Sampling and Approximation Techniques for Efficient Process Conformance Checking

Martin Bauer[a], Han van der Aa[b], Matthias Weidlich[a]

[a]*Department of Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany*
[b]*Data and Web Science Group, University of Mannheim, Germany*

Abstract

Conformance checking enables organizations to automatically assess whether their business processes are executed according to their specification. State-of-the-art conformance checking algorithms perform this task by establishing alignments between behaviour recorded by IT systems to a process model capturing desired behaviour. While such alignments clearly highlight conformance issues, a major downside is that these algorithms scale exponentially in the size of both the event data, capturing recorded behaviour, and the process model used as input. At the same time, it is crucial to recognise that event data used for such analyses typically only relates to a specific interval of process execution rather than the entire history, meaning that the employed event data is inherently incomplete. Therefore, we argue that statistical methods allow one to obtain a proper understanding of the overall conformance of a process by considering only a fraction of the available data. In this paper, we therefore present a statistical approach to conformance checking that employs trace sampling and result approximation in order to derive conformance results in an efficient manner. The approach reduces the runtime significantly, while still providing guarantees on the accuracy of the estimated conformance result. We instantiate the general approach for different measures of the overall conformance of an event log and a process model, including fitness as a direct quantification of conformance as well as the distribution of deviations over activities and deviations related to contextual factors, such as the involved resources. Moreover, to increase the robustness of our approach, we elaborate on mechanisms to reveal biases in sampling procedures. Experiments with real-world and synthetic datasets show that our approach speeds up state-of-the-art conformance checking algorithms by up to three orders of magnitude, while largely maintaining the analysis accuracy.

Keywords: Conformance Checking, Trace sampling, Result Approximation

1. Introduction

The execution of business processes is often supported by information systems [1]. The way these systems shall support the process is then captured by a process model, which defines the elementary units of work, known as activities of the process, along with causal dependencies for their execution. A process model, therefore, provides a specification against which the process execution as it materialises in terms of event data may be assessed. The comparison of the modelled behaviour of a system with its recorded behaviour is known as conformance checking [2]. Once differences between the model of an information system and its actual execution are detected, they may be interpreted, analysed, and potentially compensated, thereby leading to more consistent understanding of how the system supports the process. Yet, it is important to see that conformance analysis is not a one-off operation, but a continuous process that acknowledges the frequent changes applied to business processes and the supporting information systems [3].

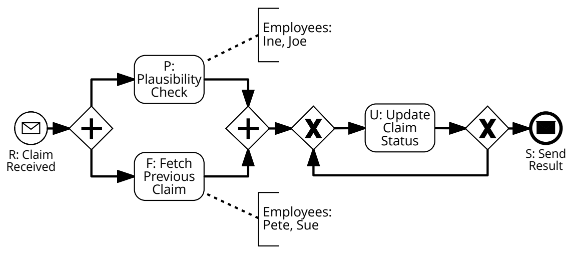
These days, information systems tend to produce increasing amounts of event data. Drivers for this trend are wide-spread process automation and large-scale instrumentation of process resources. As a consequence, event logs of process executions may contain up to billions of events [4], which imposes computational challenges for conformance checking since state-of-the-art, alignment-based techniques [5] show an exponential run-time complexity. In recent years, various techniques for efficient conformance checking have been proposed that exploit

Email addresses: martin.bauer@hu-berlin.de (Martin Bauer), han@informatik.uni-mannheim.de (Han van der Aa), matthias.weidlich@hu-berlin.de (Matthias Weidlich)

search-based methods [6, 7], planning algorithms [8], and distributed computing [9, 10]. Also, it was suggested to compromise correctness and to approximate conformance results. Examples for such techniques include notions of approximate alignments [11] as well as divide-and-conquer schemes for the computation of conformance results [12, 13, 14, 15]. Interestingly, techniques that strive for a general, but potentially not entirely precise understanding of the conformance of an event log and a process model, so far, considered mostly the adopted algorithms. Fundamentally, they still require the consideration of *all*, possibly billions, of recorded events.

The above observation has to be seen in light of common application scenarios of conformance checking, where an event log is extracted for a certain interval of process execution. In most cases, the event log does not cover the entire history of process execution, which renders it inherently incomplete [16, 17, 18]. Moreover, conformance checking may be conducted for different purposes and, hence, assume different levels of granularity. For instance, if conformance checking is driven by compliance requirements, it focuses on the perspective of individual cases and carves out single deviations related to a specific execution of the process. However, if conformance checking is used to achieve operational excellence, e.g., based on reference models and best practices, aggregated results are derived to provide a general understanding of the overall conformance of the process. Here, the quantification and the identification of hot-spots of non-conformance help to guide efforts for process improvement, re-design, and the repair of process models. Different notions for such an aggregated view on process conformance may be adopted, though. Examples include fitness measures [5], the identification of activities that are frequently part of conformance violations [12], or contextual aspects of non-conformance, such as the resources that are typically involved.

In this paper, we argue that for a general understanding of the overall conformance, it is sufficient to compute aggregated conformance results for only a small fraction of an event log. As mentioned above, an event log is typically incomplete by definition. Hence, minor differences between the conformance results obtained for the whole log and a partial log may be attributed to the inherent uncertainty of the conformance checking setting. As an example, consider a simple claim handling process as visualised in Fig. 1. Here, the events recorded for the first case (ignoring the information on the involved resources) indicate non-conformance, as a previous claim is fetched twice (f). Considering also the second case, the aggregated conformance results are the



(a)

Case	Event Sequence	Resources for F (Fetching Previous Claims)
1	r p f f u s	*Sue*, *Pete*
2	r f p u f s	*Ine*, *Ine*
3	r p f u u u s	*Sue*
4	r p f f ⊥ s	*Pete*, *Pete*

(b)

Figure 1: (a) Model of a claim handling process; (b) events of four process executions.

same, though, despite the different sequence of events: There is one deviation per trace on average and the set of non-conforming activities contains a single activity ({F}). Considering also the third and fourth case, new information on the overall conformance of process execution is obtained. Yet, the fourth case resembles the first one. Hence, its conformance (two deviations w.r.t. the model) may be approximated based on the result of the first case (one deviation) and the difference between both event sequences (one event differs).

The example illustrates two complimentary angles that can be followed to avoid computation of conformance results for all available data. First, conformance may be assessed based on *trace sampling*. Selecting random traces iteratively from an event log, for each trace, it is assessed whether it yields new information on the overall conformance. In this paper, we formulate this approach as a series of binomial experiments, each experiment being a random trace selection. Based thereon, we establish bounds on the error of the conformance result derived from a partial event log, in comparison to the whole log. A second angle for efficient conformance checking is to include *result approximation* for the impact of an individual trace on the overall conformance. In this paper, we show how a worst-case approximation of the implications of a single trace can be assessed to further reduce the amount of data for which conformance results are actually computed.

Our main contributions can be summarized as follows:

- We present a general framework for conformance checking based on trace sampling. We instantiate this framework for three types of conformance results: fitness, deviation distributions, and contextual deviations. The latter is exemplified for the context being defined by the involved resources.

- We demonstrate how conformance results can be approximated, which avoids the need to compute alignments even for certain previously unseen trace variants. We present such approximations for all the aforementioned types of conformance results.
- We show how the quality of the samples on which conformance results are based can be assured. Specifically, we propose methods to assess the internal and external quality of a sample based on behavioural representativeness and data attribute coverage and incorporate these into the proposed conformance checking techniques.

This paper is an extended and revised version of our earlier work on process conformance estimation [19]. The main conceptual extensions in this work are: (1) the consideration of context-based conformance measures in trace sampling, (2) an improved approximation technique when considering deviation distributions, and (3) methods to assess the quality of utilized log samples. Each of these extensions is assessed through additional evaluation experiments.

The remainder is structured as follows. Section 2 provides essential preliminaries. Section 3 presents a general framework and its instantiations for sample-based conformance checking, whereas Section 4 considers approximation-based methods. Section 5 considers the assessment of sample quality. Section 6 presents comprehensive evaluation experiments using both real-world and synthetic datasets conducted to assess the efficiency, accuracy, and sensitivity of the proposed conformance checking techniques. Finally, we review our contributions in light of related work (Section 7) before concluding in Section 8.

2. Preliminaries

This section provides essential definitions regarding events and event logs, process models, and alignment-based conformance checking.

Events and event logs. We adopt an event model that builds upon a finite set of activities $\mathcal{A}$. An event recorded by an information system is assumed to be related to the execution of one of these activities. By $\mathcal{E}$, we denote the universe of all events. A single execution of a process, called a *trace*, is modelled as a sequence of events $\xi \in \mathcal{E}^*$, such that no event can occur in more than one trace. An event log is a set of traces, $L \subseteq 2^{\mathcal{E}^*}$. Our example in Fig. 1b defines four traces. While each event is unique, we represent them with small letters $\{r, p, f, u, s\}$, that indicate for which activity of the process model, denoted by respective capital letters $\{R, P, F, U, S\}$, the execution is signalled. For an event $e \in \mathcal{E}$, we also write $e.\texttt{activity} \in \mathcal{A}$ to denote the activity of an event. Two distinct traces that indicate the same sequence of activity executions are of the same *trace variant*.

Each event furthermore carries information on its execution context, such as the data consumed or produced during the execution of an activity. This payload is defined by a set of data attributes $\mathcal{D} = \{D_1, \ldots, D_p\}$ with $dom(D_i)$ as the domain of attribute D_i $(1 \leq i \leq p)$. We write for $e.D$ the value of attribute D for an event e. As an illustration, an event of type R (*Claim Received*) may be associated with an `amount` attribute that reflects the claimed compensation, e.g., $e.\texttt{amount} = 1.000$. Moreover, for an event of type F (*Fetch Previous Claim*), an attribute `resource` indicates the employee that accessed information about prior claims, e.g., $e.\texttt{resource} = Pete$.

Process models. First and foremost, a process model defines the control-flow of a process, i.e., the execution dependencies between the activities of a process. For our purposes, it is sufficient to abstract from specific process modelling languages and focus on the set of execution sequences defined by a model. It is denoted by $M \subseteq \mathcal{A}^*$ and captures the sequences of activity executions that lead the process to its final state. For instance, the model in Fig. 1a defines the execution sequences $\langle R, P, F, U, S\rangle$ and $\langle R, F, P, U, S\rangle$, potentially including additional repetitions of U.

Process models may be enriched with information on further process perspectives that formalise in more detail the context of process execution. Examples include branching-conditions that refer to the data of a case (e.g., claims of a certain amount are processed differently), temporal constraints on the execution of activities (e.g., results for a claim are sent out solely on working days, or the result needs to be sent at most three weeks after the claim has been received), or resource assignments (e.g., the execution of certain activities is limited to authorized employees). To ease the presentation in the remainder of the paper, we focus on resource assignments as one particular example of such contextual information. To this end, we define responsibilities for the execution of activities by assigning each of them a set of resources. Formally, with $\mathcal{R}$ as the set of resources, the model includes a function $M_{res} : \mathcal{A} \rightarrow 2^{\mathcal{R}}$. Referring to the above example, this function would, for instance, define that there are two out of four employees that are responsible for fetching previous claims, i.e., the resources include $\mathcal{R} = \{Pete, Joe, Sue, Ine\}$ while $M_{res}(F) = \{Pete, Sue\}$.

In the remainder, we write $\mathcal{M}$ for the set of all process models, assuming that it includes the definition of the execution sequences and, potentially, further information on other process perspectives.

Alignments. State-of-the-art techniques for conformance checking construct *alignments* between traces and execution sequences of a model to detect deviations [5, 13]. An alignment between a trace ξ and a model M, denoted by $\sigma(\xi, M)$ in the remainder, is a sequence of steps, each step comprising a pair of an event and an activity, or a *skip* symbol $\perp$, if an event or activity is without counterpart. For instance, for the non-conforming trace ξ_1 (case 1 from Fig. 1b), an alignment is constructed as follows:

Trace ξ_1	r	p	f	f	u	s
Execution sequence	R	P	F	$\perp$	U	S

Assigning costs to skip steps, a cost-optimal alignment is constructed for a trace in relation to all execution sequences of a model [5]. Note though, that an optimal alignment is not necessarily unique, i.e., for a single trace, there may be multiple cost-optimal alignments with the same execution sequence or with other execution sequences of the model.

While alignments enable conclusions on the conformance of an individual trace, they also form the basis of a large range of measures for the overall conformance of a log. The respective measures may refer to different process perspectives, such as the control-flow, the data consumed and produced by activities, temporal aspects, and resource assignments. Moreover, they may employ various levels of aggregation; from a single numeric value for the whole log to a distribution for each of the activities of the process under consideration. In the remainder, we consider three measures to illustrate the spectrum of conformance checking and highlight that our approach is largely independent of a specific measure.

Fitness. The *fitness* of a log with respect to a given model provides a quantification of the control-flow conformance. With an optimal alignment $\sigma(\xi, M)$ for each trace ξ, we denote the aggregated alignment cost by $c(\xi, M)$. Then, a common fitness measure is defined as the ratio of this cost to the maximum possible cost per trace, i.e., the sum of the costs of aligning a trace with an empty model, $c(\xi, \emptyset)$, and the minimal costs of aligning an empty trace with the model, $\min_{x \in M} c(\langle\rangle, \{x\})$. Aggregating the respective costs over all traces, the fitness measure returns a single numeric value for the control-flow conformance of a log:

$$\text{fitness}(L, M) = 1 - \frac{\sum_{\xi \in L} c(\xi, M)}{\sum_{\xi \in L} c(\xi, \emptyset) + |L| \times \min_{x \in M} c(\langle\rangle, \{x\})} \quad (1)$$

Using a standard cost function (all skip steps have equal costs), the fitness value of the example log $\{\xi_1, \xi_2, \xi_3, \xi_4\}$ in Fig. 1b is 0.9.

