Sampling What Matters: Relevance-guided Sampling of Event Logs

Martin Kabierski, Hoang Lam Nguyen, Lars Grunske, and Matthias Weidlich
Department of Computer Science, Humboldt-Universität zu Berlin, Germany
martin.kabierski | hoang.lam.nguyen | lars.grunske | matthias.weidlich@hu-berlin.de

***Abstract*—The comparison of a model of a process against event data recorded during its execution, known as conformance checking, is an important means in process analysis. Yet, common conformance checking techniques are computationally expensive, which makes a complete analysis infeasible for large logs. To mitigate this problem, existing techniques leverage data samples. Then, the result quality depends on the relevance of the sample for a specific analysis task. Existing sampling strategies therefore rely on a static assumption on what constitutes relevant event data, which is generally unknown a priori.**

In this paper, we present relevance-guided sampling of event logs. Instead of employing a fixed relevance hypothesis, our approach learns the characteristics of event data that determine its relevance for conformance checking. To this end, we first explore the correlations between characteristics of the event data and the goal of a conformance checking task, before exploiting these correlations to guide the selection of a data sample. We present different instantiations of this approach and demonstrate that they significantly improve the quality of samples, and hence of conformance checking results, compared to baseline strategies.

I. Introduction

The field of conformance checking targets the comparison of a model of a process against event data recorded during its execution [1]. Conformance checking has manifold applications. It enables the assessment of operational goals and risks, supports compliance checking, and strengthens the trust into operational decisions made based on the model [2], [3].

State-of-the-art techniques for conformance checking compute alignments between the traces of an event log, i.e., sequences of events recorded during one execution of a process, and the execution sequences defined by a process model. The construction of an alignment that optimizes a cost function for deviations is computationally hard, though, and common algorithms show an exponential time complexity in the size of the trace and process model [1]. To mitigate the resulting performance issues, conformance checking may be applied only to a sample of the traces that are available [4], [5]. Generalizing the results obtained for a sample, insights into the conformance of the given data and model are then obtained in a small fraction of the time, otherwise required by exhaustive analysis.

In sample-based conformance checking, the quality of the obtained result directly depends on the relevance of the sample for a specific conformance checking setting, i.e., how well the sample approximates the information relevant for the intended analysis, as present in the complete log. For instance, if the analysis goal is to gain insights into the characteristics of non-conforming behaviour using alignments, then only the deviating traces are relevant and should be included in a sample. Existing sampling strategies therefore rely on static assumptions on how to characterise relevant traces. An example is the computation of aggregated conformance results, such as the overall fitness of all traces. Here, each trace is considered equally relevant and random sampling can be employed in order to approximate the true frequency distribution of fitness values over all traces [4]. While other static trace relevance assumptions have been proposed, e.g., based on the trace length (to avoid incomplete traces) and trace similarity (to focus on the main behaviour of a process) [6], they have not been linked to the result quality of conformance checking.

Hence, there is a notable research gap for conformance measures for which the relevance of traces does not coincide with the frequency of the represented behaviour, or other static relevance assumptions. When striving for a certain notion of *completeness*, such as detecting *all* non-conforming traces or *all* activities involved in deviations, it is an open question how to guide the selection of a sample of mostly relevant traces.

In this paper, we address this gap with an approach to learn the characteristics of traces that determine their relevance for conformance checking. Our idea is to first explore the correlations between trace characteristics and the goal of a conformance checking task, which are then exploited to guide the selection of traces. This way, the relevance of traces for the sample is assessed dynamically by learning a respective model. More specifically, our contributions are summarized as:

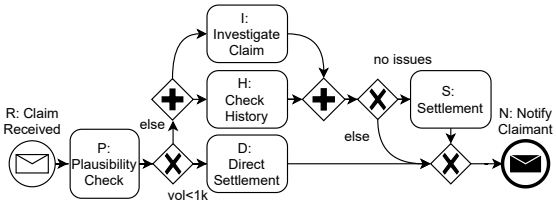
- We present a framework for relevance-guided sampling of event logs (§III). It samples traces using indices over a log. Here, the relevance of the index keys is assessed using a knowledge base, which is built up dynamically based on the conformance results obtained for traces drawn earlier.
- We provide schemes for log indexing (§IV). They rely on trace-level and event-level features, potentially combined with similarity-based hashing.
- We propose a procedure for guided sampling as part of the framework (§V). It explores and exploits the correlations between index keys and conformance results.

An evaluation with real-world event logs show the effectiveness of our approach (§VI). Compared to trace selection that is random or employs static assumptions on the relevance of traces, we increase the number of sampled traces that show a certain conformance result, i.e., that are non-conforming. This is achieved with a negligible runtime overhead that, itself, is only a fraction of the time needed for alignment construction.

2021 3rd International Conference on Process Mining (ICPM 2021), available at https://doi.org/10.1109/ICPM53251.2021

	Trace attributes	Events, potentially with event attributes	Conf.
ξ_1	vol=2k, type=regular	R, P, I, H (issues=$\emptyset$), S, N	✓
ξ_2	vol=2k, type=VIP	R, P, D, I (issues=$\emptyset$), H, S	✗
ξ_3	vol=10k type=regular	R, P, H, I (issues={duplicate}), N	✓
ξ_4	vol=500, type=VIP	R, D, N	✗
ξ_5	vol=5k, type=regular	R, P, I (issues={fraud}), H, N	✓
ξ_6	vol=1k, type=regular	R, P, H, I (issues=$\emptyset$), S, N	✓
ξ_7	vol=120, type=regular	R, D, N, P	✗
ξ_8	vol=3k, type=regular	R, P, H, I (issues={fraud}), N	✓

(a)



(b)

Figure 1: (a) Event log for executions of a claim handling process; (b) a model of the respective process.

