Model Independent Error Bound Estimation for Conformance Checking Approximation

Mohammadreza Fani Sani[1], Martin Kabierski[2], Sebastiaan J. van Zelst[3,1], and Wil M.P. van der Aalst[1,3]

[1]Process and Data Science Chair, RWTH Aachen University, Aachen, Germany
[2]Department of Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany
[3]Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany
{fanisani,s.j.v.zelst,wvdaalst}@pads.rwth-aachen.de,
martin.bauer@hu-berlin.de

Abstract Conformance checking techniques allow us to quantify the correspondence of a process's execution, captured in event data, w.r.t., a reference process model. In this context, alignments have proven to be useful for calculating conformance statistics. However, for extensive event data and complex process models, the computation time of alignments is considerably high, hampering their practical use. Simultaneously, it suffices to approximate either alignments or their corresponding conformance value(s) for many applications. Recent work has shown that using subsets of the process model behavior leads to accurate conformance approximations. The accuracy of such an approximation heavily depends on the selected subset of model behavior. Thus, in this paper, we show that we can derive a priori error bounds for conformance checking approximation based on arbitrary activity sequences, independently of the given process model. Such error bounds subsequently let us select the most relevant subset of process model behavior for the alignment approximation. Experiments confirm that conformance approximation accuracy improves when using the proposed error bound approximation to guide the selection of relevant subsets of process model behavior.

Keywords: Process mining · Conformance checking approximation · Alignments · Edit distance · Instance selection · Sampling

1 Introduction

The execution of processes in companies leaves digital event data footprints in the databases of the information systems employed, known as *event logs*. *Process mining* [1] aims to develop techniques that enhance the overall knowledge of the process by analyzing such event logs, e.g., by automatically discovering process models based on the event log. *Conformance checking* [2], i.e., one of the main sub-fields of process mining, aims at assessing to what degree a given process model and the recorded event data conform to one another. In this context, alignments [3], an established class of conformance checking artifacts, are of particular interest, as they provide an exact quantification of deviations between
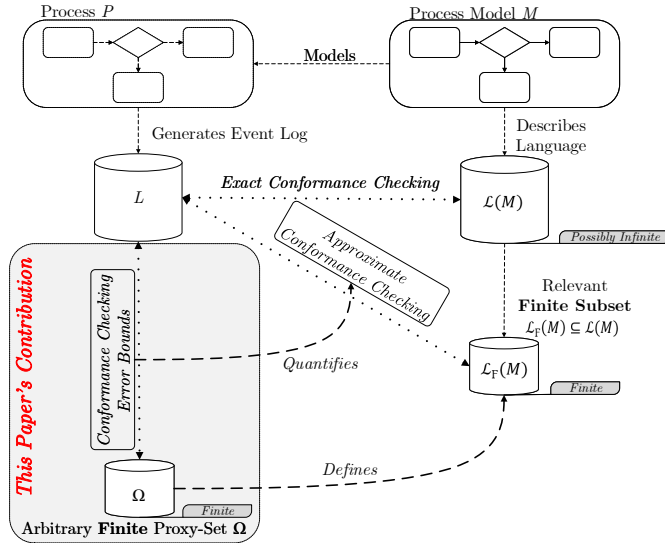
Figure 1: Overview of the proposed approach and its relation to existing work. A process model M models a process P that generates an event log L. Existing approaches either compute *exact* or *approximate conformance checking results* based on the language of the model $\mathcal{L}(M)$ (or a finite subset thereof). We propose a method that quantifies error bounds for conformance checking approximation, based on an arbitrary proxy-set Ω.

the recorded process execution and its intended behavior, as modeled by the process model.

The increasing prevalence of information systems in different domains leads to a drastic increase in the amount of recorded event data [4]. Such *high-volume event data*, combined with complex process models, yield infeasible alignment computation times, hampering their practical use. Yet, in many applications, exact alignment values are not required, i.e., it suffices to obtain an approximated value to draw meaningful conclusions. For example, in genetic process discovery [5], various *generations of candidate process models* are evaluated w.r.t. an event log. Due to the complexity of alignment computation, computing exact alignment results for each candidate process model is impractical. However, in each generation, rather than obtaining an exact alignment result to judge the process model quality, it is sufficient to know whether a newly generated process model improves its alignment results with respect to that of previous generations. Therefore, fast alignment approximation techniques, that provide guarantees on the approximation error are of particular interest.

Recently, various approaches for alignment approximation have been proposed [6, 7]. In our previous work [6], we exploit *subsets of the process model's behavior* for approximation, i.e., by using the subset of process behavior as a

representative for the complete process model behavior. In this way, we are able to provide bounds for the approximated alignment value. This branch of approximation techniques shows promising results; however, the approximation result's quality (i.e., the difference between actual and approximated value) heavily depends on the selected subset of process model behavior. Therefore, quantifying an approximation's quality based on a specific subset of model behavior prior to performing the actual approximation allows us to identify the most appropriate subset to use for said approximation. In this paper, we present a novel approach to quantify an alignment approximations quality before performing the actual approximation.

Consider Fig. 1 (page 2), in which we present a schematic overview of the approach. A process model M models a process P that generates a digital event log L. Existing approaches compute *exact* or *approximate conformance checking results* by considering the language of the model $\mathcal{L}(M)$ (possibly infinite), or, a relevant finite subset thereof. The proposed method allows us to a-priori compute error bounds for alignment approximation, using an arbitrary proxy-set Ω, i.e., a set of sequence of activities. This proxy-set Ω can be used to derive the relevant subset of process model behavior $\mathcal{L}_F(M)$ and may consist of behavior that is not part of the model nor the event log.

We evaluated our error bound estimation technique using various real event logs. Our experiments confirm a strong correlation between the predicted maximum error bounds and the eventual approximation error. As such, our experiments confirm that the conformance approximation accuracy improves when using the proposed error bound approximation to guide the selection of relevant subsets of process model behavior. Furthermore, our experiments show that the error bounds' computation time is negligible w.r.t. computing conventional exact alignments.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. In Section 3, we present preliminaries and basic notation. We explain the main methodology of our approach in Section 4 and subsequently evaluate it in Section 5. Finally, Section 6 concludes this work.

2   Related Work

Several process mining techniques exist, ranging from process discovery to process performance prediction. Here, we cover related work in the field of conformance checking and corresponding approximation techniques. We refer to [1] for an extensive overview and introduction of process mining.

Conformance checking techniques have been well studied. In [8], the authors review the various conformance checking techniques in the process mining domain. Similarly, in [2], different methods for conformance checking and its applications are covered. Alignments were introduced in [9] and have rapidly developed into the standard conformance checking technique. In [10,11], decomposition techniques are proposed for improving the performance of the alignment computation. Applying decomposition techniques improves computation time.