Deviation distribution. As a second conformance measure, we consider the *deviation distribution* of a log that enables the detection of hotspots of non-conformance in the control-flow. It captures the relative frequency with which an activity (not to be confused with a task of a process model) is part of a conformance violation. For a log L and a model M, this distribution follows from skip steps in the optimal alignments of all traces. It is formalised based on a bag of activities, which is captured as a function coined $dev(L, M)$ that assigns multiplicities to activities, i.e., its signature is given as $dev(L, M) : \mathcal{A} \rightarrow \mathbb{N}_0$. Note that multiple skip steps may relate to a single activity even in the alignment of a single trace. In the remainder, we use subscript notation for bag cardinalities, i.e., for a bag X over $\mathcal{A}$, we write $X = [a_1^{n_1}, \ldots, a_m^{n_m}]$ with $X(a_i) = n_i$ and $n_i > 0$, for $1 \leq i \leq |\mathcal{A}|$. The relative deviation frequency of an activity $a \in \mathcal{A}$ is then obtained by dividing the number of occurrences of a in the bag of deviations by the total number of deviations:

$$f_{dev(L,M)}(a) = \frac{dev(L, M)(a)}{\sum_{a \in \mathcal{A}} dev(L, M)(a)} \quad (2)$$

Consider again the example in Fig. 1 and assume that skip steps relate to the highlighted trace positions. Then, we derive that $dev(\{\xi_1, \xi_2, \xi_3, \xi_4\}, M) = [F^3, U]$ and $f_{dev(\{\xi_1,\ldots,\xi_4\},M)}(F) = 3/4$, so that the fetching of a previous claim (F) is identified as a hotspot of non-conformance.

Contextual deviations. As a third measure, we go beyond the control-flow perspective and consider conformance with respect to contextual factors. That is, we assess whether the context definition as given in the process model (e.g., data-based branching conditions, temporal constraints, resource assignments) is consistent with the context information recorded in the payload of the events in an event log.

Let $\mathcal{F}$ be a universe of contextual factors. Contextual deviations as considered here, may be captured by a function $s_{fac}(L, M_{fac})$ that assigns these factors to activities, given a log L and model M_{fac} that is annotated with contextual information. That is, the signature of this function, in general, is given as $s_{fac}(L, M_{fac}) : \mathcal{A} \to 2^{\mathcal{F}}$.

As mentioned, we illustrate the notion of a contextual deviation with the example of the resources involved in the execution of a process, which leads to the notion of a resource deviation. On the one hand, this includes violations where employees executed activities, which, according to the process model, should not have been executed, i.e., as identified in an alignment. On the other hand, it includes cases where the responsibilities for activities, defined in a process model, where violated, i.e., activities performed by unauthorized resources.

To define this conformance measure, we use $dev_r(e, M_{res}) = \{e.\texttt{resource}\} \setminus M_{res}(e.\texttt{activity})$ to denote a set that contains the resource that was unauthorized to perform event e, i.e., if $e.\texttt{resource}$ is not part of the possible assignments defined by the model for the activity, $M_{res}(e.\texttt{activity})$. If the resource was authorized, the set $dev_r(e, M_{res})$ is empty. Then, we aggregate the resource deviations per activity, leading to the set of employees involved in non-conforming activity executions:

$$s_{res(L,M_{res})}(a) = \bigcup_{\substack{\xi=\langle e_1,\ldots,e_n\rangle\in L\\ \langle(e_1,t_1),\ldots,(e_m,t_m)\rangle\in\sigma(\xi,M)\\ 1\leq i\leq m\\ e_i.\texttt{activity}=a\ \wedge\ t_i=\perp}} \{e_i.\texttt{resource}\} \ \cup \bigcup_{\substack{\xi=\langle e_1,\ldots,e_n\rangle\in L\\ 1\leq i\leq n\\ e_i.\texttt{activity}=a}} dev_r(e_i, M_{res}) \quad (3)$$

In Eq. 3, the left-hand side of the join identifies those resources associated with non-conforming event executions (i.e., skip steps), whereas the right-hand side captures the aforementioned cases involving unauthorized resources.

Assuming that, as illustrated in Fig. 1, the employees involved in the fetching of previous claims had been Sue and $Pete$ for the first and second activity execution in ξ_1, respectively; Ine for both activity executions in ξ_2; Sue for ξ_3; and $Pete$ for both activity executions in ξ_4. Then, we would obtain a set of resource deviations as $s_{res(L,M_{res})}(F) = \{Pete, Ine\}$ highlighting that the two employees were involved in fetching of previous claims that was not in line with the process model.

The above formalisation illustrates contextual deviations being defined by the resources involved in non-conforming execution of activities. Of course, the notion of resource deviations may be elevated to a higher level of detail, allowing the identification of resource roles involved in violations rather than individuals. Yet, in the same vein, contextual deviations may also be defined for other perspectives of process executions. For instance, if certain activities must not be executed for claims of a certain amount, contextual deviations may be captured for pairs of activities and amounts (potentially after applying some predicate abstraction to the latter). If the execution of some activities is limited to working days, extracting the day-of-week from the events, yields insights on the activities for which these constraints are violated.

3. Sample-Based Conformance Checking

This section describes how trace sampling can be used to improve the efficiency of conformance checking. The general idea is that it often suffices to only compute alignments for a subset of all trace variants to gain insights into the overall conformance of a log to a model. However, we randomly sample an event log trace by trace, not by trace variant, which avoids to load the entire log and step-wise reveals the distribution of traces among the variants. At some point, though, the sampled traces then do not provide new information on the overall conformance of the process.

In the remainder of this section, we first describe a general framework for sample-based conformance checking (Section 3.1), which we then instantiate for different types of conformance results: the overall fitness of log (Section 3.2), its deviation distribution (Section 3.3), and observed contextual deviations related to resource assignments (Section 3.4)

3.1. Statistical Sampling Framework

To operationalise sample-based conformance checking, we regard it as a series of binomial experiments. In this, we follow a log sampling technique introduced in the context of process discovery [20, 21] and lift it to the setting of conformance checking.

Information novelty. When parsing a log trace by trace, some traces may turn out to provide information on the conformance of a log to a model that is similar or equivalent to the information provided by previously encountered traces. To assess whether this is the case, we capture the conformance information associated with a log by a *conformance function* $\psi : 2^{\mathcal{E}^*} \times \mathcal{M} \to \mathcal{X}$. That is, $\psi(L, M)$ is the conformance result (of some domain $\mathcal{X}$) between a log L and a model M. If we are interested in the fitness of the log, ψ returns the fitness value; for the distribution of deviations, ψ provides information on the model activities for which deviations are observed; and for context information in terms of assigned resources, ψ captures the sets of employees who executed an activity not according to the process model, i.e., at the wrong position in the trace or despite being not responsible for the activity.

Based thereon, we define a random Boolean predicate $\gamma(L', \xi, M)$ that captures whether a trace $\xi \in \mathcal{E}^*$ provides new information on the conformance with model M, i.e., whether it changes the result obtained already for a set of previously observed traces $L' \subseteq 2^{\mathcal{E}^*}$. Assuming that the distance between conformance results can be quantified by a function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^+$, we define a *new information predicate* as:

$$\gamma(L', \xi, M) \Leftrightarrow d(\psi(L', M), \psi(\{\xi\} \cup L', M)) > \epsilon. \tag{4}$$

Here, $\epsilon \in \mathbb{R}_0^+$ is a relaxation parameter. If incorporating trace ξ changes the conformance result by more than ϵ, then it adds new information over L'.

Framework. We exploit the notion of information novelty for hypothesis testing when sampling traces randomly from an event log L. We determine when *enough* sampled traces have been included in a log $L' \subseteq L$ to derive an understanding of its overall conformance to a model M. Following the interpretation of log sampling as a series of binomial experiments [21], L' is regarded as sufficient if the algorithm consecutively draws a certain number of random traces that did not contain new information. Specifically, with δ as a measure that bounds the probability of a newly sampled trace to provide new information over L', at a significance level α, a minimum sample size N is computed. Based on the normal approximation to the binomial distribution, the latter is given as $N \geq z^2 * (1 - \delta)/\delta$, where z corresponds to the realisation of a standardised normal random variable for $1 - \alpha$ (one-sided hypothesis test). As such, N is calculated given values for δ and α for the desired levels of similarity and significance, respectively.

Consider $\alpha = 0.01$ and $\delta = 0.05$, so that $N \geq 128$. Hence, after observing 128 traces without new information, sampling can be stopped knowing with 0.99 confidence that the probability of finding new information in the remaining log is less than 0.05.

Using the above formulation, our framework for sample-based conformance checking is presented in Alg. 1. The algorithm takes as input an event log L, a process model M, the number of trials that need to fail N, a predicate γ to determine whether a trace provides new information, and a conformance function ψ. Going through L trace by trace (lines 4–14), the algorithm conducts a series of binomial experiments that check, if a new, randomly sampled, trace provides new information according to the predicate γ (line 8). Once N consecutive traces without new information have been selected, the procedure stops and the conformance result is derived based on the sampled log L'.

Result re-use. Note that the algorithm provides a conceptual view, in the sense that checking the new information predicate $\gamma(L', \xi, M)$ in line 8 according to Eq. 4, requires the computation of $\psi(L', M)$ and $\psi(\{\xi\} \cup L', M)$ in each iteration. A technical realisation of this algorithm, of course, shall exploit that most types of conformance results can be computed incrementally. For instance, considering the aforementioned conformance measures, an alignment is computed only once per *trace variant*, i.e., per unique sequence of activity executions, and reused in the iterations of the algorithm. Also, the value of $\psi(L', M)$ for $\gamma(L', \xi, M)$ in line 8 is always known from the previous iteration, while the conformance result in line 15 is not actually computed at this stage, as the respective result is known from the last evaluation of $\gamma(L', \xi, M)$.

In the next sections, we discuss how to define γ when the conformance function assesses the fitness of a log to a model, the observed deviation distribution, or contextual deviations related to resources.

Algorithm 1: Framework for Sample-Based Conformance Checking

```
input : $L$, an event log; $M$, a process model; $N$, a number
        of failed trials to observe; $\gamma$, a predicate that holds
        true, if a trace provides new information; $\psi$, a
        conformance function.
output: $\psi(L')$, the conformance results for sampled traces.
1  $L' \leftarrow \emptyset$;                 /* The sampled log */
2  $i \leftarrow 0$;   /* Iterations without new information */
3  $L'' \leftarrow \emptyset$;   /* Traces sampled since $i$ was reset */
4  repeat
5      $\xi \leftarrow sampleRandomTrace(L \setminus L')$;
6      $L'' \leftarrow L'' \cup \xi$;
7      if $\gamma(L', \xi, M)$ /* Check if new information   */
8       then
9          $L' \leftarrow L' \cup L''$;    /* add current traces */
10         $L'' \leftarrow \emptyset$;
11         $i \leftarrow 0$;             /* Reset the counter */
12     else
13         $i \leftarrow i + 1$;    /* Increment the counter */
14 until $i \geq N \vee (L' \cup L'') = L$;
   /* Repeats until $N$ traces without new
      information are seen consecutively         */
15 return $\psi(L')$ ;    /* Return conformance results */
```

3.2. Sample-Based Fitness

The overall conformance of a log to a model may be assessed by considering the log fitness (see Section 2) as a conformance function, $\psi_{fit}(L, M) = \text{fitness}(L, M)$. Then, determining whether a trace ξ provides new information over a log sample L' requires us to assess, if incorporating ξ leads to a difference in the overall fitness for the sampled log. Following Eq. 4, we capture this by computing the absolute difference between the fitness value for traces in the sample L' and the value of the sample plus the new trace:

$$d_{fit}(\psi_{fit}(L', M), \psi_{fit}(\{\xi\} \cup L', M)) = \left|\text{fitness}(L', M) - \text{fitness}(\{\xi\} \cup L', M)\right| \quad (5)$$

If this distance is smaller than the relaxation parameter ϵ, the change in the overall fitness value induced by trace ξ is considered to be negligible.

To illustrate this, consider a scenario with $\epsilon = 0.03$ and a sample consisting of the traces ξ_1 and ξ_3 of our running example (Fig. 1). Then, the log fitness for $\{\xi_1, \xi_3\}$ is 0.95. In this situation, if the next sampled trace is ξ_2, the distance function yields $|\text{fitness}(\{\xi_1, \xi_3\}, M) - \text{fitness}(\{\xi_1, \xi_2, \xi_3\}, M)| = 0.95 - 0.93 = 0.02$. In this case, since the distance is smaller than ϵ, we would conclude that the additional consideration of ξ_2 does not provide new information. By contrast, considering trace ξ_4 would yield a distance of $0.95 - 0.89 = 0.06$. This indicates that trace ξ_4 would imply a considerable change in the overall fitness value, i.e., it provides new information.

3.3. Sample-Based Deviation Distributions

Next, we instantiate the above framework for conformance checking based on the deviation distribution. As detailed in Section 2, this distribution captures the relative frequency with which activities are related to conformance issues.

