basis of the algorithms and the sample quality measures, is defined.

Definition 2 (Directly Follows Relation [11]). *Let* $A \subseteq \mathcal{A}$*, and let* $L \in \mathbb{B}(A^*)$ *be an event log. For any* $a, b \in A$*, the Directly Follows Relation is defined via* $a >_\sigma b :\Leftrightarrow \exists_{1 \le i < |\sigma|} \ \sigma(i) = a \ \wedge \ \sigma(i+1) = b$ *for traces, and* $a >_L b :\Leftrightarrow \exists_{\sigma \in L} \ a >_\sigma b$ *for event logs. The set of all Directly Follows Pairs (DFPs) present in an event log is defined as* $\mathcal{B}(L) = \{(x, y) \in A \times A \mid x >_L y\}$. $f_\sigma(a, b) = |\{1 \le i < |\sigma| \mid \sigma(i) = a \wedge \sigma(i+1)\}|$ *is the frequency of a DFP* (a, b) *in a trace* σ*, and* $f_L(a, b) = \sum_{\sigma \in L} L(\sigma) \cdot f_\sigma(a, b)$ *the frequency in an event log* L.

2.2 Quality measures

Plenty of process discovery and conformance checking algorithms consider the Directly Follows Relation essential [2] and various quality measures have been suggested for them [11,18]. One common quality measure is the coverage $F_L(S_L)$, which calculates the ratio of DFPs of the original event log represented in the sample, i.e. [11]:

$$F_L(S_L) = \frac{|\mathcal{B}(S_L)|}{|\mathcal{B}(L)|}. \tag{1}$$

Definition 3 (Truly sampled, oversampled, undersampled and unsampled behavior [11]). *Let $A \subseteq \mathcal{A}$ and $L \in \mathbb{B}(A^*)$ be an event log, let $S_L \subseteq L$ be a sample of L, and let $t \in \mathbb{R}$ with $0 \leq t \leq 1$ be the truly-sampled-bandwidth. Then, we call $\mathcal{T}_L^{S_L} = \{(a,b) \in \mathcal{B}(L) \mid |\rho_L^{S_L}(a,b) - r_L^{S_L}| \leq t\}$ the truly sampled behavior, $\mathcal{O}_L^{S_L} = \{(a,b) \in \mathcal{B}(L) \mid \rho_L^{S_L}(a,b) - r_L^{S_L} > t\}$ the oversampled behavior, $\mathcal{U}_L^{S_L} = \{(a,b) \in \mathcal{B}(L) \mid \rho_L^{S_L}(a,b) + t < r_L^{S_L}\}$ the undersampled behavior, and $\mathcal{N}_L^{S_L} = \mathcal{B}(L) \setminus \mathcal{B}(S_L)$ the unsampled behavior. When context is clear, the sub- and superscripts are omitted.*

Based on this classification the percentages of truly sampled ($\mathcal{P}_\mathcal{T}$), oversampled ($\mathcal{P}_\mathcal{O}$), undersampled ($\mathcal{P}_\mathcal{U}$) and unsampled ($\mathcal{P}_\mathcal{N}$) DFPs are calculated as:

$$\mathcal{P}_\mathcal{T} = \frac{|\mathcal{T}|}{|\mathcal{B}(L)|},\ \mathcal{P}_\mathcal{O} = \frac{|\mathcal{O}|}{|\mathcal{B}(L)|},\ \mathcal{P}_\mathcal{U} = \frac{|\mathcal{U}|}{|\mathcal{B}(L)|},\ \mathcal{P}_\mathcal{N} = \frac{|\mathcal{N}|}{|\mathcal{B}(L)|} \tag{2}$$

Finally, the expected and actual frequency of each DFP are defined as follows:

Definition 4 (Expected and actual DFP frequency [18]). *Let $A \subseteq \mathcal{A}$ and $L \in \mathbb{B}(A^*)$ be an event log, let $S_L \subseteq L$ be a sample of L with sampling rate $c \in \mathbb{R}$. Then, the behavior of L is $\mathcal{B}(L) = \{b_1, b_2, \ldots, b_n\} \subseteq A \times A$ with $n = |\mathcal{B}(L)|$. The expected DFP frequency for a DFP $b_i = (x,y)$ is defined as $e_i = f_L(b_i) \cdot c$ and the actual DFP frequency in the sample as $s_i = f_{S_L}(b_i)$*

Relying on these definitions, a common statistical measure comparing the two distributions – the mean absolute error (MAE) – is calculated as follows [18]:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |s_i - e_i| \tag{3}$$

In our evaluation we also used further measures which are available online (GitHub[3]) like the normalized mean absolute error, coverage, the normalized root mean square error, the symmetric root mean square percentage error, the mean average percentage error and the symmetric mean average percentage error. For reference see Van der Werf et al. [18].