However, these techniques are able to improve the performance of alignment computation, if there are numerous unique activities in the process [12]. Recently, general approximation schemes for alignments, i.e., computation of near-optimal alignments, have been proposed [13]. Finally, the authors in [7] propose to incrementally compute prefix-alignments, i.e., enabling real-time conformance checking for event data streams.

A limited amount of work considers the use of sampling in process mining. In [14], the authors recommend a trace-based statistical sampling method to decrease the required discovery time and memory footprint. Moreover, in [15], we analyzed random and biased sampling methods with which we are able to adjust the size of the sampled data for process discovery.

Some research has focused on alignment approximation. In [16], deep learning is used to approximate alignment statistics. In [17], the authors propose to incrementally sample the event log and applying conformance checking on the sampled data. The proposed method increases the sample size until the approximated value is accurate enough. The authors of [18] propose a conformance approximation method that applies relaxation labeling methods on a partial order representation of a process model, which needs to preprocess the process model each time. Unlike these approaches that do not provide bounds for the approximation, some methods have proposed to generate a subset of model behaviors using instance selection [6] and simulation [19]. [6] has proposed to compute alignments of some instances in the event log and use the corresponding model behavior for approximating the alignment of other instances.

In this paper we prove that by having a subset of model behaviors, we can estimate the approximation error for any process model, thus helping users to adjust the approximation setting. Furthermore, we propose some instance selection methods to decrease approximation error.

3   Preliminaries

This section briefly introduces basic conformance checking terminology and the notation used in this paper. We assume that the reader has a basic knowledge of sets, bags (multisets), Cartesian products functions, and sequences.

We let $\mathcal{B}(X)$ denote the set of all possible bags over X. Given $b \in \mathcal{B}(X)$, $\bar{b} = \{x | b(x) > 0\}$. X^* denotes the set of all sequences over X. Let $X' \subseteq X$ and let $\sigma \in X^*$, $\sigma_{\downarrow_{X'}}$ returns the projected sequence of σ on set X', e.g., $\langle a, b, c, b, d \rangle_{\downarrow_{\{b,d\}}} = \langle b, b, d \rangle$. Let $X_1, X_2, \ldots, X_n$ be n arbitrary sets and let $X_1 \times X_2 \cdots \times X_n$ denote the corresponding Cartesian product. Let $\sigma \in (X_1 \times X_2 \cdots X_n)^*$ be a sequence of tuples, $\pi_i(\sigma)$ returns the sequence of elements in σ at position $1 \leq i \leq n$, e.g., $\pi_2(\langle (x_1^1, x_2^1, \ldots, x_n^1), (x_1^2, x_2^2, \ldots, x_n^2), \ldots, (x_1^{|\sigma|}, x_2^{|\sigma|}, \ldots, x_n^{|\sigma|}) \rangle) = \langle x_2^1, x_2^2, \ldots x_2^{|\sigma|} \rangle$.

Given $\sigma, \sigma' \in X^*$, $\delta(\sigma, \sigma') \in \mathbb{N}_{\geq 0}$ represents the *Longest Common Subsequence (LCS) edit distance* (only using *insertions* and *deletions*) between σ and σ', i.e., the minimum number of edits required to transform σ into σ'. For example, $\delta(\langle w, x, y \rangle, \langle x, y, z \rangle) = 2$. Note that $\delta(\sigma, \sigma') = \delta(\sigma', \sigma)$ (δ is symmetrical) and

Table 1: Simple example of an event log. Rows capture *events* recorded in the context of the execution of the process.

Case-id	Event-id	Activity name	Starting time	Finishing time	...
⋮	⋮	⋮	⋮	⋮	⋯
7	35	Register(a)	2020-01-02 12:23	2020-01-02 12:25	...
7	36	Analyze Defect(b)	2020-01-02 12:30	2020-01-02 12:40	...
7	37	Inform User(g)	2020-01-02 12:45	2020-01-02 12:47	...
7	38	Repair(Simple)(c)	2020-01-02 12:45	2020-01-02 13:00	...
8	39	Register(a)	2020-01-02 12:23	2020-01-02 13:15	...
7	40	Test Repair(e)	2020-01-02 13:05	2020-01-02 13:20	...
7	41	Archive Repair(h)	2020-01-02 13:21	2020-01-02 13:22	...
8	42	Analyze Defect(b)	2020-01-02 12:30	2020-01-02 13:30	...
⋮	⋮	⋮	⋮	⋮	⋱

$\delta(\sigma, \sigma'') \leq \delta(\sigma, \sigma') + \delta(\sigma', \sigma'')$ (triangle inequality applies to δ). Given a sequence $\sigma \in X^*$ and a set of sequences $S \subseteq X^*$, we define $\Delta(\sigma, S) = \min_{\sigma' \in S} \delta(\sigma, \sigma')$.

Event logs, i.e., collections of events representing the execution of several instances of a process, are the starting point of process mining algorithms. An example event log is shown in Table 1. Events record when an activity was performed (according to their *Starting* and *Finishing time*) for an instance of the process (represented by the *Case-id* column). For some applications, e.g., alignment computation, only the *control-flow information*, i.e., sequences of activities executed in the context of a process instance, is required. Hence, we adopt the aforementioned mathematical model of an event log (in practice however, event data typically records much more features related to the executed activities, e.g., resource information and costs of activities).

Definition 1 (Event Log). *Let Σ denote the* universe of activities. *A trace σ is a sequence of activities ($\sigma \in \Sigma^*$). An* event log $L \in \mathcal{B}(\Sigma^*)$ *is a bag of traces.*

Process models are used to describe the (expected) behavior of a process. Process models come in various forms, i.e., ranging from simple conceptual drawings to mathematical concepts with associated execution semantics, e.g., Petri nets [20] or BPMN diagrams [21]. For example, in Fig. 2, we show process model M_1 in BPMN notation. The model describes that the first activity in the process should be a, followed by activities b and c are in parallel. It is possible to skip activity c. After the execution of activity b, if we execute activity d, we should again execute b. Activity e finalizes the process. In this paper's context, we do not assume a specific modeling notation; rather, we assume process models to describe a collection of sequences of activities.