To decide whether a trace ξ provides new information over a log sample L', we assess if the deviations obtained for ξ lead to a considerable difference in the overall deviation distribution. As such, the distance function for the predicate γ needs to quantify the difference between two discrete frequency distributions. This suggests to employ the *Euclidean distance* as a measure:

$$d_{dev}(\psi_{dev}(L', M), \psi_{dev}(\{\xi\} \cup L', M)) = \sqrt{\sum_{a \in \mathcal{A}} (f_{dev(L',M)}(a) - f_{dev(\{\xi\} \cup L', M)}(a))^2} \quad (6)$$

Taking up our example from Fig. 1, processing only the trace ξ_1, all deviations are related to the activity of fetching an earlier claim, i.e., $f_{dev(\{\xi_1\},M)}(F) = 1$. Notably, this does not change when incorporating traces ξ_2 and ξ_3, i.e., $f_{dev(\{\xi_1,\xi_2,\xi_3\},M)}(F) = 1$, as they do not provide new information in terms of the deviation distribution. If, after processing trace ξ_1, however, we sample ξ_4, we do observe such a difference: based on $f_{dev(\{\xi_1,\xi_4\},M)}(F) = 2/3$ and $f_{dev(\{\xi_1,\xi_4\},M)}(U) = 1/3$, we compute a distance of 0.47. For a relaxation parameter ϵ that is smaller than this value, we would thus conclude that ξ_4 provides novel information.

Given the distance functions based on trace fitness and deviation distribution, it is interesting to note that these behave differently, as illustrated in our example: If the log is $\{\xi_1, \xi_2\}$ and trace ξ_3 is sampled next, the overall fitness changes. Yet, since ξ_3 is a conforming trace, it does not provide new information on the distribution of deviations.

3.4. Sample-based Resource Deviations

Finally, we consider conformance issues related to the context of process execution. As detailed in Section 2, we illustrate the handling of contextual factors for the example of resource assignments. Specifically, a set of resource deviations includes the resources that, according to the event log, executed an activity even though this execution should not have happened in the respective state, so that the event is part of a skip step in an optimal alignment of the trace. Moreover, the set of resource deviations contains resources that executed activities, despite not being in charge of the activities according to the process model.

From the above it follows that a trace ξ provides new information over a log sample L', if the set of resource deviations changes considerably. The latter is determined as a relative change in the size of the set, which leads to the following formulation of the distance function as part of the definition of the predicate γ:

$$d_{res}(\psi_{res}(L', M), \psi_{res}(\{\xi\} \cup L', M)) = \frac{1}{|A|} \sum_{\substack{a \in \mathcal{A} \\ s_{res(\{\xi\} \cup L', M_{res})}(a) \neq \emptyset}} \frac{\left|s_{res(\{\xi\} \cup L', M_{res})}(a) \setminus s_{res(L', M_{res})}(a)\right|}{\left|s_{res(\{\xi\} \cup L', M_{res})}(a)\right|} \quad (7)$$

Returning to our running example, see Fig. 1, we observe the following. Processing only trace ξ_1, the set of resource deviations for the activity of fetching previous claims (F) includes a single employee, i.e., $s_{res(\{\xi_1\}, M_{res})}(F) = \{Pete\}$. Upon processing trace ξ_2, this set changes and now also includes a second employee, i.e., $s_{res(\{\xi_1,\xi_2\}, M_{res})}(F) = \{Pete, Ine\}$. The respective distance would be computed as $1/5 \cdot 1/2 = 0.1$.

Traces ξ_3 and ξ_4, in turn, do not provide further information on resource deviations. While trace ξ_4 also includes a deviation linked to the employee $Pete$, this does not constitute new information in terms of the *set* of resourced involved in non-conforming activities.

4. Approximation-Based Conformance Checking

In this section we propose methods that approximate the impact of a sampled trace on the overall conformance result, which can be employed to further reduce the number of computationally expensive alignments that need to be established. As such, the approximation methods complement the sampling methods introduced in the previous section: Even when a trace of a yet unseen variant is sampled, we decide whether to compute an actual conformance result or whether to approximate it. This decision is based on the *worst-case* impact that a conformance result can have according to the approximated result, which is obtained by comparing the trace at hand to similar variants for which conformance results have previously been derived.

In Section 4.1, we describe how result approximation can be integrated into our sample-based conformance checking approach. Section 4.2 discusses how similarity between trace variants can be assessed, while Section 4.3 presents instantiations of the approximation method for conformance results based on log fitness, deviation distribution, and contextual deviations related to resources.

4.1. Conformance Approximation Framework

Against this background, our technique for approximation-based conformance checking, formalised in Alg. 2, extends our procedure given in Alg. 1. In fact, it primarily provides a realisation of checking the new information predicate $\gamma(L', \xi, M)$, as done in line 8 of Alg. 1. That is, whether the sampled trace ξ, which is of an unseen variant, provides new information may be decided based on its approximated, rather than its computed impact on the overall conformance result. At the same time, however, the algorithm also needs to keep track of all sampled traces $L'_a \subseteq L'$, for which the approximated results shall be used whenever a conformance result is computed. This leads to an adaptation of the conformance function ψ, i.e., we consider a *partially approximating conformance* function $\hat{\psi} : 2^{\mathcal{E}^*} \times 2^{\mathcal{E}^*} \times \mathcal{M} \rightarrow \mathcal{X}$. Given a log L' and a subset $L'_a \subseteq L'$, this function approximates the conformance result $\psi(L', M)$ by computing solely $\psi(L' \setminus L'_a, M)$, i.e., the impact of traces L'_a is not precisely computed. In the same way, to use the approximation technique as part of sample-based conformance checking, the use of the conformance function ψ in Alg. 1 also has to be adapted accordingly.

Algorithm 2: Framework for Approximation-Based Conformance Checking

```
input  : $L'$, a log sample; $M$, a process model; $\xi$, a trace
         sampled of a yet unseen variant with $\xi \notin L'$; $d$, a
         distance function; $\epsilon$, a relaxation parameter; $k$, a
         similarity threshold; $\hat{\psi}$, a partially approximating
         conformance function; $L'_a \subseteq L'$, traces for which
         approximated results are used.
output : $v \in \{\texttt{true}, \texttt{false}\}$, indicates whether $\xi$ provides
         new information.
1  $\xi_r \leftarrow \text{argmin}_{\xi' \in L' \setminus L'_a}\ sim(\xi, \xi')$;    /* Select most
     similar trace */
2  if $sim(\xi, \xi_r) \leq k$ ;          /* Check if $k$-similar */
3  then
4      $\Phi \leftarrow approx(\xi, \xi_r, L', M)$ ;          /* Derive all
         possible approximations */
5      if $\exists\, \phi \in \Phi : d(\hat{\psi}(L', L'_a, M), \phi) > \epsilon$ ; /* Check if
         approximation indicates new information */
6      then
7          return true ;     /* Approach will compute
             actual result */
8      else  /* Approach uses approximated result */
9          $L'_a \leftarrow L'_a \cup \{\xi\}$ ;   /* Add $\xi$ to traces with
             approximated results */
10         return false;
11 else    /* No $k$-similar trace available, compute
     real result to check for new information */
12     $v \leftarrow d(\hat{\psi}(L', L'_a, M), \hat{\psi}(\{\xi\} \cup L', L'_a, M)) > \epsilon$;
13     return $v$;
```

Turning to the details of Alg. 2, its input includes a log sample L', a sampled trace $\xi \notin L'$, and a process model M, i.e., the arguments of γ in line 8 of Alg. 1, as well as a distance function d and relaxation parameter ϵ from the definition of γ (Eq. 4). Moreover, there is a similarity threshold k to determine which traces may be used for approximation. Finally, the aforementioned set of traces L'_a for which results shall be approximated and the respective adapted conformance function $\hat{\psi}$ are given as input.

From the sampled traces for which approximation is not applied (i.e., $L' \setminus L'_a$), the algorithm first selects the trace that is most similar to ξ, referred to as the reference trace ξ_r (line 1). Then, to reduce the introduced error of the approximation, we assess whether the similarity is above the similarity threshold k (line 2). If not, we check the trace for new information as done before, just using the adapted conformance function (lines 12–13). If ξ_r is sufficiently similar, however, we perform a worst-case approximation of the impact of ξ on the overall conformance result based on ξ_r (line 4). As part of that, we may obtain several different approximations Φ, each of which is checked whether it indicates new information over the current sample L' (line 6). Only if this is not the case, we conclude that ξ indeed does not provide new information and under ϵ. By adding it to L'_a we make sure that its impact on the overall conformance will always only be approximated, but never precisely computed, and that the conformance approximation of new traces will never be based on ξ (line 9).

Next, we give details on the assessment of trace similarity (function sim, Section 4.2) and the conformance result approximation (function $approx$, Section 4.3).

4.2. Trace Similarity

Given a trace ξ and the part of the sample log for which approximation did not apply ($L' \setminus L'_a$), Alg. 2 requires us to identify a reference trace ξ_r that is most similar according to some function $sim : \mathcal{E}^* \times \mathcal{E}^* \rightarrow [0, 1]$. As we consider conformance results that are based on alignments, we define this similarity function based on the alignment cost of two traces. To this end, we consider a function c_t, which, in the spirit of function c discussed in Section 2, is the sum of the costs assigned to skip steps in an optimal alignment of two traces. It is important to note here that the complexity of trace-to-trace alignment using the algorithm by Needleman and Wunsch [22], as suggested by Bose et al. [23], has a considerably smaller computational complexity ($|\xi| \times |\xi'|$) than the exponential complexity of the trace-to-model alignment that our approximation method aims to avoid.

To obtain a similarity measure, we normalise this aggregated cost by a maximal cost, which is obtained by aligning each trace with an empty trace. This normalisation resembles the one discussed for the fitness measure in Section 2. We define the similarity function for traces as $sim(\xi, \xi') = 1 - c_t(\xi, \xi')/(c_t(\xi, \langle\rangle) + c_t(\xi', \langle\rangle))$.

Considering trace $\xi_4 = \langle r, p, f, f, s\rangle$ of our running example, the most similar trace (assuming equal costs for all skip steps) is $\xi_1 = \langle r, p, f, f, u, s\rangle$, with $c_t(\xi_1, \xi_4) = 1$ and, thus, $sim(\xi_1, \xi_4) = {}^{10}/_{11}$.

4.3. Conformance Result Approximation

In the approximation step of Alg. 2, we derive a set of worst-case approximations of the impact of the trace ξ on the overall conformance result, using the reference trace ξ_r (which is at least k-similar). Based thereon, it is decided whether ξ provides new information. The approximation, however, depends on the type of conformance result.

4.3.1. Fitness Approximation

To approximate the impact of trace ξ on the overall fitness, we compute a single value, i.e., $approx(\xi, \xi_r, L', M)$ in line 4 of Alg. 2 yields a singleton set. This value is derived by reformulating Eq. 5, which captures the change in fitness induced by a sample trace. That is, we assess the difference between the current fitness, fitness(L', M), and an approximation of the fitness when incorporating ξ, i.e., fitness$(\{\xi\} \cup L', M)$. This approximation, denoted by $\widehat{fit}(\xi, \xi_r, L', M)$, is derived from (i) the change in fitness induced by the reference trace ξ_r, and (ii) the differences between ξ and ξ_r. The former is assessed using the aggregated alignment cost $c(\xi_r, M)$, whereas the latter leverages the aggregated cost of aligning the traces, $c_t(\xi, \xi_r)$. Normalising these costs, function $approx_{fit}(\xi, \xi_r, L', M)$ yields a worst-case approximation for the change in overall fitness imposed by ξ, as follows:

$$approx_{fit}(\xi, \xi_r, L', M) = \left\{ \left| \text{fitness}(L', M) - \widehat{fit}(\xi, \xi_r, L', M) \right| \right\} \quad (8)$$

Where $\widehat{fit}(\xi, \xi_r, L', M)$ is defined as:

$$\widehat{fit}(\xi, \xi_r, L', M) = 1 - \frac{\sum_{\xi' \in L'} c(\xi', M) + c(\xi_r, M) + c_t(\xi, \xi_r)}{\sum_{\xi' \in L'} c(\xi', \emptyset) + \max_{\substack{\xi' \in \\ \{\xi, \xi_r\}}} c(\xi', \emptyset) + (|L'| + 1) \min_{x \in M} \cdot c(\langle\rangle, \{x\})} \quad (9)$$

Turning to our running example, assume that we have sampled $\{\xi_1, \xi_2\}$ and computed the precise fitness value based on both traces, which is $1 - 2/(12 + 10) \approx 0.909$ using a standard cost function. If trace ξ_4 is sampled next, we approximate its impact using the most similar trace ξ_1. To this end, we consider $c(\xi_1, M) = 1$ and $c_t(\xi_1, \xi_4) = 1$, which yields an approximated fitness value of $1 - (2+1+1)/(12+6+15) = 1 - 4/33 \approx 0.879$. This is close to the actual fitness value for $\{\xi_1, \xi_2, \xi_4\}$, which is $1 - 4/(17+15) \approx 0.875$. The minor difference stems from ξ_1 being slightly longer than ξ_4.

4.3.2. Deviation Distribution Approximation

To approximate the impact of trace ξ on the deviation distribution, we follow a similar approach as for fitness approximation. However, we note that the approximation function here yields a set of possible values, as there are multiple different distributions to be considered when measuring the distance to the current distribution. The reason being that the difference between ξ and the reference trace ξ_r induces a set of possible changes of the distribution.