II. Background

A. Event logs

We rely on a relational model for event logs. It is based on a finite set of activities $\mathcal{A}$ and a universe of events $\mathcal{E}$, where each *event* $e \in \mathcal{E}$ denotes the execution of an *activity*, written as $e.act \in \mathcal{A}$. A *trace* represents a single execution of a process, captured by a sequence of events $\xi = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$, such that an event can occur in at most one trace. We write $\xi(i)$, $1 \leq i \leq |\xi|$ for the i-th event. An *event log* is a set of traces, $L \subseteq 2^{\mathcal{E}^*}$. Traces for which the events represent the same sequence of activity executions are of the same *trace variant*.

Events and traces may carry additional information on the execution context, such as data used as input or timestamps on the start and end of execution. We capture such context information by a set of *data attributes* $\mathcal{D} = \{D_1, \ldots, D_p\}$ of domain $\mathrm{dom}(D_i)$, $1 \leq i \leq p$. Here, we distinguish distinct sets of attributes defined for traces, $\mathcal{D}_t$, and events, $\mathcal{D}_e$, with $\mathcal{D} = \mathcal{D}_t \cup \mathcal{D}_e$. We write $\xi.D$ ($e.D$) to refer to the value of attribute $D \in \mathcal{D}_t$ ($D \in \mathcal{D}_e$) for a trace ξ (an event e).

An event log of a claim handling process is shown in Fig. 1a. Here, trace attributes include the claim volume and the claimant type. The events denote activities, e.g., related to the receipt of a claim (R) or a plausibility check (P), see the model in Fig. 1b for the activity names. The activity of investigating a claim further has an attribute for the detected issues.

B. Conformance checking

Conformance checking compares an event log against a process model. Abstracting from process modelling languages, we capture the behaviour of a process model as a set of sequences of activities, $M \subseteq \mathcal{A}^*$, that represent complete execution sequences as defined by the model.

To compare a trace of an event log against this set of execution sequences, an *alignment* is commonly computed [7]. Let $\perp$ be a dedicated skip symbol, yielding extended sets of events and activities, $\mathcal{A}_\perp = \mathcal{A} \cup \{\perp\}$ and $\mathcal{E}_\perp = \mathcal{E} \cup \{\perp\}$. Then, given a trace $\xi \in L$ and an execution sequence $\pi \in M$, an alignment is a sequence of steps, $\gamma = \langle (e_1, a_1), \ldots, (e_n, a_n) \rangle \in (\mathcal{E}_\perp \times \mathcal{A}_\perp)^*$ (where a step $(\perp, \perp)$ is forbidden). Here, the projection of γ on the first component, ignoring $\perp$, equals ξ, and the projection of γ on the second component, ignoring $\perp$, equals π. For a step $\gamma(i)$, $1 \leq i \leq |\gamma|$, the projections are denoted by $\gamma(i)|_1 \in \mathcal{E}$ and $\gamma(i)|_2 \in \mathcal{A}$, respectively.

The quality of an alignment is determined based on a cost function. Intuitively, it assigns high costs to steps (e_i, a_i) where events or activities are without a counterpart, i.e., either $e_i = \perp$ or $a_i = \perp$. If $e_i.act = a_i$, a cost of zero is commonly assigned. This way, a cost-optimal alignment minimizes the edit distance between a trace and an execution sequence. Search-based algorithms to construct such an optimal alignment show an exponential time complexity in the size of the trace and process model, though [1].

For instance, for the model given in Fig. 1b and trace ξ_2 of Fig. 1a, an optimal alignment is constructed as follows:

Trace ξ_2	R	P	D	I	H	S	$\perp$
Execution sequence	R	P	$\perp$	I	H	S	N

C. Conformance checking tasks

Alignments as illustrated above serve as the basis for various conformance checking tasks. This includes aggregated measures on the overall conformance, e.g., the fitness as the average, normalized cost of alignments over all traces of an event log. Due to the aggregation, all traces are equally relevant for these conformance checking tasks. Therefore, random sampling can be expected to yield a representative sample in these cases.

For other conformance checking tasks, however, the hypothesis of all traces being equally relevant is not justified. In particular, tasks that target some notion of completeness of conformance results require a different approach. Examples for such tasks are the identification of *all* traces that are non-conforming or *all* activities involved in deviations.

The event log in Fig. 1a, contains three non-conforming traces, ξ_2, ξ_4, and ξ_7. Sampling three traces randomly, the probability of selecting these traces is $3/8 \cdot 2/7 \cdot 1/6 \approx 1.8\%$. Yet, we observe that the deviating traces can be characterised by the trace attribute *type* having value *VIP* (in ξ_2 and ξ_4), as well as the occurrence of an activity pattern $\langle R, D, N \rangle$ (in ξ_4 and ξ_7). Knowing this correlation would, therefore, enable the selection of a sample that contains all deviating traces.

III. Sampling Framework

Our general approach for sampling of an event log that is guided by the relevance of traces for conformance checking is summarized in Fig. 2. It comprises two steps, log indexing and guided sampling, to construct a sample from a given log.

As a first step, the original event log is split into, potentially overlapping sets of traces that share specific characteristics.

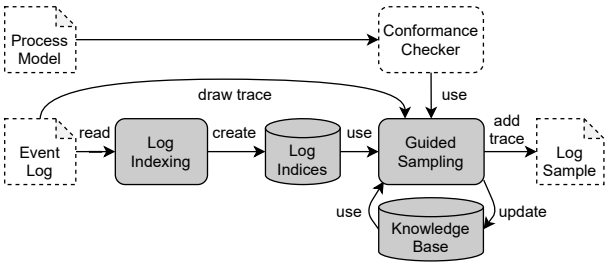


Figure 2: Framework for relevance-guided sampling.