Definition 2 (Process Model). *Let Σ denote the universe of activities. A process model M describes the intended behavior of a process. We refer to the behavior described by model M as its* language $\emptyset \subset \mathcal{L}(M) \subseteq \Sigma^*$, *i.e., a non-empty collection of activity sequences.*

For the given process model M_1 in Fig. 2, we have $\mathcal{L}(M_1) = \{\langle a, b, e \rangle, \langle a, b, c, e \rangle, \langle a, c, b, e \rangle, \langle a, b, d, b, e \rangle, \ldots\}$. Note that, due to the existence of loops, the language of a process model may be infinite.
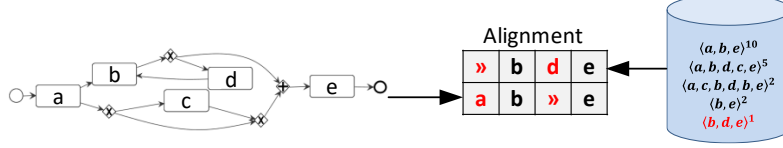
Figure 2: An example of a process model M_1 (in BPMN notation) and a simple event log L_1 (represented in multiset-view). The optimal alignment of the last trace of the event log and the process model is shown in the middle of the figure.

To quantify whether an event log conforms to a process model, we use alignments. An alignment between a trace and a model describes which events in the trace can be "aligned with activities described by the process model". Furthermore, alignments indicate whether an event cannot be explained by the model or whether an activity as described by the model was not observed. In Fig. 2, an alignment of trace $\langle b, d, e \rangle$, and the given process model is provided. Observe that the trace does not contain activity a, which should always be present according to the model. In the alignment, this is visualized by the first column $\frac{\gg}{a}$. Similarly, after the observed d-activity, no second b-activity was observed. As such, in this alignment, the occurrence of d is rendered obsolete, i.e., visualized as $\frac{d}{\gg}$. We formally define alignments as follows.

Definition 3 (Alignment). *Let Σ denote the universe of activities, let M be a process model with corresponding language $\emptyset \subset \mathcal{L}(M) \subseteq \Sigma^*$ and let $\sigma \in \Sigma^*$ be a trace. An alignment γ of σ and M, is a sequence, characterized as $\gamma \in ((\Sigma \cup \{\gg\}) \times (\Sigma \cup \{\gg\}))^*$, s.t., $\pi_1(\gamma)_{\downarrow_\Sigma} = \sigma$ and $\pi_2(\gamma)_{\downarrow_\Sigma} \in \mathcal{L}(M)$. The set of all possible alignments of trace σ and model language $\mathcal{L}(M)$ is denoted as $\Gamma(\sigma, \mathcal{L}(M))$.*

In the context of this paper, given $\gamma \in \Gamma(\sigma, \mathcal{L}(M))$, we write $\varphi(\gamma) = \pi_2(\gamma)_{\downarrow_\Sigma}$ to refer to *the "model trace" corresponding to γ*. Let $c \colon (\Sigma \cup \{\gg\}) \times (\Sigma \cup \{\gg\}) \to \mathbb{R}$, then, given $\sigma \in \Sigma^*$, $M \subseteq \Sigma^*$ and $\gamma \in \Gamma(\sigma, \mathcal{L}(M))$, we let $\kappa_c(\gamma) = \sum\limits_{1 \leq i \leq |\gamma|} c(\gamma(i))$ denote the cost of alignment γ. We let $\Gamma_c^\star(\sigma, \mathcal{L}(M)) = \underset{\gamma \in \Gamma(\sigma, \mathcal{L}(M))}{\arg\min} \kappa_c(\gamma)$ be the set of *optimal/minimal alignments*. We let $z_c(\sigma, \mathcal{L}(M)) = \underset{\gamma \in \Gamma(\sigma, \mathcal{L}(M))}{\min} \kappa_c(\gamma)$ be the optimal alignment cost for trace σ and model M (hence: $\forall \gamma \in \Gamma_c^\star(\sigma, \mathcal{L}(M)) (\kappa_c(\gamma) = z_c(\sigma, \mathcal{L}(M)))$). In the remainder, we assume that c represents the *standard cost function*, i.e., $c(a, \gg) = c(\gg, a) = 1, \forall a \in \Sigma$, $c(a, a) = 0, \forall a \in \Sigma$ and $c(a, a') = \infty, \forall a \neq a' \in \Sigma$, and we omit it as a subscript.

4   Alignment Error Bound Estimation

In this section, we show how to estimate maximal alignment approximation error bounds. We first show that *edit distance* can be used to compute conventional

optimal alignments. Subsequently, we use this result to show that we are able to quantify the maximum alignment approximation error for an arbitrarily given activity sequence. Finally, we show that we can guarantee tighter approximation bounds by exploiting a collection of arbitrary activity sequences.

4.1   Computing the Maximal Alignment Approximation Error

In this section, we show that, for given traces $\sigma, \sigma' \in \Sigma^*$ and process model M, $\Delta(\sigma, \sigma')$ quantifies a range for the optimal alignment value $z(\sigma, \mathcal{L}(M))$, when using $z(\sigma', \mathcal{L}(M))$ as an estimator for $z(\sigma, \mathcal{L}(M))$. We first show that for the standard cost function, we are able to use the LCS edit distance function to compute the cost of the optimal alignment.

Lemma 1 (Edit Distance Quantifies Alignment Costs). *Let Σ denote the universe of activities, let $\sigma \in \Sigma^*$ be a trace, let M be a process model and let $\gamma \in \Gamma^\star(\sigma, \mathcal{L}(M))$ be an optimal alignment of σ and M. Using the standard cost function, $\kappa(\gamma) = \delta(\sigma, \varphi(\gamma))$*

Proof. Observe that γ only contains elements of the form (a, a), $(a, \gg)$ and $(\gg, a)$. Let R denote the set of elements of the form $(a, \gg)$ and let I denote the set of elements of the form $(\gg, a)$. Transforming σ into $\varphi(\gamma)$ is achieved by removing activities in σ represented by the elements in R and inserting activities in σ represented by the elements in I. Hence, $\kappa(\gamma) = R + I$. Similarly, $\delta(\sigma, \varphi(\gamma))$ represents the minimum number of insertions and removals to transform σ into $\varphi(\gamma)$. Thus, if $\kappa(\gamma) < \delta(\sigma, \varphi(\gamma))$, then $\delta(\sigma, \varphi(\gamma))$ does not represent the minimal number of edits. Similarly, if $\kappa(\gamma) > \delta(\sigma, \varphi(\gamma))$ then γ is not optimal. □

Corollary 1 ($\Delta(\sigma, \mathcal{L}(M))$ equals $z(\sigma, \mathcal{L}(M))$). *Let Σ denote the universe of activities, let $\sigma \in \Sigma^*$ be a trace, let M be a process model with corresponding language $\emptyset \subset \mathcal{L}(M) \subseteq \Sigma^*$. Using the standard cost function, $z(\sigma, \mathcal{L}(M)) = \Delta(\sigma, \mathcal{L}(M))$.*

Proof. Let $\gamma \in \Gamma^\star(\sigma, \mathcal{L}(M))$.
   $z(\sigma, \mathcal{L}(M)) = \kappa(\gamma) = \delta(\sigma, \varphi(\gamma)) = \Delta(\sigma, \mathcal{L}(M))$. □

In the following, we show that it is possible to exploit an arbitrary activity sequence to derive a range on another activity sequence's alignment value.