Specifically, we denote by $ed(\xi, \xi_r)$ the edit distance of the two traces, i.e., the pure number of skip steps in their alignment. This number gives an upper bound for the number of conformance issues that need to be incorporated in addition to those stemming from the alignment of the reference trace and the model, i.e., $dev(\{\xi_r\}, M)$. Yet, the exact activities associated to these skips steps are not known. Therefore, we need to consider all bags of activities of size $ed(\xi, \xi_r)$, the set of which is denoted by $[\mathcal{A}]^{ed(\xi,\xi_r)}$. Each bag in $[\mathcal{A}]^{ed(\xi,\xi_r)}$, representing a specific combination of activities from $\mathcal{A}$ of size $ed(\xi, \xi_r)$, reflects a potential combination of deviations. As such, each of these bags can have a different impact on the approximated deviation distribution, i.e., leading to a different approximation $\widehat{f_{dev}}(\beta, \xi_r, L', M)$ of the distribution $f_{dev(\{\xi\}\cup L',M)}$. We compute those approximated impacts as follows:

$$approx_{dev}(\xi, \xi_r, L', M) = \bigcup_{\substack{k=ed(\xi,\xi_r)\\ \beta\in[\mathcal{A}]^k}} \left\{ \sum_{a\in\mathcal{A}} \left| f_{dev(L',M)}(a) - \widehat{f_{dev}}(\beta, \xi_r, L', M)(a) \right| \right\} \tag{10}$$

Here, $approx_{dev}(\xi, \xi_r, L', M)$ contains the set of potential distances between the observed deviation distribution $f_{dev(L',M)}(a)$ and the deviation distributions obtained after adding each of the possible combinations $\beta \in [\mathcal{A}]^{ed(\xi,\xi_r)}$, computed using the following distance function:

$$\widehat{f_{dev}}(\beta,\xi_r, L', M)(a) = \frac{dev(L', M)(a) + dev(\{\xi_r\}, M)(a) + \beta(a)}{|dev(L', M)| + |dev(\{\xi_r\}, M)| + |\beta|} \tag{11}$$

Reconsider our example: Based on a sample consisting $\{\xi_1, \xi_2\}$, we determine that $dev(\{\xi_1, \xi_2\}, M) = [F^2]$ and that $f_{dev(\{\xi_1,\xi_2\},M)}(F) = 1$. If ξ_4 is then sampled, we obtain an approximation based on $dev(\{\xi_1\}, M) = [F]$ and $ed(\xi_4, \xi_1) = 1$. We therefore consider the change in the distribution incurred by approximating the deviations of ξ_4 as $[F] \uplus \beta$ with $\beta \in \{[R], [P], [F], [U], [S]\}$, i.e., the approximated deviation is any combination between $[F]$ and another activity. For instance, $\widehat{f_{dev}}([R], \xi_4, \{\xi_1, \xi_2\}, M)$ yields a distribution assigning relative frequencies of $3/4$ and $1/4$ to activities F and R, respectively.

The above approximation method may be tuned heuristically by narrowing the set of activities that are considered for β, i.e., the possible deviations incurred by the difference between ξ and ξ_r. While this means that $\widehat{f_{dev}}$ is no longer a worst-case approximation, it may steer the approximation in practice, hinting at which activities shall be considered for possible deviations. Such an approach is also beneficial for performance reasons: Since β may be any bag built of the activities in $\mathcal{A}$, the number of potential combinations increases exponentially. This blow-up limits the applicability of the approximation only to traces that are rather similar, i.e., for which $ed(\xi, \xi_r)$ is small. Here, we describe three specific heuristics to prune the number of possible combinations:

1. A first heuristic, already described in [19], determines the overlap between ξ and ξ_r in terms of their maximal shared prefix and suffix of activities, for which the execution is signalled by their events. The heuristic then identifies those events that are *not* part of the shared prefix and suffix, so that we only consider the activities referenced by the non-shared events for the construction of β.
2. Second, we consider only those activities of ξ and ξ_r that correspond to skip steps while aligning ξ to ξ_r. This heuristic is more precise than the first one, given that it also recognises overlaps that occur outside of a shared prefix and suffix. Furthermore, it has the added benefit that this heuristic yields a single, uniquely defined set of activities β of size $ed(\xi, \xi_r)$, rather than a number of possible combinations.
3. Third, we refine the second heuristic by aiming to further reduce approximation error. We achieve this by only considering activities for β that have already been observed to be deviating, i.e., for which an alignment between trace and model indicated a skip step. This avoids the case where approximation identifies potential deviations for activities that actually never deviate. This, furthermore, reduces a bias of the approximation method against relatively uncommon activities, as they are less likely to occur in a reference trace ξ_r and, therefore, are more likely to be related to a skip step during the alignment of ξ to ξ_r.

In our example, traces ξ_1 and ξ_4 share the prefix $\langle r, p, f, f\rangle$ and suffix $\langle s\rangle$. Thus, ξ_1 contains one event between the shared prefix and suffix, u, while there is none for ξ_4. Hence, if the first heuristic is applied, we consider a single bag of deviations, $\beta = [U]$, and $\widehat{f_{dev}}([U], \xi_4, \{\xi_1, \xi_2\}, M)$ is the only distribution considered in the approximation. The alignment-based second heuristic would identify the same, unique potential deviation set.

To illustrate the difference between the third heuristic and the others, we consider traces ξ_1 and ξ_3. The alignment of these traces results in three skip steps, related to the executions of activities F and U (two times). The application of the first and second heuristics would consider the execution of activity U as a potential deviating activity, given that it differs between ξ_3 and the reference trace ξ_1. However, in the traces analysed so far, we have never observed a deviation involving activity U. Therefore, the third heuristic considers only activity F as potentially deviating.

4.3.3. Resource Deviation Approximation

When approximating the impact of a trace ξ on the resource deviations, we first assess the worst-case scenario: Given the resource violations already observed, we check if there would be new information (according to the predicate γ), if all resources involved in ξ would lead to resource violations. Put differently, we exploit that the number of resources referenced in the events of a trace provides an upper bound for the difference between the sizes of the sets of resource deviations before and after incorporating the trace. While this is a rather coarse-grained approximation, it can be expected to be beneficial in various application scenarios. If a trace relates to only a few resources (or none at all) or if virtually all the referenced resources have already been observed as violating, even such a worst-case approximation is likely to yield a negative γ, in which case we know for sure that no new information can be observed for ξ.

If the above test yields a positive γ for a trace ξ, we approximate its impact in a more fine-granular manner. In general, approximation should consider the two potential sources of resource deviations, i.e., (1) the resources referenced in events that are part of skip steps in an optimal alignment and (2) resources that executed activities without being assigned the respective responsibility in the process model, see also Eq. 3. The latter type of deviations does not impose any computational challenge. It can be determined precisely in linear time in the length of the trace by verifying the containment of the resources referenced by the events in the set of resources assigned to the respective activities in the process model. As such, we do not approximate this type of resource deviation. We denote the respective set of deviations for trace ξ that originate from the execution of an activity a by resources that are not responsible according to model M_{res} as $dev_{res}(\xi, M_{res})(a)$.

To approximate the resource deviations stemming from skip steps, we first identify a reference trace ξ_r for ξ based on the edit distance of the traces, $ed(\xi, \xi_r)$, as described above for the approximation of the deviation distribution. From the optimal alignment $\sigma(\xi_r, M) = \langle (e'_1, t_1), \ldots, (e'_m, t_m) \rangle$ of the reference trace, we then derive the set of activities of events that are part of skip steps in the alignment, i.e., $dev_{act}(\xi_r) = \bigcup_{1 \leq i \leq m, t_i = \perp} \{e'_i.\texttt{activity}\}$. Based on these activities, we derive all resources that are referenced in trace $\xi = \langle e_1, \ldots, e_n \rangle$ for executions of these activities, $map_{res}(\xi, \xi_r) = \bigcup_{1 \leq i \leq n, e_i.\texttt{activity} \in dev_{act}(\xi_r)} \{e_i.\texttt{resource}\}$. In addition, we need to take into account that the difference between trace ξ and the reference trace ξ_r may also lead to additional resource deviations. Here, the computation of the edit distance $ed(\xi, \xi_r)$ identifies a set of events $ev(\xi, \xi_r)$ of ξ that differ from ξ_r, which, in turn, lead to another set of resources, defined as $\mathit{diff}_{res}(\xi, \xi_r) = \bigcup_{1 \leq i \leq n, e_i \in ev(\xi, \xi_r)} \{e_i.\texttt{resource}\}$.

Summarizing the above arguments, we approximate the impact of resource deviations by defining the set of resources potentially involved in deviations of an activity a. This set encompasses the known deviations of ξ, i.e., $dev_{res}(\xi, M_{res})(a)$, the known deviations of ξ_r, i.e., $map_{res}(\xi, \xi_r)(a)$, and the resources not yet observed for violations of a, i.e., $\mathit{diff}_{res}(\xi, \xi_r)(a)) \setminus s_{res(L', M_{res})}(a)$. Formally given as:

$$\begin{aligned} &\widehat{s_{res}}(\xi, \xi_r, L', M_{res})(a) = \\ &\quad (dev_{res}(\xi, M_{res})(a) \cup map_{res}(\xi, \xi_r)(a) \cup \\ &\quad \mathit{diff}_{res}(\xi, \xi_r)(a)) \setminus s_{res(L', M_{res})}(a) \end{aligned} \tag{12}$$

Then, we approximate the change in resource deviations for trace ξ as the number of potential new resource deviations per activity:

$$\begin{aligned} &approx_{res}(\xi, \xi_r, L', M_{res}) = \\ &\qquad\qquad \left\{ \sum_{a \in \mathcal{A}} \left| \widehat{s_{res}}(\xi, \xi_r, L', M_{res})(a) \right| \right\} \end{aligned} \tag{13}$$

5. Sample Quality Assurance

The approaches proposed in our work estimate conformance results based on a sample of traces taken at random from a log, yielding considerable gains in terms of efficiency, while still providing statistical guarantees on the results quality. Despite these clear benefits, the random sampling that is necessary for our approach leaves the possibility that the obtained conformance results stem from a biased sample and, therefore, may be inaccurate. In particular, if a series of sampled traces is more homogeneous than the log, our approach may return conformance results based on a sample that does not represent the variety observed in the full event log. Directly assessing whether the conformance results for a sample L' are representative is infeasible, given that it would require what our approach aims to avoid: computing conformance results for an entire log. To still be able to assess sample quality, this section proposes alternative techniques that analyse samples of traces in terms of their behavioural and data attribute coverage. In particular, we introduce methods for:

- *Internal sample quality assessment*: We consider how the quality of a sample can be assessed by looking at the control-flow and data-attribute characteristics of the traces within the sample itself. This is achieved by extending the *new information predicate* of the sampling approach.
- *External sample quality assessment*: We consider how the quality of a sample can be assessed by comparing it against traces outside the sample, i.e., by selecting an set of traces and comparing their control-flow and data-attribute characteristics.

The two methods provide a trade-off between the quality assurance they provide and the additional runtime they require, where the external quality checks can provide stronger guarantees, but introduces more computational overhead. The internal and external checks can be employed individually, as well as together.

In the remainder, Section 5.1 first discusses various control-flow and data-attribute characteristics of a trace sample that may indicate its quality. Afterwards, Sections 5.2 and 5.3 respectively describe our proposed methods for internal and external sample quality assessment.

5.1. Sample Quality Characteristics

We propose to assess the quality of log samples by considering their *behavioural representativeness* and *data attribute coverage*.

Behavioural representativeness. When determining process conformance based on a sample taken from an event log, it is important that the process behaviour contained in the sample reflects the behaviour of the traces in the full log. To achieve this, we follow existing approaches that assess the completeness of (a sample of) an event log, see [24, 25, 26], and consider quality by analysing the behavioural relations contained in a log sample. In the remainder we shall consider the *directly follows* relation for this, as suggested by the aforementioned approaches. Nevertheless, we note that this relation can be substituted by other behavioural relations, such as (causal) behavioural profile relations [27] or the relations of the 4C spectrum [28].

When quantifying the behavioural representativeness of a sample, we consider the included behavioural relations, as well as their frequencies. The latter is done to ensure that the distribution of the relations in a log sample reflects the distribution of a full event log [24]. Therefore, given some set of traces L, (which can be a full log or a sample), we represent the behaviour of L as a function that captures the number of times that two activities directly follow each other in the traces of L, i.e., $df(L) : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{N}_0$. The function $df(L)$ can be used to compare the behaviour of two sets of traces, e.g., of two samples or of a sample to an event log.

Data attribute coverage. Aside from representativeness in terms of behaviour, it can be important to ensure that the data attributes associated with sampled traces (and their events) provide a proper reflection of the data perspective for the full event log. Such coverage of data attributes is important, given that there may be a clear correlation between data attribute values and process behaviour (and thus also conformance issues). For instance, if an event log contains traces corresponding to 80% `regular` and 20% `premium` customers, using a sample that consists for 95% of traces for `regular` customers likely introduces a certain bias into the obtained conformance results.

To incorporate data attribute coverage in our sampling framework, we capture the distribution of data attribute values. For a nominal attribute $D \in \mathcal{D}$, with a domain $dom(D)$, we capture the value distribution for a set of traces L as: $val(D, L) : dom(D) \rightarrow \mathbb{N}_0$. For instance, given a log sample L with 100 traces, we would observe $val(\texttt{customer\_type}, L) = [\texttt{regular}^{80}, \texttt{premimum}^{20}]$ for a representative sample, whereas a non-representative one would have a considerably different value distribution. For numeric attributes, we simply capture the values for attribute D per trace, i.e., $val(D, L) : L \rightarrow \mathbb{R}$.

Note that domain knowledge can be particularly helpful when considering data attribute coverage. Users may select a subset of data attribute values $\mathcal{D}' \subseteq \mathcal{D} \cup \{df\}$ that are known to be of particular relevance to observed process behaviour. For instance, in a loan handling process, attributes such as `customer_type` and `loan_amount` may be known to be important, whereas attributes such as `zip_code` may be omitted from consideration. We consider the directly follows relation as a special attribute that can be included in a subset $\mathcal{D}'$, since it shall be handled in the same way as other qualitative attributes in the procedures defined next.

5.2. *Internal Sample Quality Analysis*

The first way in which we propose to consider sample quality analysis based on behavioural representativeness and data attribute coverage is to directly incorporate such characteristics into the incremental conformance checking framework introduced in Section 3. The goal of this *internal* sample quality analysis is to check whether the characteristics of a sample are stable or are still fluctuating when new traces are added. We achieve this by extending the predicate γ from Eq. 4, which is used to determine if a trace ξ provides new information w.r.t. a log sample L'.