Intuitively, each set captures a certain context of process execution, for which the correlation with the result of a conformance checking task shall be assessed. As such, the number of considered sets of traces corresponds to the granularity of the model that is learned to understand which characteristics hint at conforming and non-conforming traces, respectively. Once a scheme to split the log has been chosen, the outcome of this step are indices over the original event log, i.e., data structures that use trace characteristics as keys mapped to the respective sets of traces. We discuss these indices in §IV.

The second step is a guided sampling procedure. It draws traces from the event log to add them to a sample. The selection of a trace is hereby guided by a knowledge base that captures the correlation of index keys, and hence sets of traces sharing specific characteristics, with conformance checking results. To learn the knowledge base dynamically, guided sampling distinguishes an exploration phase, in which traces are drawn randomly from the log, and an exploitation phase, in which traces are drawn according to a probability distribution over index keys, that is derived from the positive correlations captured in the knowledge base. In any case, a selected trace is checked for conformance and, based on the result, the knowledge base is updated. Sampling stops upon reaching a predefined termination criterion. The details of our procedure for relevance-guided sampling are given in §V.

IV. Log Indexing

To split an event log into into sets of traces that share specific characteristics, we first construct a feature index. It maps from a predefined set of features to the traces, sharing them. Once the correlation between these features and a conformance result is assumed, the feature index may be employed to guide the sampling procedure. However, when considering behavioural features, a plain feature index would be sparse, i.e., it would contain many indices mapping to small sets of indices. Therefore, we combine it with a second structure, a trace similarity index, which employs an additional similarity estimation step. Below, we first define the two indices before turning to their construction for a given event log.

The Feature Index

Given an event log L and a set of features F, a feature index is a structure $I : F \to 2^L$. It enables the retrieval of a set of traces that show a specific feature. In the remainder, we consider event-level features, F_e, and trace-level features, F_t. Given an event $e \in \mathcal{E}$ and a feature $f \in F_e$, we write $e \models f$ if the feature is present in the event. Analogously, $\xi \models f$ denotes that feature $f \in F_t$ occurs in trace $\xi \in \mathcal{E}^*$.

Event-level features. Given an event, the activity for which execution is signalled represents a feature. In addition, all values of data attributes linked to the event also denote features. To simplify notation, we assume distinct domains of data attributes. Also, for large, numeric domains, we first reduce the domain using equi-width histograms with a fixed width of buckets. Then, the set of event-level features for an event log L is:

$$F_e = \bigcup_{\xi \in L; 1 \leq i \leq |\xi|} \{\xi(i).act\} \cup \{\xi(i).d \mid d \in \mathcal{D}_e\}.$$

For the example log in Fig. 1a, the set of features would include the set of activities $\{R, P, I, H, S, N, D\}$ and the values of the *issues* data attribute, $\{\text{fraud}, \text{duplicate}\}$.

Trace-level attribute features. A first set of features for traces is induced by the values of data attributes assigned to them. Again, we assume distinct domains, while large numeric domains are abstracted with equi-width histograms. For an event log L, we define the attribute features as:

$$F_t^a = \bigcup_{\xi \in L} \{\xi.d \mid d \in \mathcal{D}_t\}.$$

For our example, Fig. 1a, features are induced by fixed width buckets for the *vol* attribute, e.g., $\{[0, 1\text{k} - 1], \ldots, [10\text{k}, 11\text{k} - 1]\}$, and the values of the *type* attribute, $\{\text{regular}, \text{VIP}\}$.

Trace-level behavioural features. A second set of trace-level features considers the control-flow behaviour. Specifically, we incorporate k-grams of activities for which the subsequent execution is indicated by the events of a trace. As such, the set of behavioural features for an event log L is defined as:

$$F_t^b = \bigcup_{\xi \in L} \{(\xi(i).act, \ldots, \xi(i+k-1).act) \in \mathcal{A}^k \mid 0 \leq i \leq |\xi| - k + 1\}.$$

Considering our example and k=3, the set of features would contain triples such as (R, P, I), (P, I, H), and (R, D, N).

The Trace Similarity Index

Since a plain index over the individual trace-level behavioural features would suffer from the data sparsity problem, we construct an additional index that captures the behavioural similarity between traces through an additional abstraction of the trace-level behavioural features. Given an event log L and the set of trace-level behavioural features F_t^b, the similarity-based index is a structure $J : L \to 2^L$. More concretely, J allows to retrieve the set of traces in L that are similar to a given trace w.r.t. their behaviour.

Behavioural Trace Similarity. Analogously to the above definition of behavioural features, given a trace ξ, we define its set of activity k-grams as:

$$S_k(\xi) = \{(\xi(i).act, \ldots, \xi(i+k-1).act) \in \mathcal{A}^k \mid 0 \leq i \leq n-k+1\}.$$

We utilize the definition of activity k-grams to define a notion of behavioural similarity between two traces.

Given a similarity threshold $\epsilon \in [0, 1]$, we consider two traces ξ_s and ξ_t similar w.r.t. their behaviour if

$$sim_{jaccard}(S_k(\xi_s), S_k(\xi_t)) \geq \epsilon, \quad sim_{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where $sim_{jaccard}(A, B)$ is the Jaccard similarity coefficient.