[3] `https://github.com/Frederik-Fonger/Sampling_RP_AB_2024`

3 Method

This section presents two new algorithms for sampling event logs. `RemainderPlus` sampling aims to optimize the representativeness of a sample, while `AllBehavior` sampling aims to generate a sample with a high coverage. The code for both algorithms can be found on GitHub[3].

Algorithm 1: `RemainderPlus` sampling

```
Data: $L \in \mathbb{B}(A)$ with $A \subseteq \mathcal{A},\ c \in (0,1],\ S_{init} \subseteq L$
Result: $S_L \subseteq L$
// First phase: initialize preliminary sample
$S_p \leftarrow [\sigma^{\lfloor L(\sigma) \cdot c \rfloor - S_{init}(\sigma)} \mid \sigma \in L \wedge \lfloor L(\sigma) \cdot c \rfloor - S_{init}(\sigma) \geq 1]$
// Second phase
while $|S_p| < \lfloor |L| \cdot c \rfloor$ do
    $\hat{F} \leftarrow \emptyset$
    for $\sigma \in L$ do
        $f_e \leftarrow L(\sigma) \cdot c - S_p(\sigma)$    // Remaining expected variant frequency
        $sgn \leftarrow 1$ if $f_e \geq 0$ else $-1$
        $\hat{f} \leftarrow sgn \cdot \lfloor 10 \cdot (|f_e| - \lfloor |f_e| \rfloor) \rfloor$    // Signed first decimal
        $\hat{F} \leftarrow \hat{F} \cup (\sigma, \hat{f})$
    end
    $\Lambda \leftarrow \arg\max_{\sigma \in L} rem(\sigma)$ with $\forall((\sigma, \hat{f}) \in \hat{F}) : rem(\sigma) = \hat{f}$

    $\Delta_{norm} \leftarrow \emptyset$
    for $\lambda \in \Lambda$ do
        $n_{\mathcal{U}} \leftarrow \sum_{b \in \mathcal{B}(\lambda) \cap \mathcal{U}_L^{S_p}} f_\lambda(b)$    // Frequency of undersampled Directly Follows Relations
        $n_{\mathcal{O}} \leftarrow \sum_{b \in \mathcal{B}(\lambda) \cap \mathcal{O}_L^{S_p}} f_\lambda(b)$    // Frequency of oversampled Directly Follows Relations
        $\delta_{norm} \leftarrow \frac{n_{\mathcal{U}} - n_{\mathcal{O}}}{\sum_{b \in \mathcal{B}(\lambda)} f_\lambda(b)}$    // Normalized difference
        $\Delta_{norm} \leftarrow \Delta \cup (\lambda, \delta_{norm})$
    end

    $S_p \leftarrow S_p \uplus \arg\max_{\lambda \in \Lambda} normdiff(\lambda)$ with $\forall((\lambda, \delta_{norm}) \in \Delta_{norm}) : normdiff(\lambda) = \delta_{norm}$
end
return $S_p$
```

3.1 RemainderPlus Sampling

The `RemainderPlus` algorithm aims to maintain the original event log's trace and behavior distribution, ensuring a representative sample. First, the occurrence of each trace variant is counted and multiplied by the sample ratio to determine the trace frequency for the sample. To avoid exceeding the sample size, the calculated frequency of traces for the sample is rounded down. Afterwards, the traces are added to the sample according to the frequency of their occurrence. When determining a preliminary sample, the expected frequencies are adjusted by subtracting the frequencies of variants already included in the preliminary sample. Next, the remaining variants are ranked based on the first decimal place of their expected frequencies relative to the original event log, considering that all expected frequencies are less than one. If two variants have the identical first decimal place, a secondary ranking criterion is applied. This criterion calculates and normalizes the difference between the number of already oversampled and currently undersampled DFPs in each variant. The variant with the highest ranking according to these criteria is then added to the sample. This ranking process is repeated, with the rankings reshuffled each time, until the target sample size is achieved. The pseudo code is provided in Algorithm 1.