In particular, given a set $\mathcal{D}' \subseteq \mathcal{D} \cup \{df\}$ of attributes to incorporate, we now use the predicate γ to check if the addition of an additionally sampled trace ξ leads to a significant difference for either the conformance result, captured by function ψ or in any of the quality assurance attributes in $\mathcal{D}'$, i.e., we check if any distance function for ψ or an element in $\mathcal{D}'$ yields a result greater than ϵ. The extended predicate, γ_q is formalized as follows:

$$\begin{aligned}&\gamma_q(L', \xi, M) \Leftrightarrow \\ &\quad \big(d(\psi(L', M), \psi(\{\xi\} \cup L', M)) > \epsilon\big) \vee \\ &\quad \big(\max_{D \in \mathcal{D}'} (d_D(L', \xi)) > \epsilon\big).\end{aligned} \tag{14}$$

To determine the distance d_D for an attribute $D \in \mathcal{D}'$, we differentiate between nominal and numerical attributes. *Nominal attributes* (including the directly follows relation df) are represented as the number of occurrences for each of the values in a domain (i.e., $dom(D) \rightarrow \mathbb{N}_0$). Therefore, to determine if new information regarding such a data attribute is observed when considering a new trace, we adapt the distance predicate used when considering the distribution of deviations (see Eq. 6), as follows:

$$d_D^{nom}(L', \xi) = \sqrt{\sum_{d \in dom(D)} (f_D(L', d) - f_D(\xi \cup L', d))^2} \tag{15}$$

In Eq. 15, we use $f_D(L', d)$ to denote the relative frequency of attribute D being associated with value d for the traces in L'. For instance, if a sample L' so far consists of 4 traces, with 1 `premium` and 3 `regular` customers and $\xi_D =$ `regular`, the distance $d_D = \sqrt{(3/4 - 4/5)^2 + (1/4 - 1/5)^2} = 0.07$. To quantify the change in a *df* relation, there are alternatives to the relative frequency f_D depicted in Eq. 15. For instance, one can employ the *dependency measure*, part of the Heuristics miner [17, p.165], as an alternative that considers relations in a bi-directional manner, i.e., it considers the number of times an activity a is directly followed by b, as well as the number of times b is directly followed by a.

For *numerical attributes*, we adapt the distance metric used in the context of fitness-based sampling, as shown in Eq. 5, which defines a distance metric based on the average value of an attribute D.

$$d_D^{num}(L', \xi) = \left| \underset{\substack{e \in \xi' \\ \xi' \in L'}}{\text{avg}} (e'.D) - \underset{\substack{e \in \xi' \\ \xi' \in L' \cup \{\xi\}}}{\text{avg}} (e'.D) \right| \tag{16}$$

By extending the new information predicate γ with such additional distance functions, our approach shall only return conformance results when it has been determined that both the conformance results and the attribute values over sample L' are statistically likely to be representative for the log L. Whereas this method incorporates additional assurances through internal sample analysis, we next consider how sample representativeness can also be achieved by looking at traces outside a sample.

5.3. *External Sample Quality Analysis*

To assess the quality of a sample externally, we compare the chosen data attributes in a post-hoc test against additionally sampled data. In particular, given a sample $L' \subset L$, we compare the value distributions for the characteristics in $\mathcal{D}'$ to those of a set of traces outside the sample, i.e., a validation set L''. Formally, we are testing whether the distributions

Given a log sample L' and a validation set L'', our goal is to ensure that L' is representative along all attributes in $\mathcal{D}'$. To achieve this, we check if there is a statistically significant difference (given a desired significance level α) for any $D \in \mathcal{D}'$. To this end, we perform tests that determine if the value distributions for an attribute D statistically differ between L' and L'', i.e., we test the null hypothesis that $val(D, L')$ and $val(D, L'')$ are statistically likely to stem from the same distribution. For this, we employ the well-known *chi-square two sample test* [29] for nominal attributes and the Kolmogorov-Smirnov test [30] for numerical attributes.

Conducting the statistical tests is incorporated as an additional check into the conformance checking framework described in Alg. 1. In particular, when sufficient consecutive traces without new information have been observed, i.e., when $i = N$, the algorithm draws an additional random sample of traces from $L \setminus L'$, referred to as L''. Then, for each attribute $D \in \mathcal{D}'$, a statistical check is performed between the value distributions of D for the two samples, i.e., comparing $val(D, L')$ to $val(D, L'')$. As soon as any attribute yields a significant difference, i.e., larger than a desired α, we conclude that the sample L' is not sufficiently unbiased. Therefore, we then reset parameter i to 0, indicating that a new run of N consecutive traces without new information is required.

Note that selecting the size of the validation sample L'' represents an interesting parameter here, set to the size of the current sample L' by default. A larger size increases the confidence that can be placed into the performed quality check, although it also requires additional computation. Nevertheless, it is important to recognise that sampling additional traces for quality analysis is considerably more efficient than increasing the sample size of L' itself given that we avoid the need to compute additional alignments in this case.

6. Evaluation

This section reports on an experimental evaluation of the proposed techniques for sample-based and approximation-based conformance checking. Section 6.1 describes the three real-world and seven synthetic event logs used in the experimental setup, described in Section 6.2. We present an in-depth analysis evaluation of various aspects of our approach in Section 6.3. Overall, these demonstrate that our techniques achieves considerable efficiency gains, while still providing highly accurate conformance results.

6.1. Datasets

We conducted our experiments based on three real-world and seven synthetic event logs, which are all publicly available.

Real-world data. The three real-world event logs differ considerably in terms of the number of unique traces they contain, as well as their average trace lengths, which represent key characteristics for our approach:

- *BPI-12* [31] is a log of a process for loan or overdraft applications at a Dutch financial institute that was part of the Business Process Intelligence (BPI) Challenge. The log contains 13,087 traces (4,366 variants), with 20.0 events per case (avg.).
- *BPI-14* [32] is the log of an ICT incident management process used in the BPI Challenge. For the experiments, we employed the event log of incidence activities, containing 46,616 traces (31,725 variants), with 7.3 events per case (avg.).
- *Road Traffic Fines* [33] is a log of an information system managing road traffic fines (RTF). The log contains 150,370 traces (231 variants), with 3.7 events per case (avg.).

We obtained accompanying process models for these logs by applying the inductive miner infrequent [34] with its default parameter settings (i.e., 20% noise filtering) on the full log. Furthermore, for the Traffic Fines log, we also conducted experiments with a manually established process model from [35]. Note that we shall refer to the results obtained with this model as RTFr, whereas the results from the discovered model are referred to as RTF.

Table 1: Characteristics of the synthetic data collection

Characteristic	PrA	PrB	PrC	PrD	PrE	PrF	PrG
Activities	363	317	317	429	275	299	335
Traces	1,200	1,200	500	1,200	1,200	1,200	1,200
Variants	1,049	1,126	500	1,200	1,200	1,200	1,200
Events per trace	31.6	41.5	42.9	248.6	98.8	240.8	143.1

Synthetic data. To analyse the scalability of our techniques, we employed a synthetic dataset specifically designed to stress-test conformance checking techniques [36]. It consists of seven process models and accompanying event logs. The models are considerably large and complex, as characterised in Table 1, which impacts the computation of alignments. Also, the included event logs consist of a high number of variants compared to the number of traces (i.e., almost no repeated variants), which may affect the effectiveness of sampling.

6.2. Experimental Setup

We employed the following measures and experimental setup to conduct the evaluation.

Measures. We measure the *efficiency* of our techniques by the fraction of traces from a log required to obtain our conformance results. This fraction indicates for how many traces the conformance computation was not needed due to the trace not being sampled. Simultaneously, we consider the fraction of the total trace variants for which our techniques actually had to establish alignments. As these fractions provide us with analytical measures of efficiency, we also assess the *runtime* of our techniques, based on a prototypical implementation. Again, this is compared to the runtime of the conformance checking over the complete log. Finally, we assess the impact of sampling and approximation on the *accuracy* of conformance results. We determine the accuracy by comparing the results, e.g., the fitness value or the deviation distribution, obtained using sampling and approximation, to the results for the total log.

All presented results are determined based on 10 experimental runs (i.e., replications) of which we report on the mean value, along with the 10th and 90th percentiles.

To ensure that the results of the various experiments can be fairly compared, we fixed the seeds for the replications throughout the evaluation.

Environment. Our approach has been implemented as a publicly available[1] plugin in ProM [37]. For the computation of alignments, we rely on the ProM implementation of the search-based technique proposed in [7].

All runtime measurements have been obtained on a NUMA server featuring 120 CPUs (Xeon E7-4880, 2.50GHz) and 1TB RAM, running Oracle Java 1.8. We used 10 parallel threads to obtain the results. Note that this high degree of parallelization means that it is more difficult to improve over baseline techniques that consider the complete log. Therefore, the reported improvements obtained using our techniques would be stronger in sequential settings. Our experimental setup, therefore, corresponds to an enterprise-level infrastructure that would be used in cloud-based commercial tooling.

6.3. Evaluation Results

This section first considers the overall efficiency and accuracy of our approach on the real-world event logs (Section 6.3.1), before conducting a parameter sensitivity analysis (Section 6.3.2), and demonstrating the scalability of our approach by showing that its performance also applies to complex, synthetic datasets (Section 6.3.3). Afterwards, we evaluate the performance of the novel contributions w.r.t. the conference paper [19], i.e., we evaluate the new approximation heuristics (Section 6.3.4), the techniques developed for the detection of contextual deviations related to resources (Section 6.3.5), and the introduced quality assessment methods (Section 6.3.6). Finally, Section 6.3.7 provides a recap of the main configurations of our approach and their impact on conformance checking accuracy and efficiency.

For brevity and clarity, we often highlight results obtained for a subset of the employed data collection in the remainder of this section. However, we note that the overall trends were consistent across all event logs and, furthermore, provide the raw result data and detailed figures obtained for the entire collection on the repository linked in Section 6.2.

6.3.1. Overall Results

We first assess the overall efficiency and accuracy obtained with our approach using default parameter values, i.e., for $\delta = 0.01$, $\alpha = 0.99$, $\epsilon = 0.01$, $k = 0.2$ and the third heuristic for deviation approximation introduced in Section 4.3.2.

Efficiency. We explore the efficiency in terms of sample size and runtime for four configurations: conformance in terms of fitness, without (*f*) and with approximation (*fa*), as well as for the deviation distribution without (*d*) and with approximation (*da*).

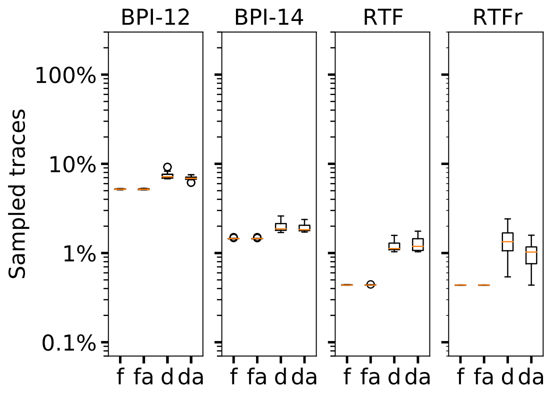


Figure 2: Percentage of total traces used to obtain results relative to the baseline

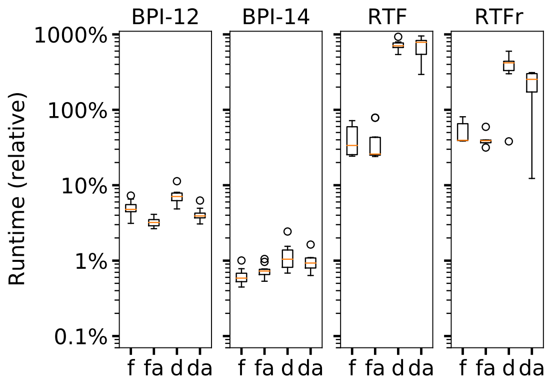


Figure 3: Runtime of our approach relative to the baseline

Fig. 2 reveals that all configurations only used a fraction of the complete log in order to produce conformance results. For instance, for BPI-14, the sample-based fitness computation (*f*) requires only 681 traces (on average) out of the total of 46,616 traces (i.e., 1.4% of the log). This sample included traces from 327.1 variants on average, which means that the approach established just 327.1 alignments as opposed to the total number of 31,725 variants in the log (roughly 1%). The smallest sample sizes are required for the Traffic Fines dataset, where the computation of fitness requires only about 0.5% of the event log. This result is due to the relatively

[1] https://github.com/Martin-Bauer/Conformance_Sketching

low number of variants (231 against 150,370 traces) and short traces (3.7 events per case on average). Both factors positively impact the number of traces necessary to obtain accurate conformance results. In fact, for almost all configurations, the required sample size is close to the minimal sample size, which highlights the fast convergence of the targeted conformance measures. Only for *d* and *da* for RTF and RTFr larger samples are generated, due to the lower convergence rate. Here, the low number of deviations occuring in these logs causes the few traces containing deviations to induce significant changes in the relative deviation distribution, if they are sampled, thus effectively increasing the sample size.

The small samples also lead to considerable gains for the runtime of our approach, as shown in Fig. 3. We observe that our approach is able to obtain fitness results in about 5.0% of the time for BPI-12, 0.06% for BPI-14, 41.5% for RTF and 51.2% for RTFr. These gains are slightly smaller when considering the deviation distribution for the BPI-12 and BPI-14 logs. In contrast, for RTF and RTFr the runtime of *d* and *da* actually exceeds the runtime of the baseline. This is due to the fact that these evaluation scenarios are very simplistic and, hence, do not impose computational challenges. Specifically, the RTF traces are very short (3.7 events on average), the associated process models have a simple structure, and there is a low number of trace variants (231). Exploiting parallelization on the aforementioned NUMA server, the runtime for RTF and RTFr becomes negligible, since the baseline only requires about 300ms. As such, any overhead induced by an incremental update to the deviation distributions does not amortize.