Hashing-based similarity estimation. In order to efficiently determine similar pairs of traces, we rely on two established techniques from the field of data mining, namely Minhashing and Locality-Sensitive Hashing (LSH) [8]. Minhashing allows to derive compact representations of sets, called *signatures*, that can be used to estimate the Jaccard similarity. A minhash signature is determined as follows: Let S be a set (in our setting, the set of activity k-grams) and $h_1(x), \ldots, h_n(x)$ be n hash functions over the elements of S. The minhash signature $h_{min}(S)$ of S is then computed as the sequence of minimum hash values (minhashes) of all elements in S, i.e., $h_{min}(S) = \langle \min_{s \in S} h_1(s), \ldots, \min_{s \in S} h_n(s) \rangle$. The estimated Jaccard index between two sets corresponds to the fraction of minhash values their minhash signatures agree on.[1] Locality-Sensitive Hashing (LSH) is a complementary technique that avoids the naive approach of computing pairwise similarities by determining *candidate pairs* that are likely to be similar. LSH divides each minhash signature into b bands of r rows and applies a hash function to each band. If, for a given band, two minhash signatures hash into the same bucket, they are considered a candidate pair as they have matching portions in their signatures. Each candidate pair is then likely to have a similarity value over a threshold t, which is commonly approximated as $(1/b)^{(1/r)}$. We use $n = 100$ hash functions and set $b = 10, r = 10$, which guarantees a similarity threshold $\epsilon \approx 0.79$ with an expected error ≤ 0.1 in the estimation.

Index Construction

The construction of the two indices I and J, given an event log L, sets of trace-level and event-level features F_t, F_e, and the set of behavioural trace-level features $F_t^b \subseteq F_t$, is summarized in Alg. 1. The algorithm starts by initializing the indices (line 1). Then, it iterates over the log to index each trace (lines 2-13):

Indexing by event-level features. First, the trace is associated with each event-level feature it contains. This is done by iterating over the set of event-level features and checking for each feature whether there exists an event in the trace where the particular feature is present (lines 3-4).

Indexing by trace-level features. Similarly, by iterating over the set of trace-level features, traces are added to the feature index I for trace-level features (lines 5-9).

Indexing by behavioural trace similarity. In addition, during the processing of the trace-level features, the set of observed behavioural trace-level features (i.e., the set of activity k-grams) is computed (lines 6 and 9). Afterwards, the corresponding minhash signature is derived (line 10) and the bands of the signature are hashed into LSH buckets (line 11). Finally, the trace is added to each bucket in index J (lines 12-13).

[1]For the similarity estimation, minhashing using n hash functions guarantees an expected error of $\mathcal{O}(1/\sqrt{(n)})$.

Algorithm 1: Log Indexing

input : L, an event log;
$F = F_t \cup F_e$, set of (trace-level and event-level) features;
$F_t^b \subseteq F_t$, set of behavioural trace-level features.
output : $I : F \to 2^L$, a feature index;
$J : L \to 2^L$, a trace similarity index.

1 $I, J \leftarrow \emptyset$; /* Initialize indices */
2 **for** $\xi \in L$ **do** /* For each trace */
3 **for** $f \in F_e$ **do** /* For each event-level feature */
4 **if** $\exists\, 1 \leq i \leq |\xi| : \xi(i) \models f$ **then** $I(f) \leftarrow I(f) \cup \{\xi\}$;
5 **for** $f \in F_t$ **do** /* For each trace-level feature */
6 $F_b^\xi \leftarrow \emptyset$; /* Behavioural features in ξ */
7 **if** $\xi \models f$ **then**
8 $I(f) \leftarrow I(f) \cup \{\xi\}$;
9 **if** $f \in F_t^b$ **then** $F_b^\xi \leftarrow F_b^\xi \cup \{f\}$;
10 $m \leftarrow minhash(F_b^\xi)$;
11 $B \leftarrow LSH(m)$; /* Locality-Sensitive Hashing */
12 **for** $b \in B$ **do** /* For each LSH bucket */
13 $J(b) \leftarrow J(b) \cup \{\xi\}$;
14 **return** I, J; /* Return the indices */

V. Sampling Procedure

We first introduce our general procedure for guided sampling, before instantiating it with the two approaches to log indexing. Finally, we elaborate on further design choices.

A. General Algorithm

Our procedure for guided sampling is defined in Alg. 2. In addition to the general input of any conformance checking task, i.e., an event log L and a process model M, it takes as input a set of (trace-level and event-level) features F, and the indices I, J as defined above. It returns a log sample $L' \subseteq L$.

First, the sample L', the knowledge base K^+, K^-, K_d^+, K_d^-, and the correlation coefficients r are initialized (lines 1-3). The knowledge base includes counters for four situations, induced by the presence of a feature (present $K_{(d)}^+$ vs. absent $K_{(d)}^-$) and the conformance result (conforming $K^{+/-}$ vs. deviating $K_d^{+/-}$). They serve to compute coefficients φ of the correlation of each feature with non-conformance.

The sample is constructed trace by trace, while a termination predicate ψ_{stop}, discussed below, is not reached (lines 4-37).

Exploration and exploitation. The selection of the trace ξ to add is realized as exploration or exploitation. If the predicate $\psi_{explore}$, also discussed below, indicates that the procedure is in the exploration phase (line 6), one of the remaining traces of the event log is chosen randomly (line 7). In the exploitation phase, in turn, the selection is based on the correlation coefficients. To this end, all features that are positively correlated with non-conformance are collected (line 9), a probability distribution over the features is derived from the correlation coefficients (lines 10-12), and one feature is drawn according to this distribution (line 13). Using this feature f, the trace ξ is drawn using the indices over the event log (line 14). The specific realization of this step, in combination with the choice of the set of considered features, yields a specific instance of our sampling procedure. Below, we provide two such instantiations.

Algorithm 2: Procedure for Guided Sampling

input : L, an event log; M, a process model;
$F = F_t \cup F_e$, set of (trace-level and event-level) features;
I, J, a feature index and a trace similarity index.
output : $L' \subseteq L$, a sample of the event log.