3.2 AllBehavior Sampling

The second algorithm, `AllBehavior` sampling, aims to maximize the coverage of the final sample, i.e., it aims to cover as many DFPs of the original event log as possible. This algorithm follows two phases. Firstly, the variants are ranked by the cumulative frequencies of still unsampled DFPs. These are normalized by trace length to avoid a bias of traces with repeated DFPs due to loops. Next, a trace variant with the highest rank is added to the sample. This is repeated until there are no unsampled DFPs left. When the `AllBehavior` algorithm reaches optimal coverage (i.e., all DFPs are in the sample) but has not met the target sample size, it terminates and `RemainderPlus` sampling (see Section 3.1) is applied to complete the sample. If trace coverage is more essential than representativeness, then the `AllBehavior` sampling algorithm is a better alternative to the `RemainderPlus` algorithm. The pseudo code is provided in Algorithm 2.

4 Evaluation

To evaluate our sampling algorithms, we used seven event logs from the BPM community (see Table 1) and compared the results to three other sampling algorithms, namely, random, stratified and C-min [4] sampling. We only considered these three algorithms, because they do not need a discovery model prior to sampling. Therefore, we follow our goal to provide sampling algorithms for the general use in process mining including process discovery. We generated samples with target sampling ratios of $c \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ for each of the seven event logs. We used the quality measures as presented in Section 2.2 for the first part of the evaluation (Section 4.1). Following this, we evaluated the algorithms for conformance checking (see Section 4.2) and process discovery (see Section 4.3). We used the F1-score to evaluate the models [17]. For non-deterministic (random and stratified) samplings, we repeated the sampling four times and calculated an average of the quality measures.

Algorithm 2: `AllBehavior` sampling

```
Data: $L \in \mathbb{B}(A)$ with $A \subseteq \mathcal{A},\ c \in (0,1],\ S_{init} \subseteq L$
Result: $S_L \subseteq L$
// First phase
$S_p \leftarrow [\,]$                                    // Initialize preliminary sample
while $\mathcal{N}_L^{S_p} \neq \emptyset$ do                                 // See Def. 2
    if $|S_p| = \lfloor |L| \cdot c \rfloor$ then
        return $S_p$
    else
        $\sigma_{max} \leftarrow \arg\max_{\sigma \in L} \frac{1}{|\sigma|} \sum_{b \in \mathcal{N}_L^{S_p} \cap \mathcal{B}(\sigma)} f_\sigma(b)$
        $S_p \leftarrow S_p \uplus \sigma_{max}$
    end
end
// Second phase
return RemainderPlus($L,\ c,\ S_p$)
```

4.1 Sample Quality Measures

All evaluation results are available in the GitHub repository [4]. The tables and figures only show the results for a sample ratio of 0.3 due to space limitations. All other results can be found in the GitHub repository. For each sampling algorithm, we tested 84 combinations in total, stemming from 12 sample ratios for each of the 7 event logs.

Table 1. Event logs used for the evaluation.

Name	Traces	Trace Variant	Events	Activity
BPIC 2012 [6]	13087	4366	262200	24
DDLog (BPIC 2020) [7]	10500	99	56437	17
ID Log (BPIC 2020) [7]	6449	753	72151	34
Permit Log (BPIC 2020) [7]	7065	1478	86581	51
PTC Log (BPIC 2020) [7]	2099	202	18246	29
RfP Log (BPIC 2020) [7]	6886	89	36796	19
Sepsis Log [14]	1050	846	15214	16

Result Mean Absolute Error and Coverage A small *MAE* (Eq. 3) value indicates a higher representativeness. `RemainderPlus` achieves the lowest values for *MAE* for all samplings compared to existing approaches. The `RemainderPlus` (`Allbehavior`) sampling algorithm achieved the best results in 67% (24%) of the evaluated combinations. The performance of the other algorithms depends on

[4] `https://github.com/Frederik-Fonger/Sampling_RP_AB_2024`

the properties of the event log (e.g., size, trace frequencies). For example, the random sample nearly matches the best results in the "BPI-Challenge 2012" log, but underperforms for the event logs "Domestic Declarations" or "Request for Payment". The `AllBehavior` algorithm achieves the highest *coverage* among all evaluated event logs for sample ratios of up to 0.5. The evaluation indicates that other sampling algorithms achieve similar coverage for sample ratio values above 0.6. Although the `AllBehavior` algorithm does not match the optimal *coverage* at a sample ratio of 0.01, it still outperforms all other algorithms. For additional details, see GitHub [5]. The evaluation results for `RemainderPlus` show that it ranks closely with other methods, usually slightly below the random sample, but superior to both C-min and stratified sampling. All algorithms tend to produce a higher *coverage* value when the sample size increases, which might be explained by the increasing probability of rare cases occurring.