When considering the impact of approximation, we observe that the *fa* and *da* configurations are typically faster than their counterparts without approximation. The biggest gain is observed for *fa* on the BPI-12 log, where the relative runtime is reduced from 5.0% to only 3.3%. However, we do note that the inclusion of approximation can also lead to an increased runtime. This can be observed for fitness approximation (*fa*) in the BPI-14 log and for deviation distribution approximation (*da*) for the RTF case. In these cases, the additional overhead required to approximate the impact of a trace on the conformance result may outweigh the time required to compute an alignment.

Accuracy. The observed gains in efficiency are obtained while maintaining highly accurate conformance results. According to both Fig. 4, the fitness computed using sample-based conformance checking differs by less than 0.1% from the original fitness (indicated by the dashed, blue line).

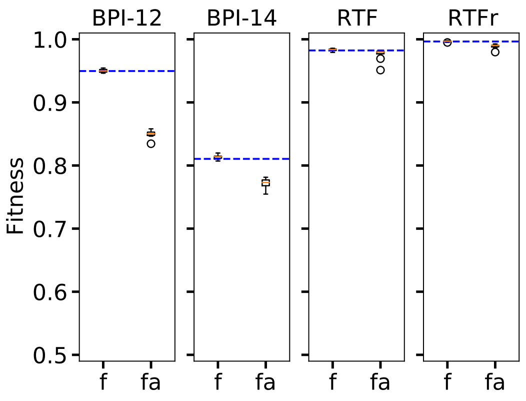


Figure 4: Result accuracy (fitness). Dashed, blue lines denote fitness values obtained for the complete log

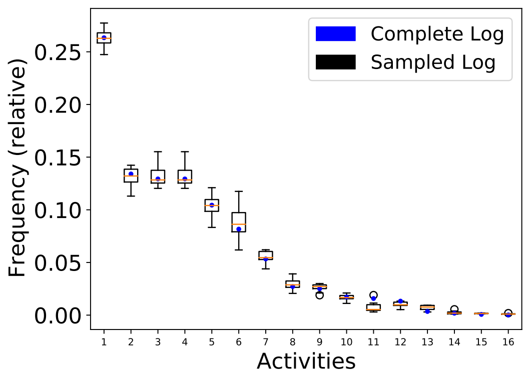


Figure 5: Result accuracy (deviation distribution) for BPI-12. Blue points denote deviation frequencies as observed in the complete log

Since accuracy in terms of deviation distribution is hard to capture in a single value, we use Fig. 5 to demonstrate that the deviation distributions obtained by our sample-based technique closely follow the distribution for the complete BPI-12 log. In decreasing order Fig. 5 depicts the activities with their relative deviation frequency observed in the complete log and in the sampled log. As shown, our technique clearly identifies the activities that are most often affected by deviations, i.e., our technique correctly identifies the main hotspots of non-conformance. Note that the accuracy of the *da* approach is not depicted here, but shall be explored in detail in Section 6.3.4. Finally, we highlight that the results for the other datasets (available online at the aforementioned repository) confirm the observed trends.

6.3.2. Parameter Sensitivity

We analysed the sensitivity of our proposed approach to differing parameter values for the BPI-12, BPI-14, and Traffic Fines (RTF and RTFr) cases. While we show the

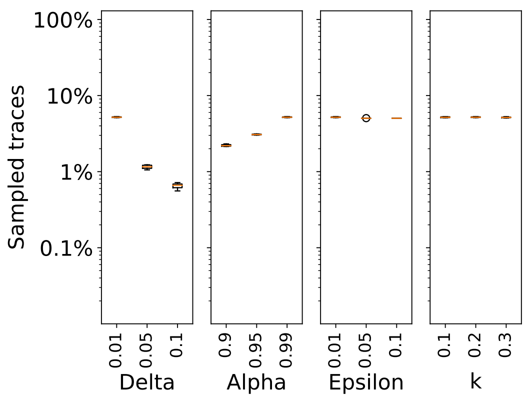


Figure 6: Effect of parameter values on the number of sampled traces for BPI-12

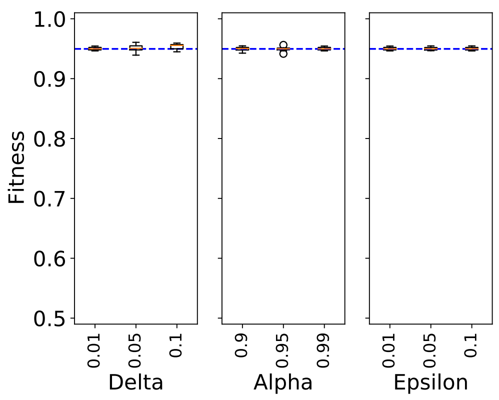


Figure 7: Effect of parameter values on the obtained fitness for BPI-12 using sampling. Dashed, blue lines denote the fitness value obtained for the complete log

results for BPI-12 in detail, the reported trends turned out to be consistent across all cases (and, again, the results can be found in the accompanying repository).

Sampling approach. To determine the impact of the parameters on the sampling approach, we explored how parameters δ (probability bound), α (significance value) ϵ (relaxedness value), and k (approximation pruning parameter) affect the performance of our approach in terms of efficiency (number of traces) and accuracy (fitness).[2]

Fig. 6 shows that the selection of δ and α has a considerable impact on the sample required for conformance checking. For instance, for $\delta = 0.01$ we require an average sample size of 680.4, whereas when we relax the bound to $\delta = 0.10$, the average size is reduced to 85.1 traces. For α, i.e., the confidence, we observe a range from 289.5 to 680.4 traces. By contrast, varying epsilon is shown to result in only marginal differences (ranging between 661.1 and 696.0). Still, for all these results, it should be considered that even the largest sample sizes represent only 5% of the traces in the original log.

Notably, as shown in Fig. 7, the average accuracy of the sample-based approach remains highly stable throughout this sensitivity analysis. However, we do observe that the variance across replications differs for the parameter settings using smaller sample sizes, specifically for $\delta = 0.1$. Here, the obtained fitness values range between 0.945 and 0.959. This indicates that for such sample sizes, the selection of the particular sample may impact the obtained conformance result in some replications.

Approximation approach. By contrast, when incorporating approximation alongside sampling, we observe that the parameter settings allow users to trade-off result accuracy against computational efficiency. In particular, as shown in Fig. 8, we observe a loss in accuracy for smaller values of δ, larger values of α, and larger values of k. The size of this accuracy loss differs per case, though. For instance, comparing the results shown here for the relatively complex model and log of BPI-12 with those of RTF and RTFr (available online), we note that approximation errors become smaller for the latter cases, which are significantly simpler. For BPI-12, see Fig. 8, the mean fitness ranges from 0.835 for $\delta = 0.01$ to 0.909 for $\delta = 0.1$. For δ and α this quality loss indicates, that the accuracy gains obtained for larger sample sizes in the sample-based approach are lost due the introduction of the approximation error during approximation. This error accumulates for larger sample sizes (i.e smaller δ and larger α), as more trace variants are explored, which are then approximated. Thus, for approximation, more accurate results are expected for smaller sample sizes.

Regarding the pruning parameter k, we observe a loss in accuracy, ranging from an average of 0.925 for $k = 0.1$ to 0.769 for $k = 0.3$, as well as an increase in variance for larger values. We also note that, while the pruning parameter k has an impact on the accuracy, it has no impact on the number of sampled traces, as seen in Fig. 6, as it is only used to assess whether the conformance result of a trace may be approximated or not.

6.3.3. *Scalability*

The results obtained for the synthetic datasets confirm that our approaches are able to provide highly accurate conformance checking results in a small fraction of the runtime. Here, we reflect on experiments performed using our fitness-based sampling approach with $\delta =$

[2]While keeping the other parameters at their respective defaults.

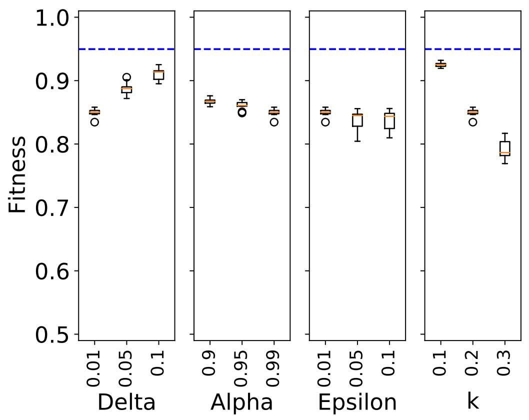


Figure 8: Effect of parameter values on the obtained fitness for BPI-12 using sampling and approximation. Dashed, blue lines denote the fitness value obtained for the complete log

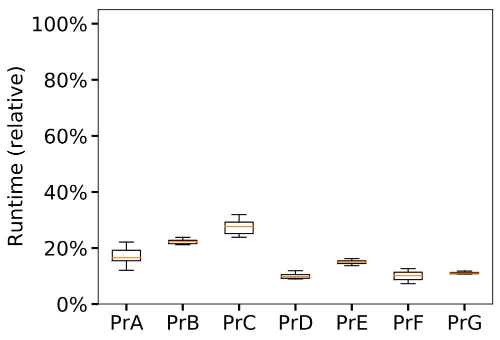


Figure 9: Relative runtime of the sample-based approach for fitness computation obtained for the synthetic data collection

0.05, $\alpha = 0.99$, and $\epsilon = 0.01$. Fig. 9 shows that, for all seven cases, runtime is reduced to ranges between 10.0% and 27.5% of the time needed for the total log (sample sizes range from 10.67% to 30.12%). At the same time, for all cases, the obtained fitness results are virtually equivalent to those of the total log, see Fig. 10, where the fitness values of the total logs are given by blue crosses. When comparing these results to those of the real-world datasets, it should be noted that the synthetic logs hardly have any re-occurring trace variants, which makes it harder to generalize over the sampled results. Still, overall, the results on the synthetic data demonstrate that our approach is beneficial in highly complex scenarios.

6.3.4. *Novel Approximation Heuristics*

In the context of the approximation of deviation distributions, Section 4.3.2 describes three heuristics for the selection of deviating activities in the currently sampled trace, of which two are novel in comparison to our earlier work [19]. In this section, we evaluate their efficiency and accuracy for different values of the pruning parameter k. Again, we rely on the BPI-12 log for illustration purposes, whereas results for the other datasets are available online.

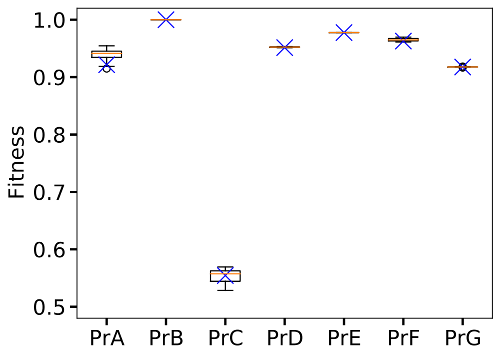


Figure 10: Result accuracy (fitness) obtained for the synthetic data collection. Blue crosses denote fitness obtained for the complete log

In Fig. 11, the aggregated runtime measurements for the three heuristics are depicted compared to the average mean runtime of conformance checking without approximation. Specifically, we observed that for Heuristic 1 no value of k yields runtime improvements when compared to the sample-based approach. Even though a larger k indicates that more trace variants are approximated, this comes at the cost of the aforementioned exponential blow-up during the creation of the bags of deviating activities. Thus, the potential runtime improvements of larger k are neutralized by this overhead for larger traces. For the alignment-based heuristics, we observe runtime reductions for larger k that are somewhat consistent between both Heuristic 2 and 3. On the contrary, for the simple cases, RTF and RTFr, Heuristic 2 is not able to provide any runtime improvement, whereas Heuristic 3 results in steady runtime improvements for larger k. The reason being that the alignment-based heuristic incorporating all deviations needs to incorporate more activities into the relative distribution, which ultimately results in a lack of runtime gains for larger values of k, due to the increased update overhead. The results in Fig. 8 and Fig. 11, again emphasize that the value of k provides a trade-off between accuracy and runtime.

Also, we note that low values of k result in worse run times than without approximation. In these cases, fewer traces are approximated, thus alignment calculation is employed most of the time on top of the overhead required to select trace variants and performing the approximation.