1 $L' \leftarrow \emptyset$; /* The sampled log */
2 $K^+, K^-, K_d^+, K_d^- \leftarrow \emptyset$; /* The knowledge base */
3 $r \leftarrow \emptyset$; /* The correlation coefficients */
4 **while** $\neg\psi_{stop}$ **do** /* While sampling shall continue */
5 $\xi \leftarrow \emptyset$;
6 **if** $\psi_{explore}$ **then** /* EXPLORATION phase */
7 $\xi \leftarrow sampleRandomTrace(L \setminus L')$;
8 **else** /* EXPLOITATION phase */
/* Set of positively correlated features */
9 $F^+ \leftarrow \{f \in F \mid r(f) > 0\}$;
10 $r_\Sigma = \sum_{f \in F^+} r(f)$; /* Sum of correl. coeff. */
11 $d \leftarrow \emptyset$; /* Probability distribution over F^+ */
12 **for** $f \in F^+$ **do** $d(f) \leftarrow r(f)/r_\Sigma$;
13 $f \leftarrow sampleFeatureFromDistribution(d)$;
14 $\xi \leftarrow sampleRandomTrace(f, L', I, J)$;
15 **if** $\xi = \emptyset$ **then continue** ;
16 $L' \leftarrow L' \cup \xi$; /* Add trace to sample */
17 $\gamma \leftarrow getAlignment(\xi, M)$;
18 $C \leftarrow \emptyset$; /* Deviation context */
19 **for** $1 \leq i \leq |\gamma|$ **do** /* For each alignment step */
20 **if** $\gamma(i)|_1 = \perp \vee \gamma(i)|_2 = \perp$ **then** /* If non-conf. */
21 $C \leftarrow C \cup \psi_{context}(\gamma(j)|_1)$; /* Select events */
22 **for** $f \in F_t$ **do** /* For each trace-level feature */
23 **if** $\xi \models f$ **then**
24 **if** $C = \emptyset$ **then** $K^+(f) \leftarrow K^+(f) + 1$;
25 **else** $K_d^+(f) \leftarrow K_d^+(f) + 1$;
26 **else**
27 **if** $C = \emptyset$ **then** $K^-(f) \leftarrow K^-(f) + 1$;
28 **else** $K_d^-(f) \leftarrow K_d^-(f) + 1$;
29 **for** $f \in F_e$ **do** /* For each event-level feature */
30 **for** $1 \leq i \leq |\gamma|$ **do**
31 **if** $e = \gamma(i)|_1 \models f$ **then**
32 **if** $e \in C$ **then** $K_d^+(f) \leftarrow K_d^+(f) + 1$;
33 **else** $K^+(f) \leftarrow K^+(f) + 1$;
34 **else**
35 **if** $e \in C$ **then** $K_d^-(f) \leftarrow K_d^-(f) + 1$;
36 **else** $K^-(f) \leftarrow K^-(f) + 1$;
/* Update correlation coefficients */
37 **for** $f \in F$ **do** $r(f) \leftarrow \varphi(K^+(f), K^-(f), K_d^+(f), K_d^-(f))$;
38 **return** L' ; /* Return log sample */

In any case, trace selection in the exploitation phase is repeated, if no trace was chosen (line 15).

Alignment. The selected trace is added to the sample (line 16) and aligned with the process model (line 17). The alignment is computed using standard methods, see §II, or reused, if a trace of the same variant was handled earlier.

Deviation contexts. Next, using the alignment, we determine the contexts, modelled as a set of events C of trace ξ, in which non-conformance is observed (lines 18-21). For each step of the alignment that indicates non-conformance, the respective event and preceding events that denote the context of the deviation are collected. The precise definition of the context is modelled by the function $\psi_{context}$, discussed below.

Maintaining the knowledge base. The deviation context C, indicates whether the trace is non-conforming at all and, if so, which events can be linked to the non-conformance. As such, it is used to update the knowledge base (lines 22-36). For each feature, trace-level or event-level, the counters that indicate the presence or absence of the feature for the conformance result, conformance or deviation, are incremented accordingly. Finally, using these counters, the correlation coefficients are updated for all features (line 37) Specifically, we compute the Phi Coefficient, defined as:

$$\varphi(K^+, K^-, K_d^+, K_d^-) = \frac{K^+K_d^- - K^-K_d^+}{\sqrt{(K^+ + K_d^+)(K^- + K_d^-)(K^+ + K^-)(K_d^+ + K_d^-)}}.$$

B. Instantiations

To employ our sampling procedure, a set of features has to be selected. As discussed above, the selection of these features goes along with the use of the indices to select traces as part of the exploitation phase (line 14 in Alg. 2). Specifically, the feature index may be used directly to guide the selection of trace, or it may be combined with similarity-based hashing for trace-level behavioural features. Against this background, we provide two specific instantiations of the sampling procedure:

Feature-only. We employ all event-level and trace-level features, i.e., $F = F_e \cup F_t$ with $F_t = F_t^a \cup F_t^b$. In the exploitation phase, once a feature f is drawn based on the probability distribution over the positively correlated features, this feature is used directly: The trace ξ is drawn randomly from $I(f) \setminus L'$, i.e., from the set of remaining traces showing feature f (thereby realizing the function of line 14 in Alg. 2).

Behavioural. The second instance focuses on the behavioural features and, to avoid issues stemming from index sparsity, also leverages similarity-based hashing. As such, the set of features is given as $F = F_t^b$. Moreover, given the feature f determined from the set of positively correlated features, we first randomly draw an auxiliary trace ξ' from $I(f) \cap L'$, i.e., from the set of already sampled traces showing feature f. Then, the actual trace ξ to add to the sample is drawn randomly from $J(\xi') \setminus L'$, i.e., from the set of remaining traces that are similar to trace ξ' (which, again, realizes line 14 in Alg. 2)

C. Further Design Choices

While the above instantiations provide fundamentally different approaches to guided sampling, our algorithm involves further design choices. We reflect on these in the remainder.

Termination predicate ψ_{stop}. To determine when to stop the construction of a sample, the predicate may check the size of the sample against a threshold. A more dynamic approach assesses the convergence of the knowledge base, i.e., providing a bound for the change in the correlation coefficients observed in several, subsequent iterations of selecting a trace.