Theorem 1 (Edit Distance Provides Approximation Bounds). *Let $\sigma, \sigma' \in \Sigma^*$ be two traces and let M be a process model with corresponding language $\emptyset \subset \mathcal{L}(M) \subseteq \Sigma^*$. The optimal alignment value $z(\sigma, \mathcal{L}(M))$, is within $\delta(\sigma, \sigma')$ of $z(\sigma', \mathcal{L}(M))$, i.e., $z(\sigma', \mathcal{L}(M)) - \delta(\sigma, \sigma') \leq z(\sigma, \mathcal{L}(M)) \leq z(\sigma', \mathcal{L}(M)) + \delta(\sigma, \sigma')$.*

Proof. Let $\gamma \in \Gamma^\star(\sigma, \mathcal{L}(M))$ and let $\gamma' \in \Gamma^\star(\sigma', \mathcal{L}(M))$. Triangle inequality of LCS edit distance yields $\delta(\sigma, \varphi(\gamma')) \leq \delta(\sigma, \sigma') + \delta(\sigma', \varphi(\gamma'))$, which we can rewrite (Lemma 1) to $\delta(\sigma, \varphi(\gamma')) \leq \delta(\sigma, \sigma') + z(\sigma', \mathcal{L}(M))$. Since $z(\sigma, \mathcal{L}(M)) \leq \delta(\sigma, \varphi(\gamma'))$, we have: $z(\sigma, \mathcal{L}(M)) \leq z(\sigma', \mathcal{L}(M)) + \delta(\sigma, \sigma')$.
   Similarly, $\delta(\sigma', \varphi(\gamma)) \leq \delta(\sigma, \sigma') + \delta(\sigma, \varphi(\gamma))$.   Again,   we   deduce $\delta(\sigma', \varphi(\gamma)) \leq \delta(\sigma, \sigma') + z(\sigma, \mathcal{L}(M))$. Since $z(\sigma', \mathcal{L}(M)) \leq \delta(\sigma', \varphi(\gamma))$, we deduce $z(\sigma', \mathcal{L}(M)) - \delta(\sigma, \sigma') \leq z(\sigma, \mathcal{L}(M))$. Hence, we obtain $z(\sigma', \mathcal{L}(M)) - \delta(\sigma, \sigma') \leq z(\sigma, \mathcal{L}(M)) \leq z(\sigma', \mathcal{L}(M)) + \delta(\sigma, \sigma')$. □

For example, reconsider process model M_1 and event log L_1 in Fig. 2. Observe that $z(\langle a,c,c,b,d,e\rangle, \mathcal{L}(M_1))=2$ and $\delta(\langle a,c,c,b,d,e\rangle, \langle a,c,b,d,e\rangle)=1$. Hence, we deduce $1{\le}z(\langle a,c,b,d,e\rangle, \mathcal{L}(M_1)){\le}3$. If $z(\langle a,c,c,b,d,e\rangle, \mathcal{L}(M_1))$ is unknown, $\delta(\langle a,c,c,b,d,e\rangle, \langle a,c,b,d,e\rangle)=1$ implies that using it as an estimator for $z(\langle a,c,b,d,e\rangle, \mathcal{L}(M_1))$ *yields a maximal absolute approximation error of* 1.

4.2   Generating Proxy-Sets

The result of Theorem 1 implies that, given a process model M and traces $\sigma,\sigma'{\in}\Sigma^*$, when using $z(\sigma', \mathcal{L}(M))$ as an estimator for $z(\sigma, \mathcal{L}(M))$, we obtain an approximation error $\epsilon{\le}\delta(\sigma,\sigma')$. Interestingly, the bound on ϵ is determined independently of the model. Furthermore, σ' is allowed to be an arbitrary sequence, i.e., it is perfectly fine if $\sigma'{\notin}\mathcal{L}(M)$, and, given some $L{\in}\mathcal{B}(\Sigma^*)$ s.t. $\sigma{\in}\overline{L}$, $\sigma'{\notin}\overline{L}$. Hence, given an arbitrary set of sequences $\Omega{\subseteq}\Sigma^*$, $\arg\min_{\sigma'\in\Omega}\delta(\sigma,\sigma')$ represents the members of Ω that minimize the expected maximum error when using $z(\sigma', \mathcal{L}(M))$ as an estimator (i.e., for $\sigma'{\in}\arg\min_{\sigma'\in\Omega}\delta(\sigma,\sigma')$). In the remainder, we refer to such a set of sequences $\Omega{\subseteq}\Sigma^*$ as a *proxy-set*, i.e., we intend to "align by proxy through Ω".

Observe that, for an event log $L{\in}\mathcal{B}(\Sigma^*)$ and proxy-set $\Omega{\subseteq}\Sigma^*$, $\forall\sigma{\in}\overline{L}\left(\min_{\sigma'\in\Omega}\delta(\sigma,\sigma')=0\right){\Leftrightarrow}\Omega{\supseteq}\overline{L}$, i.e., if every member of the log has an edit distance of 0 w.r.t. the proxy-set, then every member of the event log is a member of the proxy-set (and vice versa). Clearly, in such a case, using proxy-set Ω yields optimal alignments, yet, at the same (or even worse) time and memory complexity as computing conventional optimal alignments.

In the remainder, given an event log $L{\in}\mathcal{B}(\Sigma^*)$ and proxy-set $\Omega{\subseteq}\Sigma^*$, we let $\epsilon_\Omega(L){=}\sum_{\sigma\in\overline{L}}L(\sigma){\cdot}\min_{\sigma'\in\Omega}\delta(\sigma,\sigma')$. Given two proxy-sets $\Omega,\Omega'{\subseteq}\Sigma^*$, Ω *dominates* Ω' for event log L if and only if $\epsilon_\Omega(L){\le}\epsilon_{\Omega'}(L)$ and $|\Omega|{<}|\Omega|'$. In such a case, we refer to Ω' as a *redundant* proxy-set. A proxy-set Ω is *k-optimal* for event log L if and only if $\forall\Omega'{\in}\Sigma^*\,(|\Omega'|{=}k\Longrightarrow\epsilon_\Omega(L){\le}\epsilon_{\Omega'}(L))$. A *k-optimal* proxy-set Ω is *k-primal* if $|\Omega|{=}k$. For example, $\Omega{=}\overline{L}$ is $|\overline{L}|$-primal, 1-optimal, 2-optimal, ..., $|\overline{L}|$-optimal. Furthermore, it is easy to see that any (k-primal) proxy-set Ω with $|\Omega|{>}L$ is dominated by L and hence redundant. More interestingly, primal proxy-sets that are smaller than the event log are never redundant.