Runtime The runtime of each sampling iteration is shown in Table 2. The computation was performed on a processor with 6 x 3.7 GHz and 16 Gigabyte of Memory. The random and the stratified sampling both rely on a randomized selection of traces. As a result, there is no additional calculation required for the selection of cases. That being considered, it is expected that they have the shortest runtime in comparison to other approaches. The increase in runtime for the `AllBehavior`, `RemainderPlus` and C-min sampling depends on the size and complexity of the event log to be sampled. The `RemainderPlus` algorithm is the fastest of the non-randomized algorithms in 92% of the evaluated combinations.

Table 2. The runtime of sampling iterations in seconds for a sample ratio of 0.3.

	AB	RP	Random	Stratified	C-min
BPIC 2012	149.8878	119.715	**0.0219**	0.9362	205.0922
DD Log	9.7479	9.8556	**0.0120**	0.2296	149.2675
ID Log	10.6056	7.7004	**0.0135**	0.2588	421.1549
Permit Log	29.8082	20.4941	**0.0434**	0.4446	35.3743
PTC Log	1.114	1.0612	**0.0045**	0.0900	5.5172
RfP Log	5.2101	5.3636	**0.0107**	0.1927	36.8803
Sepsis Log	2.1732	2.2789	**0.0037**	0.2045	3.7131

4.2 Sampling for Conformance Checking

To evaluate the sampling methods for conformance checking, the process model was discovered from the original event log and conformance checking was applied with and without sampling. We applied process discovery with the miner-threshold of $t \in \{0,\ 0.2,\ 0.4,\ 0.6,\ 0.9\}$. This evaluation was done for the aforementioned 84 combinations, resulting in 420 combinations. Conformance checking was also applied for the original event log to have a baseline. A score for

[5] `https://github.com/Frederik-Fonger/Sampling_RP_AB_2024`

comparing the sample algorithms was calculated by dividing the F1-score when sampling was applied by the baseline F1-score of the original event log, which we denote as $\lambda_{f_1} = |\frac{f_1^{sample}}{f_1^{mioriginallog}} - 1|$. From here, we present the data for a sample ratio of $r = 0.01$ in the tables, as the differences for process discovery and conformance checking are more significant for smaller sample ratios. All results for other sample ratios can be found in the GitHub [3]. Overall, `AllBehavior` outperforms all other algorithms in terms of keeping the F1-score consistently close to the baseline. This is obvious since the algorithm focuses on high coverage and retaining the DFPs for smaller sample ratios (see Sec. 4.1). Most algorithms retain results close to the baseline for sample ratios larger than $r = 0.05$, as shown in Fig. 2. The `RemainderPlus` algorithm mostly achieves the second best results, as shown in Table 3. For the small sample ratio of $r = 0.001$, all algorithms significantly deviate from the baseline. The decreased processing time for the conformance checking calculation is significant. E.g., for the Permit Log, the calculation time decreases from 55 seconds on the original log to 23 seconds on a sample with $r = 0.5$ and to 1.9 seconds on a sample with $r = 0.01$.

Table 3. F1-score ratio λ_{f_1} between the sample and original log, with both F1-scores calculated against the model discovered from the original log (sample ratio $r = 0.01$, inductive miner with threshold set to 0)

	AB	RP	Random	Stratified	C-min
BPIC 2012	0.1478	0.2196	**0.0948**	0.1320	0.1325
DD Log	**0.0219**	0.1590	0.2275	0.1575	0.2090
Permit Log	**0.1667**	0.3833	0.3709	0.4159	0.3860
ID Log	**0.0573**	0.2695	0.2652	0.2853	0.2954
PTC Log	**0.2957**	0.3698	0.3735	0.4055	0.4486
RfP Log	**0.2024**	0.2175	0.3269	0.2358	0.4221
Sepsis Log	**0.6118**	0.6187	0.6274	1.0000	0.6621

4.3 Sampling usage for Process Discovery

To evaluate the sampling algorithms for process discovery, the inductive and heuristic miners are applied to every sample. Afterwards, conformance checking is applied to the original event log. The score is calculated by dividing the conformance checking results through the results from the original event log (λ_{f_1}). Due to runtime constraints, we only evaluated process models for 5 of the 7 event logs, as we stopped the conformance checking calculation for these models after 48 hours. Therefore, the number of combinations decreased for this setting to 300 combinations. The results are shown in Table 4. The results vary depending on the configuration and event log quality. For example, for the International Declarations log, `AllBehavior` sampling is able to reach an equal F1-score to the baseline until a sample ratio of $r = 0.05$ (Fig. 3). With smaller sample ratios it still remains close to the baseline. On the other hand, for the Request for