Exploration predicate $\psi_{explore}$. To balance between exploration and exploitation, a simple solution is a static threshold

on the sample size, e.g., exploration for the first k traces, and exploitation afterwards. Dynamic options are a random decision based on a learning rate (increasing the probability of exploitation over time), or the presence or absence of positive correlations of features with conformance results (moving to exploitation only once positive correlations have been found).

Context function $\psi_{context}$. For an alignment step that indicates non-conformance, different strategies to select events that denote the context of the deviation may be used. A greedy approach considers all preceding events in the trace. However, a more narrow context is achieved by considering solely a window of preceding events (e.g., the k previous events). Also, preceding events may be selected probabilistically, using a fall-off factor that decreases the selection probability of events that are more distant from the detected deviation in the trace.

VI. Evaluation

We evaluated our proposed guided sampling procedure, to answer the following questions:

RQ1 Are our sampling procedures able to reliably sample more traces containing non-conforming behaviour, compared to random sampling or static relevance assumptions?

RQ2 Is the conformance analysis based on guided samples still more efficient than using the complete log?

RQ1 is concerned with the quality of samples generated using the proposed approach, which we evaluated using two measures. First, we compared the fraction of sampled traces that contained non-conforming behaviour. As a second measure, we assessed the reliability of the exploitation phase by measuring how fast the knowledge base converges to a stable state.

We expect our approach to sample more deviations over all used datasets. Furthermore, we expect, that reliable correlation coefficients can be obtained based on a few dozens of traces.

By answering RQ2, we ensure that our approach is justified from the viewpoint of computational efficiency.

We implemented our sampling techniques in Python, using PM4Py [9]. The implementation and evaluation results are publicly available.[2] We relied on three public event logs, namely *BPI-Challenge 2012* [10], *Sepsis Cases* [11], and *BPI-Challenge 2018* [12]. *BPI-2012* is a log of an insurance claim handling process, consisting of 13087 trace and 4366 trace variants. In contrast, *Sepsis Cases*, which contains recorded trajectories of hospital patients, only consists of 1050 traces and 846 trace variants. Finally, *BPI-2018* contains 43809 process instances and 28457 trace variants describing the processing for financial funding in the agriculture sector. To ensure the timely termination of all evaluation scenarios, we removed the longest 5% of traces from *BPI-2018*. For each event log, we applied the Inductive Miner algorithm [13] with noise threshold of 20% to discover a process model used for checking conformance. We instantiated the two proposed sampling procedures, as well as an approach based on random sampling, and one using a static hypothesis, i.e., sampling the k longest trace variants. For each of the instantiations, we sampled 100, 200, 300, 400, and 500 traces in ten repetitions, i.e, ψ_{stop} was set as a static sample size. For $\psi_{explore}$, we used a static random selection of either exploration in 20% of the traces, or exploitation in 80% of the traces and for $\psi_{context}$ we considered the 3 preceeding events (including the deviating activity) of a deviation as its deviation contexts. For the three datasets, we considered 2 (*BPI-2012*), 29 (*Sepsis Cases*) and 5 (*BPI-2018*) event-features. Additionally, for *BPI-2012*, we considered one trace-feature, and for each dataset k-grams of lengths 3. In the following, we first report on the qualitative aspects in §VI-A, followed by a discussion of the efficiency in §VI-B

[2] see https://github.com/MartinKabierski/Guided_Conformance_Sampling

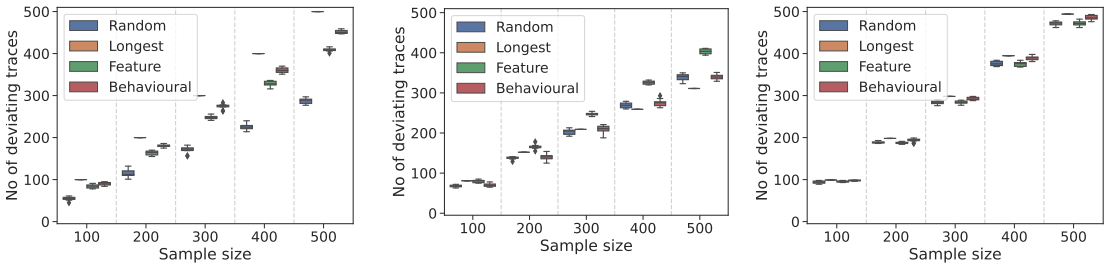
A. Qualitative Evaluation

Fraction of deviating traces. In Fig. 3, the number of traces containing non-conforming behaviour in the sample is illustrated. While the extent differs over datasets, both feature-based and behaviour-based sampling yield more deviating traces than random sampling. For *BPI-2012*, where this difference is most notable, for sample sizes of 100 traces, the random approach samples 54.7 deviating traces on average, whereas the feature-based and behaviour-based approaches select 83.9 and 90.2 traces on average, respectively. This difference increases with increasing sample sizes. For sample size 500, the random approach draws 287.1 deviating traces, while the guided approaches yield 409 and 451.7 traces on average.

For *Sepsis Cases*, there are minor differences between the random sampling and behaviour-based sampling (67.6 vs. 70.1 for sample size 100, up to 338.6 vs. 339.5 for sample size 500). Feature-based sampling performs notably better, e.g., 79.4 for sample size 100, up to 403.3 for sample size 500. As the log is very unstructured, behaviour-based sampling is not able to reliably determine exploitable traces based on n-grams.