Theorem 2 (Primal Proxy-Sets are Non-Redundant). *Let* $L{\in}\mathcal{B}(\Sigma^*)$ *be an event log and let* $\Omega{\subseteq}\Sigma^*$ *be a proxy-set such that* $|\Omega|{<}|\overline{L}|$ *and* Ω *is* k-primal. Ω *is* non-redundant.

Proof. Assume that Ω *is redundant. Hence,* $\exists\Omega'{\subseteq}\Sigma^*\,(|\Omega'|{<}|\Omega|\wedge\epsilon_{\Omega'}(L){\le}\epsilon_\Omega(L))$. *However, observe that, we are able to create* $\Omega''{=}\Omega'{\cup}L''$ *with* $|L''|{=}|\Omega|{-}|\Omega'|$ *and* $\sigma{\in}L''\Longrightarrow\sigma{\in}\overline{L}\wedge\sigma{\notin}\Omega'$ *(note that* $|\Omega|{=}|\Omega''|$). *Observe that* $\epsilon_{\Omega''}(L){<}\epsilon_{\Omega'}(L)$ *and as a consequence* $\epsilon_{\Omega''}(L){<}\epsilon_\Omega(L)$, *contradicting the fact that* Ω *is k-primal.* $\square$

Observe that Theorem 2 implies that for any event log $L{\in}\mathcal{B}(\Sigma^*)$ and $k{\in}1,2,\ldots|L|$, there exists a k-primal proxy-set Ω. A k-primal proxy-set minimizes the maximal possible error bound, and hence, can be regarded as the

optimal proxy-set to use of size k. However, providing such proxy-sets is usually NP-Hard. In the upcoming paragraphs, we briefly introduce various proxy-set generation methods and their relation to primal proxy-sets.

Sampling Proxy-sets can be generated using sampling methods: either sampling members for the input event log, the given process model, or a mixture thereof. In previous work, we investigated sampling of model behavior using uniform distributions [6] and event-log-guided process model simulation [19].

Strictly sampling the behavior from the process model, i.e., $\Omega \subseteq \mathcal{L}(M)$, particularly when using event-log-guided simulation yields (under standard cost function) $z(\sigma', \mathcal{L}(M))=0$, $\forall \sigma' \in \Omega$. On the one hand, it is very unlikely that such a proxy-set is k-primal. On the other hand, the fact that $z(\sigma', \mathcal{L}(M))=0$, $\forall \sigma' \in \Omega$, can be exploited. For example, given some trace $\sigma \in \overline{L}$ and some $\sigma' \in \arg\min_{\sigma' \in \Omega} \delta(\sigma, \sigma')$, rather than using $z(\sigma', \mathcal{L}(M))$ (i.e., value 0) as an estimator for $z(\sigma, \mathcal{L}(M))$, one can use $\frac{\delta(\sigma, \sigma')}{2}$. Hence, the maximal approximation error is reduced by half.

Sampling Ω from the event log is likely to result in a proxy-set that is closer to a k-primal solution, i.e., in particular when prioritizing sampling of $\sigma \in \overline{L}$ with high $L(\sigma)$ values. Hence, using event log-based sampling typically yields smaller values for the maximal obtainable approximation error. However, since the actual $z(\sigma', \mathcal{L}(M))$ for $\sigma' \in \Omega$ is unknown, we cannot tighten the estimator.

Centroid-Based Clustering For a given target size k, the best possible obtainable proxy-set is k-*primal*. As an alternative approach to sampling, *clustering algorithms* [22] are a suitable proxy-set selection mechanism. A clustering algorithm groups a set of objects into subgroups (clusters) such that the members of a cluster are similar/close, given some similarity/distance metric. In the case of proxy-set generation, the edit distance serves as a distance metric. *Centroid-based clustering algorithms*, i.e., algorithms that define clusters using a *central object* (the centroid), are of particular interest. In centroid-based clustering, the algorithms assign the objects to the centroid objects that are at a minimal distance of the object. As an example, the *K-Medoids* algorithm [23] uses objects from the object set as centroids and minimzizes the pair-wise dissimilarity of the objects and the centroids. Clearly, several variations of centroid-based clustering algorithms can be used. Whereas the clustering algorithms can be applied on an arbitrary set of activity sequences, applying them on the input event log yields proxy-sets that are close to the k-*primal* solution.

4.3 Deriving Exact Alignment Approximation Bounds

Thus far, given sequence $\sigma \in \Sigma^*$ and process model M, we have shown that a proxy-set Ω and proxy-sequence $\sigma' \in \Omega$ quantify the *maximum approximation error*, when using $z(\sigma', \mathcal{L}(M))$ as an estimator for $z(\sigma, \mathcal{L}(M))$. In this section, we show that we can exploit proxy-sets to derive *exact approximation bounds*.

When approximating alignments using proxy-set Ω, we first compute the alignments of the proxy-set traces (i.e., $\sigma' {\in} \Omega$). We derive the upper and lower bound of the alignment cost of $z(\sigma, \mathcal{L}(M))$ by simply adding/subtracting $\delta(\sigma, \sigma')$ to $z(\sigma', \mathcal{L}(M))$. Observe that, when using the standard cost function, the lower bound of any alignment is bounded, i.e., it cannot be lower than 0. Furthermore, in certain cases, it is possible to derive a tighter lower-bound. Let $\Sigma_M {=} \{a {\in} \Sigma | \exists \sigma {\in} \mathcal{L}(M)(a {\in} \sigma)\}$, then, for any $\sigma {\in} \Sigma^*$, it is easy to see that $z(\sigma, \mathcal{L}(M)) {\geq} |\sigma_{\downarrow_{\Sigma \setminus \Sigma_M}}|$, i.e., the elements of $\sigma_{\downarrow_{\Sigma \setminus \Sigma_M}}$ are always moves of the form $\frac{a}{\gg}$. Furthermore, in case $|\sigma| {<} \min\limits_{\sigma' {\in} \mathcal{L}(M)} |\sigma'|$, we need at least $|\sigma'| {-} |\sigma|$ (where $\sigma' {\in} \arg\min\limits_{\sigma' {\in} \mathcal{L}(M)} |\sigma'|$) moves of the form $\frac{\gg}{a}$. Hence, the theoretical lower-bound of any $\sigma {\in} \Sigma^*$ is equal to $\max(0, \min\limits_{\sigma' {\in} \mathcal{L}(M)} (|\sigma'|) {-} |\sigma|) {+} |\sigma_{\downarrow_{\Sigma \setminus \Sigma_M}}|$. We correspondingly define the Ω-driven lower and upper bound as follows.