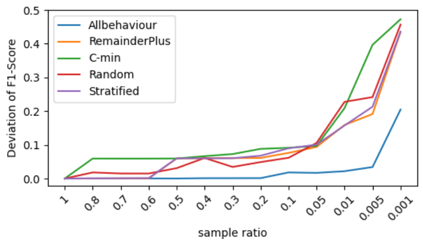

Fig. 2. F1-score ratio λ_{f_1} between the sample and original log with both F1-scores calculated against the model discovered from the original log (Domestic Declarations log, models discovered using the inductive miner with threshold set to 0)

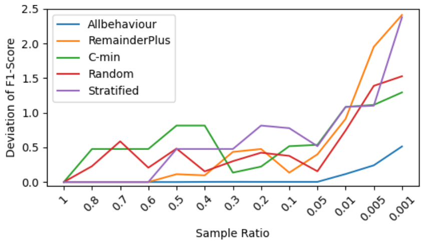

Fig. 3. F1-score ratio λ_{f_1} between the model discovered from the sample and the model from the original log, with both F1-scores calculated against the original log (International Declarations log, models discovered using the inductive miner with threshold set to 0)

Payments log, the results are not fully convincing. Overall, in 66 % of the tested combinations, `AllBehavior` sampling's F1-score was closest to the baseline.

Table 4. F1-score ratio λ_{f_1} between the model discovered from the sample and the model from the original log with both F1-scores calculated against the original log (sample ratio $r = 0.01$, inductive miner with threshold set to 0)

	AB	RP	Random	Stratified	C-min
DD Log	0.3406	0.1078	**0.1068**	0.1378	0.6575
Permit Log	**0.0644**	1.034	0.9953	0.5118	0.5709
PTC Log	**0.0384**	1.7245	1.6028	2.9022	3.0264
RfP Log	0.3795	0.8021	**0.0859**	0.6859	0.5423
ID Log	**0.1155**	0.9129	0.7448	1.0847	1.0849

5 Related Work

Different random sampling approaches have been evaluated in the literature [4,11,13,18]. The C-min sampling algorithm aims to optimize the earth mover's distance between the original log and the sample [4]. The log-rank algorithm uses a graph-based ranking model for sampling, however, due to the lack of code availability it is not included in our evaluation [13]. Therefore, we compared `RemainderPlus` and `AllBehavior` against random sampling and C-min sampling algorithms and discussed the results above. However, to the best of our knowledge, no structured approach exist to compare sampling algorithms. While some algorithms compare metrics of event logs [4,11], others consider the process model as the indicator for quality [13]. Also, a direct comparison of existing approaches is difficult since the approaches were not evaluated using the same sample sizes [4,10,11,18]. To overcome these shortcomings in existing sampling evaluations, we evaluated the algorithms based on plenty of configurations to demonstrate the efficiency of our two algorithms. Sani et al. propose using different sampling approaches depending on the quality of event logs and use cases [16] as well as evaluate different sampling strategies [8,9,15].

Sampling strategies have also been suggested for conformance checking [3, 10]. In [10], the authors suggest a conformance checking sampling algorithm that takes a log and a model as input and applies relevance-guided sampling of event logs [10] to decrease runtime processing. The approach of [3] decreases the runtime of optimal alignments. Additionally, Bauer et al. found that a subset of a log, sometimes with a size below 1% of the original log, is enough to assess its quality while significantly reducing processing times [3]. Therefore, the size of a representative subset depends on the complexity of the log [3].

6 Conclusion

This paper introduces two novel event log sampling algorithms, `RemainderPlus` and `AllBehavior`. While `RemainderPlus` focuses on optimizing the representativeness of the sample, `AllBehavior` aims to maximize coverage. Both algorithms select traces for sampling deterministically, which means that the results are reproducible and not random. The evaluation results show that `RemainderPlus` outperforms existing algorithms in terms of sample representation to the original event log by demonstrating lower error values for MAE. It is therefore well-suited for general application where representative samples are required. The `AllBehavior` algorithm consistently produces samples with high coverage, but trades enhanced coverage for trace distribution accuracy. This trade-off is especially evident for smaller sample sizes. Surprisingly, while `RemainderPlus` performs better on the individual algorithm metrics on the model level, `AllBehavior` outperforms `RemainderPlus` for process discovery and conformance checking applications. This is most probably because `AllBehavior` focuses on covering as much behavior as possible. Generally, the evaluation results indicate different significance for process discovery and conformance checking for smaller sample ratios. The results for process discovery show that `AllBehavior` is more appropriate.