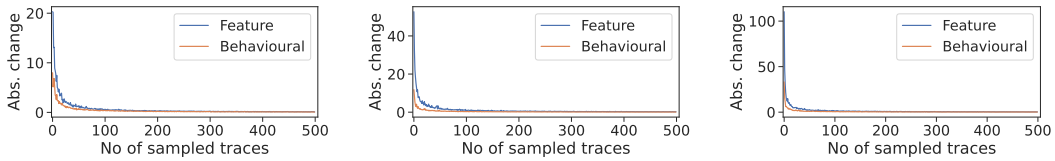
For *BPI-2018*, we note small, but notable differences in the sampled traces for large sample sizes. For sample size 500, random sampling selects 471.5 deviating traces, similar to the 471.7 for the feature-based approach, both being lower than the 485.8 traces obtained by behaviour-based sampling. Here, the large index size with 3174 exploitable keys does not allow for a fine-grained differentiation of traces. For the static relevance assumption that deviations correlate with the longest trace variants, we obtain mixed results. While this hypothesis only samples deviating trace variants for *BPI-2012*, and with 494 deviating traces, also the most for *BPI-2018*, the opposite holds for *Sepsis Cases*. Here, sampling using this hypothesis gives us only 311 deviating traces, which is even worse than random sampling. This emphasizes that static assumptions on the relevance of traces for a given conformance checking setting cannot be expected to work in the general case.

Knowledge base convergence. Next, we turn to controlled experiments, in which we measured how fast the knowledge base stabilizes in terms of the correlation coefficients. This way, we assessed the stability and reliability of the exploitation phase. We measured for both guided sampling approaches the sum of absolute changes over all feature correlation coefficients for increasing sample sizes. In Fig. 4, the mean of these changes over ten repetitions is illustrated. We see, that for all three input



(a) BPI-2012 (b) Sepsis Cases (c) BPI-2018

Figure 3: Fraction of traces containing deviations for different sample sizes



(a) BPI-2012 (b) Sepsis Cases (c) BPI-2018

Figure 4: Convergence of the knowledge base for increasing sample sizes

Size	BPI-2012	Sepsis Cases	BPI-2018
100	14.36	0.36	> 180 (15.39h)
200	31.86	0.88	> 180 (30.78h)
300	47.62	1.39	> 180 (46.17h)
400	68.77	1.77	> 180 (61.00h)
500	129.54	2.11	> 180 (76.95h)
complete	> 180(56.5h)	4.19	> 180 (280 days)

Table I: Mean runtimes for alignment construction (in min)

sets, starting form an initial phase with many changes, both approaches stabilize relatively fast. Due to the larger index sizes of the feature-only approach, the knowledge base converges a bit slower, than the behaviour-based approach. Yet, in both cases, the knowledge bases stabilize before a sample size of 200, i.e., stable, exploitable correlations are found quickly. Also, for all datasets, a positive, and thus exploitable, correlation, occurred consistently during the first ten traces, for each of the three datasets. Thus, the procedure was able to transition from exploration to exploitation rather fast.

B. Efficiency Evaluation

Finally, we measured the mean runtime for different phases of the sampling procedures, i.e., the *alignment construction time*, the *sampling time*, and for the guided approaches also the *indexing time*. We expect, that, due to the introduced overhead in the indexing and sampling phase, the guided approaches run longer than random sampling, or sampling using a static relevance assumption. Yet, we expect the overhead to be negligible compared to the runtime of alignment computation.

Table I lists the mean times for the construction of alignments for random samples of different sizes. Where it was possible to obtain results on the complete dataset in <180 min, we report the mean time of this analysis. If the process took longer, we omit the results and provide an estimate of the expected time. With this bound, analysis of the full event log completed only

	Size	Indexing		Sampling			
		Feature	Behav.	Feature	Behav.	Random	Longest
BPI-2012	100	4.47	1.26	<1	<1	<1	<1
	200			<1	<1	<1	<1
	300			<1	<1	<1	<1
	400			1.03	<1	<1	<1
	500			1.29	<1	<1	<1
Sepsis Cases	100	0.85	0.11	<1	<1	<1	<1
	200			1.33	<1	<1	<1
	300			2.16	<1	<1	<1
	400			2.71	<1	<1	<1
	500			3.28	<1	<1	<1
BPI-2018	100	105.79	10.42	1.72	<1	<1	<1
	200			3.32	1.57	<1	<1
	300			4.99	2.42	<1	<1
	400			6.62	3.25	<1	<1
	500			8.02	4.03	<1	1.14

Table II: Breakdown of mean runtimes (in s)

for *Sepsis Cases*, where the analysis took 4.19 min on average. For *BPI-2012*, we were able to compute alignments for the random samples in 14.36 min on average for sample size 100, up to 129.54 min on average for samples of size 500. For *BPI-2018*, the runtimes were even worse and the analysis did not complete within 180 min even for the smallest sample.

Table II lists the mean runtimes of the sampling strategies. In contrast to Table I, results here are reported in seconds. The overhead of index creation turns out to be miniscule, compared to the runtimes of alignment construction. The feature index consisted of 609, 828, and 3174 keys for *BPI-2012*, *Sepsis Cases*, and *BPI-2018*, respectively. For the latter, k-gram features are most frequent, as, they yield roughly 2000 index keys, which explains the relatively long indexing phase.

Likewise, the mean sampling overhead was consistently below 10s. As expected, guided sampling took longer than the baseline approaches, which mostly stay below 1s. Since the feature-based approach considers more features during

exploitation, the respective runtimes were the highest. However, contrasting the absolute runtimes of indexing and sampling that for alignment computation, we conclude that the overhead of guided sampling is negligible. As such, guided sampling may increase the result quality of a conformance checking task, without noticeably increasing the runtime.

VII. Related Work

For conformance checking, alignments [7] have emerged as the de-facto standard. To increase the efficiency of alignment computation, decomposition schemes [14], effective search heuristics [15], specific problem encodings [16], or dedicated data structures [17] may be employed. Also, algorithms to approximate alignments are available, e.g., based on relaxation labelling [18] or the edit distance to known alignments [4].