Definition 4 (Ω-Driven Alignment Bounds). *Let Σ denote the universe of activities, let M be a process model with corresponding language $\emptyset {\subset} \mathcal{L}(M) {\subseteq} \Sigma^*$ and let $\Omega {\subseteq} \Sigma^*$ be a proxy-set. We let $\top_{\Omega,M} \colon \Sigma^* {\to} \mathbb{N}$ denote the Ω-driven upper bound and we let $\bot_{\Omega,M} \colon \Sigma^* {\to} \mathbb{N}$ denote the Ω-driven lower bound where:*

$$\top_{\Omega,M}(\sigma) {=} \min_{\sigma' {\in} \Omega} (z(\sigma', \mathcal{L}(M)) {+} \delta(\sigma, \sigma')) \tag{1}$$

$$\bot_{\Omega,M}(\sigma) {=} \max(\max(0, \min_{\sigma' {\in} \mathcal{L}(M)} (|\sigma'|) {-} |\sigma|) {+} |\sigma_{\downarrow_{\Sigma \setminus \Sigma_M}}|, \max_{\sigma' {\in} \Omega} (z(\sigma', \mathcal{L}(M)) {-} \delta(\sigma, \sigma'))) \tag{2}$$

Finally, given $\top_{\Omega,M}$ and $\bot_{\Omega,M}$, we quantify the alignment approximation value of $\sigma {\in} \Sigma^*$, i.e., $\hat{z}_\Omega(\sigma, \mathcal{L}(M))$, as $\hat{z}_\Omega(\sigma, \mathcal{L}(M)) {=} \frac{\top_{\Omega,M}(\sigma) {-} \bot_{\Omega,M}(\sigma)}{2}$. Theoretically, it is possible to give different weights to lower and upper bounds. However, finding the best weight is not the scope of this paper.

5   Evaluation

In this section, we explore the accuracy and the performance of our proposed method. First, we briefly describe the implementation, after which we explain the experimental setting. Finally, we report on the experimental results.

Implementation To apply the proposed conformance approximation method, we implemented the *Conformance Approximation* plug-in in the `ProM` [24] framework[1], including various proxy-set generation methods (both sampling and centroid-based clustering, cf. Section 4.2).

---

[1] `svn.win.tue.nl/repos/prom/Packages/LogFiltering`

Table 2: Statistics regarding the real event logs that are used in the experiment.

Event Log	Activities	Traces	Variants	DF#
$BPIC$-2012 [25]	23	13087	4336	138
$BPIC$-2018-$Inspection$ [26]	15	5485	3190	67
$BPIC$-2019 [27]	42	251734	11973	498
$Hospital$-$Billing$ [28]	18	100000	1020	143
$Road$ [29]	11	150370	231	70
$Sepsis$ [30]	16	1050	846	115

5.1 Experimental Setup

We applied the proposed methods to six real event logs. Basic information, e.g., the number of distinct activities, traces, and variants, of the event logs used, is given in Table 2. For each event log, we apply conformance checking using different process models. To obtain these process models, we used the Inductive Miner [31] process discovery algorithm, with infrequent thresholds equal to 0.2, 0.4, and 0.6. Typically, these models describe a strict subset of the input event log. We used four different proxy-set generation methods, all using the event log as a primary driver, i.e., *random sampling*, *frequency-based sampling*, *K-Medoids clustering* and *K-Center clustering*. In random sampling, we randomly sample variants (without replacement) from the event log to act as a proxy. In frequency-based sampling, we select traces based on their $L(\sigma)$-values, in descending order. In K-Medoids clustering, centroids are determined by minimizing the pair-wise dissimilarity. In K-Center clustering, the maximum distance between centroids and the objects is minimized. To determine the size of proxy-sets we use a different percentage of the number variants in the event logs, i.e., 5%, %10, 20%, 30%, 50%. Moreover, we have repeated each experiment four times as some results are non-deterministic.

Using the described experimental setup, we investigate the relationship between the maximum error,i.e., $\sum_{\sigma \in L} (L(\sigma) \times \min_{\sigma' \in \Omega} \delta(\sigma, \sigma'))$ and the eventual approximation error, as well as the performance of the approach.

5.2 Results

In this section, we discuss the results of the experiments. We first investigate the relationship between the estimated maximum error and the actual approximation error. Secondly, we investigate the time-performance of the estimation. Lastly, we investigate the role of the new proposed lower bound.

Estimated Maximum Error versus Approximation Error Observe that minimizing the expected maximum error does not guarantee a minimal approximation error. For example, given some model M, $\sigma \in \Sigma^*$, $\Omega = \{\sigma_1, \sigma_2\}$ and $\Omega' = \{\sigma_1, \sigma_3\}$, assume that $\delta(\sigma, \sigma_1) = 2$, $\delta(\sigma, \sigma_2) = 3$ and $\delta(\sigma, \sigma_3) = 1$. Clearly, the maximal error based on Ω is 2, and, based on Ω', it is 1. As such, we intuitively favor Ω' over Ω. However, if $z(\sigma_1, \mathcal{L}(M)) = 7$, $z(\sigma_2, \mathcal{L}(M)) = 2$ and $z(\sigma_1, \mathcal{L}(M)) = 6$,
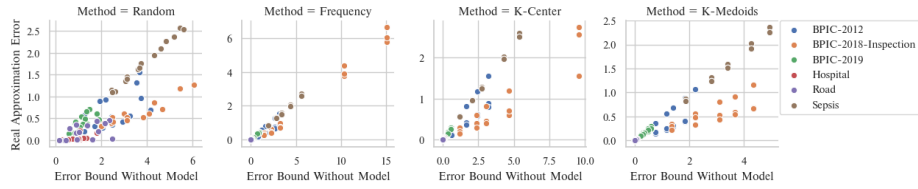
Figure 3: Scatter plots of the estimated maximum error bounds and the real approximation error using different proxy-set generation methods.

we obtain $\perp_{\Omega,M}(\sigma)=\top_{\Omega,M}(\sigma)=5$, whereas $\perp_{\Omega',M}(\sigma)=5$ and $\top_{\Omega',M}(\sigma)=7$. Hence, from Ω, we derive that $z(\sigma,\mathcal{L}(M))=5$ (note $\hat{z}_{\Omega}(\sigma,\mathcal{L}(M))=5$), whereas from Ω', we derive $5\leq z(\sigma,\mathcal{L}(M))\leq 7$ (with $\hat{z}_{\Omega'}(\sigma,\mathcal{L}(M))=6$). Thus, using Ω to derive the alignment approximation value actually yields the exact alignment value using Ω' yields an error of 1.