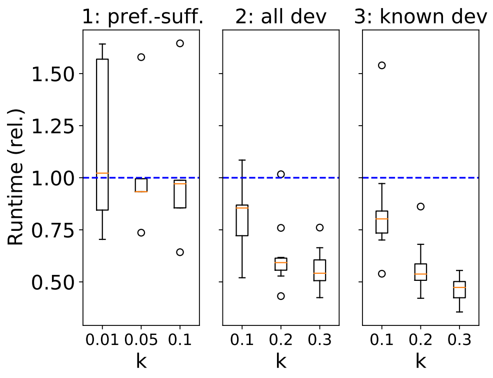


Figure 11: Runtime sensitivity of approximation heuristics for varying k compared to the sample-based approach for BPI-12

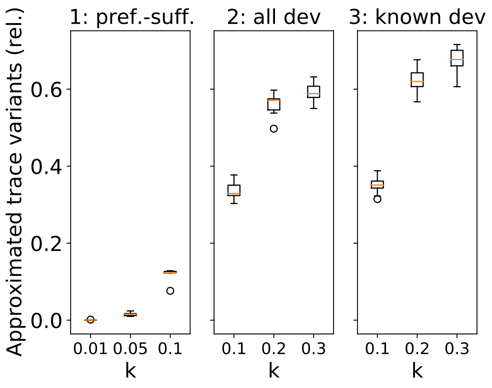


Figure 12: Fraction of sampled trace variants that are approximated for varying k for BPI-12

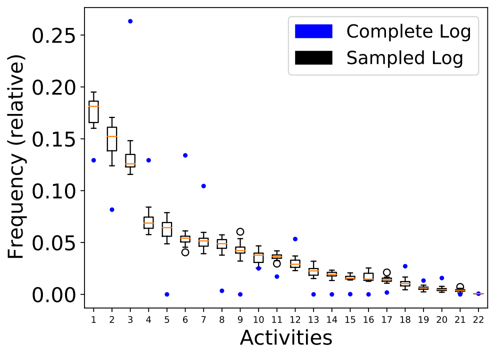


Figure 13: Result accuracy of Heuristic 2 (deviation distribution) for BPI-12

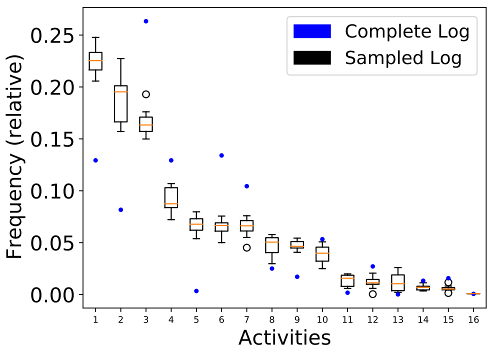


Figure 14: Result accuracy of Heuristic 3 (deviation distribution) for BPI-12

Furthermore, in Fig. 12, we depict the fraction of the sampled trace variants that our approach approximated instead of aligned. For $k = 0.2$, on average 62% of approximated trace variants signal significant change for Heuristic 3, while only 56% signal a significant change for Heuristic 2. Likewise, for $k = 0.3$, on average 67% of approximated trace variants signal a significant change for Heuristic 2, while this value increases only to 59% for Heuristic 3. To assess the accuracy of the two, new alignment-based heuristics, Fig. 13 and Fig. 14, respectively, depict the activities and their deviation frequencies for the complete and sampled logs using $k = 0.2$. The figures show the improvement that Heuristic 3 provides over Heuristic 2. The latter heuristic estimates deviations for a considerable number of activities that never deviate, whereas Heuristic 3 avoids this issue and, thus obtains more accurate results.

6.3.5. *Resource Deviation Detection*

To assess the efficacy of our sampling and approximation techniques when incorporating contextual factors, we consider the detection of resource deviations and evaluate it based on two recall values: (1) the fraction of the unique resources involved in deviations our technique detected and (2) the fraction of total deviations that these resources were involved in. Given that there are no resource schedules available for any of the real-world event logs (M_{res}), we generated these using an 80/20 principle (similar as used for the inductive miner to obtain process models). In particular, for each activity, we computed the average number of activity executions individual resources were associated with in an event log. Given this average, we marked any resource performing this activity fewer than 20% of the average as an unauthorized resource, whereas the others were considered to be part of M_{res} for that particular activity. In the BPI-12 log, there are 68 unique resources recorded, which are together involved in a total of 896 activity-resource combinations, with a minimum of 1 resource per activity and an observed maximum of 60. By using a 20% threshold to identify authorized resources, we retain a total of 602 combinations in the resource authorization function M_{res}, with a minimum of 1 authorized resource per activity and a maximum of 49 authorized resources.

Table 2: Average results obtained for the detection of resource deviations (BPI-12 log)

Config.	Traces	Recall (uniq.)	Recall (total)
S	4,974.4 (38.0%)	72.8%	97.2%
S+A	3,983.9 (30.0%)	55.6%	94.8%

The main results when detecting resource deviations based on this established authorization are presented in Table 2, where *S* denotes the results obtained using sampling and *S+A* those obtained using both sampling and approximation. We observe that the sample sizes required here are slightly larger than for the conformance checking based on fitness and deviation distribution, requiring samples of about 38.04% of the traces, rather than the 7.5% for the deviation distribution. This is in line with expectations, given the monotonic nature of this conformance measure. These samples contain on average 72.8% of the resources involved in deviations for *S* and 55.6% for *S+A*. Furthermore, we recognize that there is a considerable long tail of resources involved in only a small amount of deviations. This becomes clear when looking at the fraction of the total deviations that these resources are involved in. In particular, for *S*, these 72.8% sampled deviating resources are involved in more than 97% of all resource deviations. Likewise, for *S+A*, these 55.6% of sampled deviating resources are involved in 94% of deviations. This means that by sampling less than 30% of the traces, we are already able to detect more than half of the violating resources that, also, provide a near-complete coverage of the total resource deviations.

6.3.6. *Quality Checking Methods*

We performed experiments to assess the impact of both the internal and external quality methods proposed in Section 5, each involving various configurations. Again, we use the BPI-12 case for illustration, whereas the results for the other cases are available online. While trends are mostly consistent, we discuss differences explicitly.

Internal quality checking. To assess the impact of the proposed methods internal quality checking for five configurations:

1. Behavioural representativeness according to the directly-follows relation (configuration *DF*).
2. Behavioural representativeness according to the dependency measure relation (configuration *DM*).
3. Data attribute coverage based on the `org:resource` attribute associated with the completion activity, `W_completeren aanvraag` (configuration *attribute*).
4. Both directly follows and attribute-based quality checking (configuration *DF + attribute*).
5. Both dependency measure and attribute-based quality checking (configuration *DM + attribute*)

The results obtained for these configurations, as well as a baseline without quality checks are depicted in Table 3.

The sample sizes depicted in the table clearly show that all configurations of the internal quality checking mechanism have an impact on the conformance checks performed: specifically, for the setting with no quality checking, on average 681 traces are sampled, whereas this number increases to up to 828.5 for the *DM + attribute* configuration. These greater sample sizes lead to increased runtimes (from 7.0s on average to up to 9.1s), though they also improve the results accuracy, as can be seen by the *fitness error*, which is reduced from 0.00223 (on average) to 0.00146 for the last configuration.

However, comparing the sample sizes of BPI-12 and RTFr, we notice that those for RTFr are significantly larger, showing that the effect of quality checking is highly dependent on the convergence of an analysed distribution and, thus, input-dependent. For instance, while the DM configuration results in an increased sample size of 92.3 traces (on average) for BPI-12, it leads to an increase of 1,856.2 traces for RTFr.

Furthermore, we can observe that the two configurations that include the dependency measure (DM) have a larger impact on the sample size than those only incorporating the directly-follows relations (DF), which means that this measure converges less quickly. The main cause for this is that the normalization of the dependency measure does not consider the number of times a certain directly-follows relation has been observed. As a result this measure leads to more triggers of the quality checking mechanism than the other configurations.

External quality checking. We assessed the external quality checking methods in the same manner as for the internal checks. However, the configurations using the dependency measure (DM) cannot be applied here, given that this measure is not suitable for the statistical tests required to perform external quality checks.

Table 4 depicts the results obtained given $\alpha = 0.05$. Here, we observe that the external quality checking methods were triggered various times, resulting in increased sample sizes for many configurations (and replications). For instance, we obtain sample sizes that are on average more than 10 times as large than for the configuration without quality checking when we incorporate an analysis of attribute values. These larger sample sizes are

Table 3: Results obtained using internal quality checking methods for sample-based fitness computation (BPI-12 log)

Configuration	**Sample size**		**Runtime**		**Fitness error**	
	avg.	**max.**	**avg.**	**max.**	**avg.**	**max.**
No q. checking	681.0	692	7.0s	9.0s	0.00223	0.00507
DF	690.9	754	7.4s	10.3s	0.00197	0.00482
DM	773.3	941	8.5s	12.1s	0.00205	0.00314
Attribute	762.9	886	8.2s	10.8s	0.00195	0.00342
DF + attribute	788.5	889	8.4s	11.0s	0.00159	0.00475
DM + attribute	828.5	949	9.1s	13.4s	0.00146	0.00242

Table 4: External quality checking results for sample-based fitness computation (BPI-12 log, $\alpha = 0.05$)

Configuration	**Sample size**		**Runtime**		**Fitness error**	
	avg.	**max.**	**avg.**	**max.**	**avg.**	**max.**
No q. checking	681	692	7.0s	9.0s	0.00223	0.00507
DF	1,139.9	3,970	12.0s	37.7s	0.00213	0.00369
Attribute	8,266.8	9,877	84.6s	98.0s	0.00052	0.00125
DF + attribute	7,251.2	9,230	74.2s	97.3s	0.00049	0.00117

also mirrored by proportionally larger runtimes, e.g., up to 98.0s for the longest run, versus 9.0 seconds without quality checking. While we do observe that, naturally, such larger sample sizes lead to lower fitness errors, it is also important to note that the gains are marginal in absolute terms, whereas the relative gain does not increase linearly along with the runtime. For example, an average runtime that is more than 12 times as large (7.0s versus 84.6s) results in an average error that is only about 4 times as low (0.00223 versus 0.00052).

Overall, the quality checking results, especially those obtained using the external quality checking methods, provide some interesting insights. First, one can observe that by incorporating the quality checking methods, one can obtain slightly more accurate results for an increase in runtime. However, it is important to note that the results obtained using just sampling are already highly accurate, whereas the accuracy gains resulting from quality checking are not necessarily proportional to the additional runtime that is required. Second, it is also interesting to observe that there is not necessarily a correlation between, for instance, the distribution of attribute values and conformance checking results. In particular, even when external quality checking methods conclude that the distribution of attribute values may not be statistically representative, the obtained conformance checking results nonetheless are highly accurate. Still, when users have domain knowledge about the importance of a certain attribute or want to ensure that conformance checking results are obtained on the basis of a sample that is representative with respect to such an attribute, the quality checking methods allow one to obtain this insurance in a convenient manner.

6.3.7. Recap

Finally, we bring everything together with a brief recap that compares the impact of the main configurations that our approach provides for all the considered cases, BPI-12, BPI-14, RTF, and RTFr. In particular, our approach includes techniques for:

- *Trace Sampling*, which determines when sufficient traces have been observed to provide reliable results,
- *Approximation*, to further reduce the runtime by approximating results for specific traces instead of computing expensive alignments, and
- *Quality Checking*, to increase the results accuracy, by ensuring the conformance results are provided on the basis of a representative sample.

Trace sampling represents the core notion of our approach and is therefore always incorporated in a configuration. Then, users can optionally choose to add approximation and/or quality checking. Where, generally, approximation can improve the computational efficiency, though it may reduce the result accuracy. The incorporation of quality checking, by contrast, generally has the reverse effects. To highlight these trade-offs, we here report on results obtained for the various main configurations of our approach for the four inputs, as shown in Table 5 for fitness-based conformance checking. The table presents the results for the baseline algorithm, the standard configuration with only sampling (*S*), sampling and approximation (S+A), sampling with internal quality checking[3] (S+Q), and both approximation and quality checking (S+A+Q).

First, we observe that for BPI-12 and BPI-14 all combinations of techniques result in runtimes that are a fraction of the runtime achieved by the baseline algorithm. In contrast, for RTF and RTFr the incorporation of quality checking methods results in runtimes that are significantly higher than those obtained by applying the baseline conformance checking method. This again highlights that the overhead of our techniques can outweigh the benefits for such simplistic cases where runtime is not an issue in general.

Impact of approximation. Furthermore, the results show that the incorporation of approximation noticeably decreases the runtime whilst increasing the mean error in accuracy for those inputs, with a moderate amount of trace variants and average trace length. For these inputs, we expect, a reasonably fast selection of a k-similar trace variant and subsequent higher fraction of approximated traces. For BPI-14 we have a large number of trace variants, as well as long traces, thus increasing the overhead introduced by the approximation schemes.

Impact of quality checking. In contrast to approximation, the incorporation of quality checking methods mitigates the error, while increasing the runtime, consistently for all inputs. This can also be seen in the maximum sample sizes for the settings that incorporate quality checking, i.e. for S+Q and S+A+Q. For these configurations, the maximum sample size observed in the repetitions is noticeably larger than for those configurations without quality checking. Here, the quality checking ensures a more unbiased result, by enforcing larger samples.

For the case, in which both approximation and quality checking were used (S+A+Q), we can observe that the negative accuracy impact of approximation outweighs the positive impact of the quality checking methods, while the runtime and sample size is bounded by the quality checking methods. Here, the increased sample size enforced by the quality checking come with a proportionally increasing approximation error. This ultimately leads to runtimes comparable to the runtimes obtained by configuration S+Q, as well as higher error rates than configuration S+A. As a result, this configuration has a lower accuracy than the S configuration, though it is also faster.

Runtime versus accuracy. These results clearly show the applicability of the sampling procedure for a significant reduction in runtime under acceptable error rates. Furthermore, the results illustrate the flexibility that our approach provides to users, who can choose to incorporate approximation when runtime is more important or to add quality checking when result accuracy is key. As shown in Section 6.3.2 and Section 6.3.6, this trade-off can be further tuned by altering the parameters of the individual components. For instance, by increasing the value for k in the approximation component, one can further trade-off result accuracy against more efficiency gains.

7. Related Work

Conformance checking can be grounded in various notions. Non-conformance may be detected based on a comparison of sets of binary relations defined over events of a log and activities of a model, respectively [38]. Other work suggests to 'replay' traces in a process model, thereby identifying whether events denote valid activity executions [39]. However, both of these streams have limitations with respect to completeness [2]. Therefore, alignment-based conformance checking techniques [5, 13], on which we focus in this work, are widely recognized as the state-of-the-art. A considerable downside of these techniques is their computational complexity.