Furthermore, the use of a sample of an event log has been studied. To this end, an incremental scheme for random sampling of an event log was proposed [5], [4]. It gives statistical guarantees for aggregated conformance measures, for which all traces are equally relevant. Similarly, the behaviour of a process model may be sampled to simplify the computation of alignments, under various hypotheses for the relevance of the model's execution sequences [19], [20]. In the context of process discovery, the effect of further static relevance assumptions on the result quality was shown empirically in [6].

Since the quality of an analysis result depends on the representativeness of the selected sample, it is important to relate input specifics to the output quality. Independent of a specific analysis task, it was suggested to assess the representativeness of a sample of an event log in terms of over-and undersampled behaviour [21]. Recently, the characteristics of event logs have also been linked to the result quality of process discovery algorithms [22]. Yet, apart from the aforementioned work that is limited to aggregated conformance measures [5], [4], our approach is the first to link the characteristics of an event log with conformance results in terms of deviating traces.

Balancing the trade-off between exploration and exploitation is a common problem that is faced, e.g., in evolutionary algorithms. Various strategies may manage this trade-off [24], with auto-tuning [23] being a prominent example.

VIII. Conclusion

In this paper, we proposed a strategy to guide the construction of a sample of an event log with the intention of increasing the fraction of deviating traces. Our approach learns correlations of characteristics of traces and deviations in constructed alignments, and, based thereon, guides the sampling of subsequent traces. We instantiate the framework using two approaches, one directly leveraging trace-level and event-level features, and one combining trace-level features with similarity-based hashing. Experimental results with several public event logs indicate that our relevance-guided sampling strategy reliably selects more deviating traces than random sampling and provides more coherent results compared to static assumptions on the relevance of traces. We achieve this with a negligible runtime overhead that is dominated by the alignment construction time.

Acknowledgements. This work was partially funded by the German Research Foundation (DFG), project 421921612.

References

[1] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking - Relating Processes and Models*. Springer, 2018.
[2] F. Caron, J. Vanthienen, and B. Baesens, "Comprehensive rule-based compliance checking and risk management with process mining," *Decis. Support Syst.* 54, no. 3, pp. 1357–1369, 2013.
[3] M. Jans, M. G. Alles, and M. A. Vasarhelyi, "The case for process mining in auditing: Sources of value added and areas of application," *Int. J. Account. Inf. Syst.* 14, no. 1, pp. 1–20, 2013.
[4] M. Bauer, H. van der Aa, and M. Weidlich, "Sampling and approximation techniques for efficient process conformance checking," *Inf. Syst.*, 2020.
[5] M. Bauer, H. van der Aa, and M. Weidlich, "Estimating process conformance by trace sampling and result approximation," in *BPM*, LNCS 11675. Springer, 2019, pp. 179–197.
[6] M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst, "The impact of biased sampling of event logs on the performance of process discovery," *Computing* 103, no. 6, pp. 1085–1104, 2021.
[7] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC*. IEEE Computer Society, 2011, pp. 55–64.
[8] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive data sets*. Cambridge university press, 2020.
[9] A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, "Process Mining for Python (PM4Py) : Bridging the Gap Between Process- and Data Science," in *ICPM Demos*, CEUR 2374, 2019, pp. 13–16.
[10] B. van Dongen, "Bpi challenge 2012," 2012. https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204/1
[11] F. Mannhardt, "Sepsis cases - event log," 2016. https://data.4tu.nl/articles/dataset/Sepsis_Cases_-_Event_Log/12707639/1
[12] B. van Dongen and F. F. Borchert, "Bpi challenge 2018," 2018. https://data.4tu.nl/articles/dataset/BPI_Challenge_2018/12688355/1
[13] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *BPM Workshops*, LNBIP 171. Springer, 2013, pp. 66–78.
[14] J. Munoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Single-entry single-exit decomposed conformance checking," *Inf. Syst.* 46, pp. 102–122, 2014.
[15] A. Syamsiyah and B. F. van Dongen, "Improving alignment computation using model-based preprocessing," in *ICPM*. IEEE, 2019, pp. 73–80.
[16] M. Boltenhagen, T. Chatain, and J. Carmona, "Optimized SAT encoding of conformance checking artefacts," *Computing* 103, no. 1, pp. 29–50, 2021.
[17] D. Reißner, A. Armas-Cervantes, R. Conforti, M. Dumas, D. Fahland, and M. L. Rosa, "Scalable alignment of process models and event logs: An approach based on automata and s-components," *Inf. Syst.* 94, p. 101561, 2020.
[18] L. Padró and J. Carmona, "Approximate computation of alignments of business processes through relaxation labelling," in *BPM*, LNCS 11675. Springer, 2019, pp. 250–267.
[19] M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst, "Conformance checking approximation using subset selection and edit distance," in *CAiSE*, LNCS 12127. Springer, 2020, pp. 234–251.
[20] M. F. Sani, J. J. G. Gonzalez, S. J. van Zelst, and W. M. P. van der Aalst, "Conformance checking approximation using simulation," in *ICPM*. IEEE, 2020, pp. 105–112.
[21] B. Knols and J. M. E. M. van der Werf, "Measuring the behavioral quality of log sampling," in *ICPM*. IEEE, 2019, pp. 97–104.
[22] J. M. E. M. van der Werf, A. Polyvyanyy, B. R. van Wensveen, M. Brinkhuis, and H. A. Reijers, "All that glitters is not gold - towards process discovery techniques with guarantees," in *CAiSE*, LNCS 12751. Springer, 2021, pp. 141–157.
[23] L. Lin and M. Gen, "Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation," *Soft Comput.* 13, no. 2, pp. 157–168, 2009.
[24] M. Crepinsek, S. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.* 45, no. 3, pp. 35:1–35:33, 2013.