Given that there is no causal relation, we investigate, using the described proxy-set generation methods and event logs, the strength of the correlation between the estimated maximum error and the effective approximation error when using the proxy-set. In Fig. 3, we show the scatter plots of these two values, for each method, using different colors for the different event logs. Moreover, the corresponding Pearson correlation coefficients are presented in Table 3.

Interestingly, for frequency-based sampling, K-Center, and K-Medoids, we observe a strong correlation between the estimated maximal approximation error and the effective approximation error. For random sampling, as expected, the correlation is less strong, particularly for the Hospital-Billing and Road logs. For all event logs, the highest correlation is achieved by the K-Center method.

Table 3: Pearson correlation coefficients between the estimated bounds and the real approximation errors for different methods and event logs.

Log Name	Random	Frequency	K-Center	K-Medoids
BPIC-2012	0.577	0.701	0.820	0.806
BPIC-2018-Inspection	0.933	0.945	0.995	0.862
BPIC-2019	0.682	0.842	0.998	0.843
Hospital	0.534	0.782	0.969	0.962
Road	0.468	0.631	0.997	0.998
Sepsis	0.997	0.990	0.998	0.994

In Fig. 4, we show the effect of choosing different proxy-set generation methods and different percentages of variants in the event log on the accuracy of approximated alignment costs. As we expect, the K-Center and K-Medoids approaches provide the highest accuracy. Therefore, using these approaches, we are able to generate more suitable proxy-sets and consequently obtain more accurate approximations. Moreover, results indicate that the alignment cost error is reduced by increasing the proxy-sets' size (i.e., the selection percentage). However, for some event logs, specifically if the variants in the event log are similar, this reduction is not always significant, i.e., by just using a few trace-variants we already obtain an accurate approximation. Thus, the approximation's provided bounds help users adjust the setting more efficiently, as a user may make the choice of increasing the proxy set size, thus increasing the accuracy of approximation.
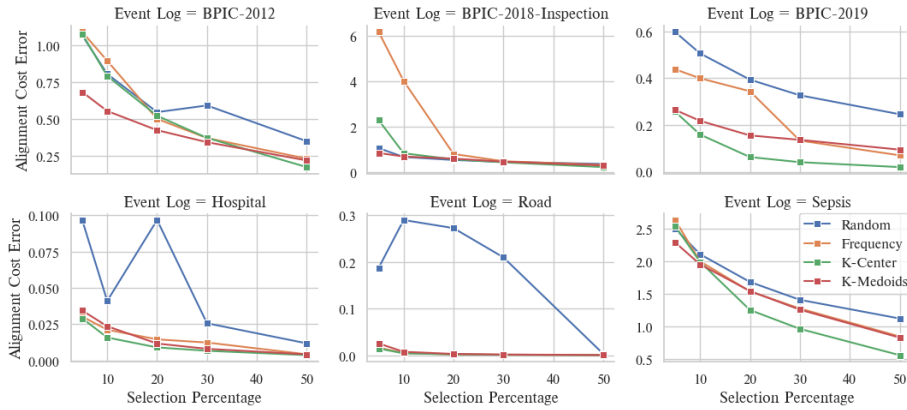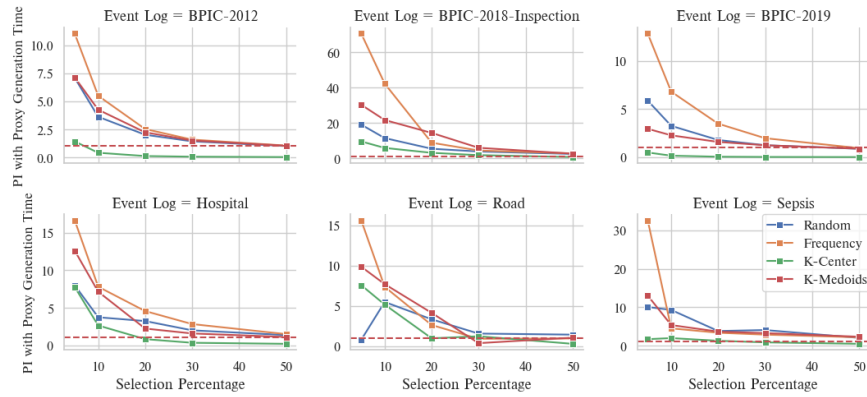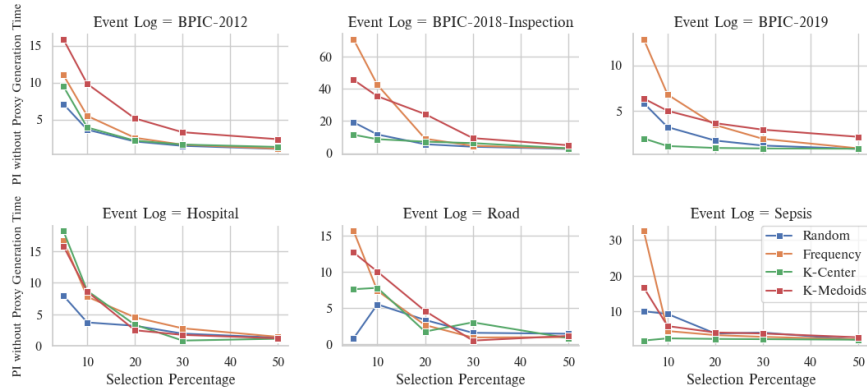
Figure 4: Effect of increasing the selected percentage of variants on approximated alignments' accuracy for different methods.

Conformance Checking Performance Improvement Here, we analyze the time performance of different proxy-set generation methods. Figures Fig. 5a and Fig. 5b show the conformance checking performance improvement using the proposed approach. To compute the *performance improvement PI*, we divide conventional alignment computation time by the alignment approximation time (both including and excluding proxy-set generation time). A higher PI-value indicates a higher performance improvement and a PI-value less than 1 indicates that there is no improvement. The greatest improvement (when we consider the proxy generation time) is achieved by using the frequency-based method as it quickly selects variants and generates the proxies. The Random method has a lower PI-value because it may select some variants that usually need more time to compute their alignments. Furthermore, the results indicate that by increasing the percentage of the size of proxies, the performance improvement is reduced. In some cases, we do not improve the performance when the proxy generation time is considered. Thus, it is crucial to avoid selecting too many traces as a proxy. Generally, the proxy generation time for K-Center and K-Medoids methods is high, especially when the z size of proxy is high. But, if we separate the proxy generation time (that is possible in some applications as explained in Section 1), we improve the conformance checking procedure's performance.