Acknowledging this, various approaches [7, 6, 9, 40], discussed in Section 1, have been developed to improve the runtime efficiency of alignment computation. Other works aim to achieve efficiency gains by sacrificing accuracy through alignment approximation [41]. These include techniques that approximate total sets of optimal alignments [42] and recent work that uses relaxation labeling to obtain approximate results [43]. While these approaches also lead to efficiency gains, their goal and methods are fundamentally different from ours. In particular, these approaches aim to approximate individual trace alignments, whereas our approach estimates a conformance result for an entire event log. As a result, the runtime gains that our approach is able to achieve, possibly reducing the runtime to less than 1% of the original, are of a different order of magnitude than those of methods that aim to approximate alignments. However, any method that more efficiently obtains alignments (whether approximated or not) should be regarded as complementary to our work, since such methods can be incorporated in our framework.

The sampling technique that we employ is based on sampling used in sequence databases, i.e., datasets that contain traces. Sampling techniques for event logs have

[3]The configuration here corresponds to the DM + attribute configuration discussed in Section 6.3.6

Table 5: Overview of the main results obtained using various combinations of the proposed contributions.

Case	Configuration	Sample size avg.	Sample size max.	Runtime avg.	Runtime max.	Fitness error avg.	Fitness error max.
BPI-12	Baseline	13,087.0	13,087	147.3s	156.3s	-	-
	S	681.0	691	7.4s	10.7s	0.00219	0.00476
	S+A	680.4	693	4.8s	6.1s	0.10023	0.11516
	S+Q	828.5	949	9.1s	13.4s	0.00146	0.00242
	S+A+Q	808.3	1061	5.0s	6.6s	0.10281	0.1154
BPI-14	Baseline	46,616.0	46,616	2,315.2s	2,415.6s	-	-
	S	680.1	702	14.5s	23.3s	0.00374	0.00927
	S+A	679.6	702	17.2s	24.2s	0.03914	0.05578
	S+Q	865.2	971	18.2s	27.5s	0.00303	0.00643
	S+A+Q	843.6	932	19.0s	27.8s	0.04307	0.05893
RTF	Baseline	150,370.0	150,370	0.379s	0.499s	-	-
	S	660.2	666	0.157s	0.272s	0.00184	0.00339
	S+A	660.3	669	0.149s	0.298s	0.00629	0.03115
	S+Q	20,822.1	37,894	120.574s	138.718s	0.00044	0.00124
	S+A+Q	22,225.7	38,067	125.137s	138.137s	0.02068	0.13018
RTFr	Baseline	150,370.0	150,370	0.244s	0.29s	-	-
	S	658.0	658	0.125s	0.20s	0.00079	0.00156
	S+A	658.0	658	0.097s	0.15s	0.00774	0.01679
	S+Q	20,056.8	32,654	118.136s	129.0s	0.00012	0.00029
	S+A+Q	27,466.4	36,161	126.815s	139.3s	0.01205	0.02322

been previously applied for specification mining [44], for mining of Markov Chains [45], and for process discovery [46, 21]. This focus on sampling has also yielded various works that assess the samples themselves, as well as their impact on various tasks [47]. This includes completeness assessment techniques [24, 25, 26] similar to those employed in Section 6.3.6. Despite such earlier works, we are the first to apply these sampling techniques to conformance checking, a use case in which computational efficiency is arguably even more important than in discovery scenarios. Following up on our original work [19], we have successfully lifted the notions underlying our approach for efficient conformance checking in the presence of ordering uncertainty [48].

8. Conclusion

This paper advocates the usefulness of sampling and approximation in the context of the computationally expensive conformance checking task. Our focus is on application scenarios that require insights into the overall conformance of an event log with respect to a process model, i.e., where aggregated conformance results instead of single deviations are of interest. For such a scenario, we argued that conformance results can be obtained without computing conformance results for *all* traces. Specifically, we considered two angles to achieve efficient conformance checking: First, through trace sampling, we achieve that only a small share of the traces of a log are considered in the first place. By phrasing this sampling as a series of random experiments, we are able to give guarantees on the introduced error in terms of a potential difference of the overall conformance result. Second, we introduced result approximation as a means to avoid the computation of conformance results even for some of the sampled traces. Exploiting similarities of two traces, we derive an upper bound for the conformance of one trace based on the conformance of another trace. Both techniques, trace sampling and result approximation, have been instantiated for three notions of conformance results: fitness as a numerical measure of overall conformance, the deviation distribution that highlights hotspots of non-conformance in terms of individual activities, and deviations that incorporate the execution context into conformance checking. We illustrated the latter for resource assignments in particular. Moreover, to increase the robustness of our approach, we introduced mechanisms to reveal and counteract biases in log samples.

To assess the efficiency and accuracy of our proposed techniques, we conducted comprehensive evaluation experiments using both real-world and synthetic data sets.

The experimental results highlight dramatic improvements in terms of conformance checking efficiency: Only a fraction of the traces of event logs are required in order to obtain virtually equivalent conformance results to those obtained without sampling. As such, we are able to gain orders of magnitude improvements in the runtime of conformance checking, without sacrificing the accuracy of the overall conformance results. The experiments also revealed that our approximation technique has considerably improved in comparison to its previous version, presented in [19], while furthermore covering a broader scope of conformance results.

Acknowledgment

This work has received funding from the Deutsche Forschungsgemeinschaft (DFG), grant number 421921612, and the Alexander von Humboldt Foundation.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Second Edition, Springer, 2018. doi:10.1007/978-3-662-56509-4.

[2] J. Carmona, B. van Dongen, A. Solti, M. Weidlich, Conformance Checking – Relating Processes and Models, Springer, 2018.

[3] B. Weber, M. Reichert, S. Rinderle-Ma, Change patterns and change support features - enhancing flexibility in process-aware information systems, DKE 66 (3) (2008) 438–466. doi:10.1016/j.datak.2008.05.001.

[4] W. M. P. Van der Aalst, Data Scientist: The Engineer of the Future, Springer International Publishing, Cham, 2014, pp. 13–26. doi:10.1007/978-3-319-04948-9_2.

[5] A. Adriansyah, B. van Dongen, W. M. P. Van der Aalst, Conformance checking using cost-based fitness analysis, in: EDOC, 2011, pp. 55–64.

[6] D. Reißner, R. Conforti, M. Dumas, M. L. Rosa, A. Armas-Cervantes, Scalable conformance checking of business processes, in: OTM to Meaningful Int. Syst., 2017, pp. 607–627. doi:10.1007/978-3-319-69462-7_38.

[7] B. F. van Dongen, Efficiently computing alignments - using the extended marking equation, in: Business Process Management, 2018, pp. 197–214. doi:10.1007/978-3-319-98648-7_12.

[8] M. de Leoni, A. Marrella, How planning techniques can help process mining: The conformance-checking case, in: Italian Symp. on Advanced Database Syst., 2017, p. 283.

[9] J. Evermann, Scalable process discovery using map-reduce, IEEE TSC 9 (3) (2016) 469–481. doi:10.1109/TSC.2014.2367525.

[10] C. Luo, F. He, C. Ghezzi, Inferring software behavioral models with MapReduce, Sci. Comput. Program. 145 (2017) 13–36. doi:10.1016/j.scico.2017.04.004.

[11] F. Taymouri, J. Carmona, A recursive paradigm for aligning observed behavior of large structured process models, in: Business Process Management, Springer, 2016, pp. 197–214. doi:10.1007/978-3-319-45348-4_12.

[12] W. M. P. V. der Aalst, H. M. W. Verbeek, Process discovery and conformance checking using passages, Fundam. Inform. 131 (1) (2014) 103–138. doi:10.3233/FI-2014-1006.

[13] J. Munoz-Gama, J. Carmona, W. v. d. Aalst, Single-entry single-exit decomposed conformance checking, Inf. Syst. 46 (2014) 102–122.

[14] S. J. J. Leemans, D. Fahland, W. M. P. V. der Aalst, Scalable process discovery and conformance checking, Software and System Modeling 17 (2018) 599–631. doi:10.1007/s10270-016-0545-x.

[15] P. M. Dixit, J. C. A. M. Buijs, H. M. W. Verbeek, W. M. P. V. der Aalst, Fast incremental conformance analysis for interactive process discovery, in: BIS, Springer, 2018, pp. 163–175. doi:10.1007/978-3-319-93931-5_12.

[16] K. M. van Hee, Z. Liu, N. Sidorova, Is my event log complete? - A probabilistic approach to process mining, in: Int. Conf. on Research Challenges in Information Science, 2011, pp. 1–7. doi:10.1109/RCIS.2011.6006848.

[17] W. M. P. Van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, Springer, 2011. doi:10.1007/978-3-642-19345-3.

[18] A. Rogge-Solti, A. Senderovich, M. Weidlich, J. Mendling, A. Gal, In log and model we trust? A generalized conformance checking framework, in: BPM, 2016, pp. 179–196.

[19] M. Bauer, H. van der Aa, M. Weidlich, Estimating process conformance by trace sampling and result approximation, in: International Conference on Business Process Management, 2019, pp. 179–197.

[20] N. Busany, S. Maoz, Behavioral log analysis with statistical guarantees, in: ICSE, ACM, 2016, pp. 877–887.

[21] M. Bauer, A. Senderovich, A. Gal, L. Grunske, M. Weidlich, How much event data is enough? A statistical framework for process discovery, in: CAiSE, 2018, pp. 239–256. doi:10.1007/978-3-319-91563-0_15.

[22] S. B. Needleman, C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of molecular biology 48 (3) (1970) 443–453.

[23] R. P. J. C. Bose, W. M. P. van der Aalst, Process diagnostics using trace alignment: Opportunities, issues, and challenges, Inf. Syst. 37 (2) (2012) 117–141. doi:10.1016/j.is.2011.08.003.
URL https://doi.org/10.1016/j.is.2011.08.003

[24] B. Knols, J. M. E. M. van der Werf, Measuring the Behavioral Quality of Log Sampling, in: 2019 International Conference on Process Mining (ICPM), IEEE, 2019, pp. 97–104. doi:10.1109/ICPM.2019.00024.

[25] L. Bao, N. Busany, D. Lo, S. Maoz, Statistical Log Differencing, in: Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering, 2019, pp. 851 – 862.

[26] H. Yang, L. Wen, J. Wang, An approach to evaluate the local completeness of an event log, Proceedings - IEEE International Conference on Data Mining, ICDM (2012) 1164–1169doi:10.1109/ICDM.2012.66.

[27] M. Weidlich, A. Polyvyanyy, J. Mendling, M. Weske, Causal behavioural profiles–efficient computation, applications, and evaluation, Fundamenta Informaticae 113 (3-4) (2011) 399–435.

[28] A. Polyvyanyy, M. Weidlich, R. Conforti, M. L. Rosa, A. H. M. ter Hofstede, The 4c spectrum of fundamental behavioral relations for concurrent systems, in: Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, 2014, pp. 210–232. doi:10.1007/978-3-319-07734-5_12.

[29] M. L. McHugh, The chi-square test of independence, Biochemia medica: Biochemia medica 23 (2) (2013) 143–149.

[30] N. M. Razali, Y. B. Wah, et al., Power comparisons of shapiro-

wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests, Journal of statistical modeling and analytics 2 (1) (2011) 21–33.
[31] B. Van Dongen, BPI Challenge 2012, https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f (2012). doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
[32] B. Van Dongen, BPI Challenge 2014, https://doi.org/10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35 (2014). doi:10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35.
[33] M. M. De Leoni, F. F. Mannhardt, Road traffic fine management process, https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5 (2015). doi:10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5.
[34] S. J. J. Leemans, D. Fahland, W. M. P. V. der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: BPM Workshops, Vol. 171 of LNBIP, Springer, 2013, pp. 66–78.
[35] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. V. der Aalst, Balanced multi-perspective checking of process conformance, Computing 98 (4) (2016) 407–437. doi:10.1007/s00607-015-0441-1.
URL https://doi.org/10.1007/s00607-015-0441-1
[36] Munoz-Gama, J., Conformance checking in the large (dataset) (2013). doi:10.4121/UUID:44C32783-15D0-4DBD-AF8A-78B97BE3DE49.
[37] E. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, W. M. P. V. der Aalst, Prom 6: The process mining toolkit, in: Business Process Management (Demonstration Track), 2010.
[38] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, M. Weske, Process compliance analysis based on behavioural profiles, Inf. Syst. 36 (7) (2011) 1009–1025.
[39] A. Rozinat, W. M. P. Van der Aalst, Conformance checking of processes based on monitoring real behavior, Information Systems 33 (1) (2008) 64–95.
[40] F. Taymouri, J. Carmona, Model and event log reductions to boost the computation of alignments, in: SIMPDA, 2016, pp. 1–21.
[41] B. F. van Dongen, J. Carmona, T. Chatain, F. Taymouri, Aligning modeled and observed behavior: A compromise between computation complexity and quality, in: CAISE, Vol. 10253, 2017, pp. 94–109.
[42] F. Taymouri, J. Carmona, An evolutionary technique to approximate multiple optimal alignments, in: BPM, 2018, pp. 215–232.
[43] L. Padró, J. Carmona, Approximate computation of alignments of business processes through relaxation labelling, in: International Conference on Business Process Management, Springer, 2019, pp. 250–267.
[44] H. Cohen, S. Maoz, Have we seen enough traces?, in: ASE, IEEE, 2015, pp. 93–103.
[45] A. W. Biermann, J. A. Feldman, On the synthesis of finite-state machines from samples of their behavior, IEEE Trans. Computers 100 (6) (1972) 592–597.
[46] J. Carmona, J. Cortadella, Process mining meets abstract interpretation, in: ECML/PKDD (1), Vol. 6321 of Lecture Notes in Computer Science, Springer, 2010, pp. 184–199.
[47] M. F. Sani, S. J. van Zelst, W. M. P. van der Aalst, The impact of event log subset selection on the performance of process discovery algorithms, in: New Trends in Databases and Information Systems ADBIS, 2019, pp. 391–404.
[48] H. van der Aa, H. Leopold, M. Weidlich, Partial order resolution of event logs for process conformance checking, Decision Support Systems 136 (2020) 113347.