Potential validity threats to our methods and evaluation include the risk of overfitting from testing on specific event logs, which we countered by using a wide range of sample sizes and event log characteristics. Also, challenges arose with respect to a comparative analysis against existing approaches since some code implementations are not accessible. A promising direction for future research might be a holistic combination of sampling techniques that can be individually adjusted based on the characteristics of the event log and some specific requirements imposed by the data analysis. Both algorithms presented in this paper promise a more efficient analysis of large event logs for process mining, which allow novel insights in disciplines with high volume of data [12].

Acknowledgments. This project has received funding from the Federal Ministry for Economic Affairs and Climate Action under the Marispace-X project grant no. 68GX21002E and the State of Schleswig-Holstein under the Datencampus project grant no. 220 21 016. The project ProcessPig is funded by the European Union within the framework of the European Innovation Partnership (EIP-AGRI) and the state program rural areas of the state Schleswig-Holstein (LPLR) (`www.eip-agrar-sh.de`).

References

1. van der Aalst, W., et al.: Process Mining Manifesto. In: BPM 2011 Workshops. LNBIP, vol. 99, pp. 169–194. Springer, Berlin, Heidelberg (2012)
2. van der Aalst, W.M.: Foundations of process discovery. In: Process Mining Handbook, pp. 37–75. Springer, Cham (2022)
3. Bauer, M., van der Aa, H., Weidlich, M.: Sampling and approximation techniques for efficient process conformance checking. Inf. Syst. **104**, 101666 (2022)
4. Bernard, G., Andritsos, P.: Selecting Representative Sample Traces from Large Event Logs. In: ICPM 2021. pp. 56–63. IEEE, Eindhoven, Netherlands (2021)
5. Carmona, J., van Dongen, B., Weidlich, M.: Conformance checking: foundations, milestones and challenges. In: Process Mining Handbook, pp. 155–190. Springer, Cham (2022)
6. van Dongen, B.: Bpi challenge 2012 (2012), `https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204`
7. van Dongen, B.: Bpi challenge 2020 (2020), `https://data.4tu.nl/collections/_/5065541/1`
8. Fani Sani, M., 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: ADBIS 2019, CCIS, vol. 1064, pp. 391–404. Springer, Cham (2019)
9. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: The impact of biased sampling of event logs on the performance of process discovery. Computing **103**(6), 1085–1104 (2021)
10. Kabierski, M., et al.: Sampling what matters: Relevance-guided sampling of event logs. In: ICPM 2021. pp. 64–71 (2021)
11. Knols, B., van der Werf, J.M.E.M.: Measuring the Behavioral Quality of Log Sampling. In: ICPM 2019. pp. 97–104. IEEE, Aachen, Germany (2019)
12. Koschmider, A., et al.: Process Mining for Unstructured Data: Challenges and Research Directions. In: Modellierung 2024, pp. 119–136. GI, Bonn (2024)
13. Liu, C., et al.: Sampling business process event logs using graph-based ranking model. Concurrency Computat.: Pract. Exper. **33**(5), e5974 (2021)
14. Mannhardt, F., Blinde, D.: Analyzing the Trajectories of Patients with Sepsis using Process Mining. In: RADAR+EMISA 2017, Essen, Germany, June 12-13, 2017. pp. 72–80. CEUR Workshop Proceedings, CEUR-WS.org (2017)
15. Sani, M., Van Zelst, S., Van Der Aalst, W.: Improving the performance of process discovery algorithms by instance selection. Comp. Sci. Inf. Syst. **17**(3), 927–958 (2020)
16. Sani, M.F., et al.: Event Log Sampling for Predictive Monitoring. vol. 433, pp. 154–166 (2022)
17. Van Der Aalst, W.: Process Mining. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
18. Van Der Werf, J.M.E., Polyvyanyy, A., Van Wensveen, B.R., Brinkhuis, M., Reijers, H.A.: All that glitters is not gold: Four maturity stages of process discovery algorithms. Inf. Syst. **114**, 102155 (2023)