Efficiency of the Proposed Lower Bound Finally, in the last experiment, we analyze the role of the lower bound without M', i.e., $\max(0, \min_{\sigma' \in \mathcal{L}(M)} (|\sigma'| - |\sigma|) + |\sigma_{\downarrow_{\Sigma \setminus \Sigma_M}}|$ and the lower bound that uses M', i.,e., $\max_{\sigma' \in \Omega} (z(\sigma', \mathcal{L}(M)) - \delta(\sigma, \sigma')))$ in computing the final lower bound. Table 4 shows the percentage of traces that have higher values using the different bounds. In case the highest value is returned by both methods, we consider both of them as the used bound.

(a) Performance improvement with consideration of proxy-set generation time.

(b) Performance improvement without consideration of proxy selection time.

Figure 5: Effect of increasing the selected percentage of variants on performance improvement of different proxy selection methods.

The results show that in most cases, it is sufficient to use the lower bound, which is based on the proxy-set and its alignments. Consequently, using the new proposed method for bound computation, we will have tighter bounds.

Table 4: Average percentage of times that lower bounds have the highest value.

Log Name	Without Ω	With Ω
BPIC-2012	80%	100%
BPIC-2018-	66%	62%
BPIC-2019	95%	99%
Hospital	50%	100%
Road	87%	100%
Sepsis	100%	100%

6   Conclusion

In this paper, we propose to select a set of traces, compute their alignments and use these to approximate the alignments of other traces in the event log. Furthermore, we have shown that we can derive bounds for the so-introduced approximation error,

independently of the model. Additionally, we prove that based on the selected traces (i.e., a proxy-set), we can provide a bound for the approximation error that helps users estimate the approximation error and thus aids in selecting an appropriate proxy set. The experiments on the real event logs indicate, that by using the proposed instance selection methods, we are able to reduce the maximum and the average error in the alignment cost approximation. Besides, by increasing the number of the selected traces, the average possible error is reduced. But, for certain event logs, this reduction is not significant, which shows we are able to select a few traces and have an accurate approximation.

We used the approximation solutions for K-Center and K-Medoids problems in this work. However, it is useful for some applications to compute optimal solutions and find the best K variants. Moreover, it is beneficial to provide an incremental approach that keeps selecting the traces until we guarantee a tight bound for the approximation error.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer Berlin Heidelberg (2016)
2. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer (2018)
3. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B., van der Aalst, W.M.P.: Alignment based Precision Checking. In: International Conference on Business Process Management, Springer (2012) 137–149
4. Giacalone, M., Cusatelli, C., Santarcangelo, V.: Big data compliance for innovative clinical models. Big Data Res. **12** (2018) 35–40
5. Buijs, J.C., van Dongen, B., van der Aalst, W.M.P.: On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In: OTM, " On the Move to Meaningful Internet Systems", Springer (2012) 305–322
6. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Conformance checking approximation using subset selection and edit distance. In: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. Volume 12127., Springer (2020) 234–251
7. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: On-line conformance checking: relating event streams to process models using prefix-alignments. International Journal of Data Science and Analytics (2017) 1–16
8. Elhagaly, M., Drvoderić, K., Kippers, R.G., Bukhsh, F.A.: Evolution of compliance checking in process mining discipline. In: 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), IEEE (2019) 1–6
9. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **2**(2) (2012) 182–192
10. van der Aalst, W.M.P.: Decomposing petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507
11. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. Information Systems **46** (2014) 102–122

12. Verbeek, H.M.W., van der Aalst, W.M.P., Munoz-Gama, J.: Divide and conquer: A tool framework for supporting decomposed discovery in process mining. Comput. J. **60**(11) (2017) 1649–1674
13. Taymouri, F., Carmona, J.: A recursive paradigm for aligning observed behavior of large structured process models. In: International Conference on Business Process Management, Springer (2016) 197–214
14. Bauer, M., Senderovich, A., Gal, A., Grunske, L., Weidlich, M.: How much event data is enough? A statistical framework for process discovery. In: International Conference on Advanced Information Systems Engineering, Springer (2018) 239–256
15. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Improving the performance of process discovery algorithms by instance selection. Comput. Sci. Inf. Syst. **17**(3) (2020) 927–958
16. Nolle, T., Seeliger, A., Thoma, N., Mühlhäuser, M.: Deepalign: Alignment-based process anomaly correction using recurrent neural networks. In: International Conference on Advanced Information Systems Engineering, Springer (2020) 319–333
17. Bauer, M., van der Aa, H., Weidlich, M.: Estimating process conformance by trace sampling and result approximation. (2019) 179–197
18. Padró, L., Carmona, J.: Approximate computation of alignments of business processes through relaxation labelling. In: International Conference on Business Process Management, Springer (2019) 250–267
19. Fani Sani, M., Garza Gonzalez, J.J., van Zelst, S.J., van der Aalst, W.M.P.: Conformance checking approximation using simulation. In van Dongen, B.F., Montali, M., Wynn, M.T., eds.: 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020, IEEE (2020) 105–112
20. Petri, C.A., Reisig, W.: Petri net. Scholarpedia **3**(4) (2008) 6477
21. Chinosi, M., Trombetta, A.: BPMN: an introduction to the standard. Comput. Stand. Interfaces **34**(1) (2012) 124–134
22. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals of Data Science **2**(2) (2015) 165–193
23. Park, H., Jun, C.: A simple and fast algorithm for k-medoids clustering. Expert Syst. Appl. **36**(2) (2009) 3336–3341
24. van der Aalst, W.M.P., van Dongen, B., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: Prom: The process mining toolkit. BPM (Demos) **489**(31) (2009)
25. Van Dongen, B.F. (Boudewijn): Bpi challenge 2012 (2012)
26. Van Dongen, B.F. (Boudewijn), Borchert, F. (Florian): BPI challenge 2018 (2018)
27. Van Dongen, B.F. (Boudewijn): BPI challenge 2019 (2019)
28. Mannhardt, F.: Hospital billing-event log. Eindhoven University of Technology. Dataset (2017) 326–347
29. De Leoni, M., Mannhardt, F.: Road traffic fine management process. Eindhoven University of Technology. Dataset (2015)
30. Mannhardt, F.: Sepsis cases-event log. Eindhoven university of technology (2016)
31. Leemans, S.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In: BPI. (2014) 66–78