Conformance Checking Approximation Using Simulation

Mohammadreza Fani Sani^{1}, Juan J. Garza Gonzalez^{1}, Sebastiaan J. van Zelst^{2,1}, and Wil M.P. van der Aalst^{1,2}

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

Abstract—**Conformance checking techniques are used to compute to what degree a process model and real execution data correspond to each other. In recent years, alignments have proven to be useful for calculating conformance statistics. Most alignment techniques provide an exact conformance value. However, in many applications, it suffices to have an approximated alignment value. Specifically, for large event data and using standard hardware, current alignment techniques are time-consuming and sometimes intractable. This paper proposes to use simulated behaviors of process models to approximate the conformance checking value. To simulate a process model, we exploit the behavior in the given event data. This method is independent from the process model notation and provides upper and lower bounds for the approximated alignment value. We assess the quality of our approximations and compare it to existing approximation techniques. The experiments on real event data show that using the proposed method, it is possible to achieve significant performance improvements.**

Index Terms—**Process Mining, Conformance Checking Approximation, Alignment, Simulation, Edit Distance**

I. Introduction

Conformance checking aims to investigate the conformity of a discovered/designed process model w.r.t., real process executions [1]. Conformance checking techniques are used to detect deviations and to measure how accurate a process model is. It is possible to apply this branch of techniques to assess both event data and the process model. Alignments [2] were developed with the concrete goal to describe and quantify deviations in a non-ambiguous manner. Computing alignments has rapidly turned into the de facto standard conformance checking technique [3]. Moreover, alignments serve as a basis for other process mining methods that link event data to process models, e.g., they support performance analysis, decision mining [4], business process model repair [5], and prediction techniques. However, alignment computations may be time-consuming for real large event data.

In some scenarios, the diagnostic information that is produced by alignments is not required and we simply need an objective measure of model quality to compare process models, i.e., the alignment value. Moreover, in many applications, it is required to compute alignment values several times. For example, if we aim to discover an appropriate process model from event data, it is required to discover several process models using various process discovery algorithms with different settings, and, measure how each process model fits, w.r.t., the event data, i.e., by applying alignment techniques. As normal alignment methods take a considerable time for large real event data, analyzing many candidate process models is impractical. Therefore, by decreasing the alignment computation time, it is possible to consider more candidate process models in a limited time. Thus, by having an approximated conformance value, we are able to find a suitable process model faster. By providing bounds, we guarantee that the accurate alignment value does not exceed a range of values, and, consequently we determine if it is required to do further analysis or not, which saves a considerable amount of time. Thus, it is valuable to have a quick approximated conformance value and it is excellent worth to let users adjust the level of approximation.

It is shown in [6] that using the edit distance function, a subset of model behaviors can be used instead of the process model for conformance approximation. It suggests to sample some behaviors from the event log and approximate their alignments. In this paper, we extend the previous work by proposing to use process model simulation (i.e., some of its possible executable behaviors) to create a subset of process model behaviors. The core idea of this paper is to have simulated behaviors close to the recorded behaviors in the event log. Moreover, we provide bounds for the actual conformance value. Using the proposed method, users are able to adjust the amount of process model behaviors considered in the approximation, which affects the computation time and the accuracy of alignment values and their bounds. As the proposed method just uses the simulated behaviors for conformance approximation, it is independent of any process model notation. Because we use the edit distance function and do not compute any alignment, even if there is no reference process model and just some of the correct behaviors of the process (e.g., some of the valid variants) are known, the proposed method is able to approximate the conformance value. The method additionally returns problematic activities, based on their deviation rates.

We implemented the proposed method using both the ProM [7] and RapidProM [8] platforms. Moreover, we applied it to several large real event data and process models. We also compared our approach with the state-of-the-art alignment approximation method [6]. The results show that the proposed simulation method improves the performance of the conformance checking process while providing approximations close to the actual values.

The remainder of this paper is structured as follows. In Section II, we discuss related work. Section III defines preliminary notation. We explain the proposed method in Section IV and evaluate it in Section V. Section VI concludes the paper and presents some future work.

II. Related Work

In this section, we explain some conformance checking and simulation techniques in the process mining domain. For a complete overview of conformance checking techniques in process mining, we refer to [9] and [10]. Early work in conformance checking uses token-based replay [11]. This technique replays a trace of executed events from the event log on a Petri net and adds missing tokens if transitions are not able to fire. After the replay phase, the conformance statistic is computed based on remaining and missing tokens. Alignments were introduced in [12] and have rapidly developed into the standard conformance checking technique [3]. In [13] and [14], decomposition techniques are proposed for alignment computation. Applying decomposition techniques generally improves computation time. These techniques use the divide-and-conquer paradigm. However, these techniques are primarily beneficial when there are lots of unique activities in the process [15]. The authors in [3] and [16] also propose to incrementally compute prefix-alignments, and providing real-time conformance checking for event data streams. Recently, a general stochastic conformance checking method is proposed in [17] which requires a stochastic process model that is not available in many cases.

Some research has been done to approximate the alignment value. [18] uses deep learning to approximate alignment statistics. Moreover, a recursive general approach for approximating the alignment, i.e., computation of near-optimal alignments, has been proposed in [19]. Moreover, [20] uses a statistical trace sampling approach that approximates the conformance value without proving bounds for the actual alignment value. The authors of [21] suggest a conformance approximation method that applies relaxation labeling methods to a partial order representation of a process model. Similar to the previous method, it does not provide any guarantee for the approximated value. Furthermore, it needs to preprocess the process model each time. Finally, [6] recommends using a subset of model behaviors for conformance approximation and bounds for the actual value. It suggests applying the alignment technique for some traces to build the subset of model behaviors. In this paper, we propose a guided simulation method to build the subset of model behavior to generate behaviors that are closer to recorded behaviors in the event data. Unlike many conformance checking methods, this method is independent of process model notation and considers a process model as a set of possible behaviors.

Different approaches to simulation in process mining have been proposed. These approaches are mostly at an instance level, i.e., a detailed level such as the framework presented in [22], which uses the process detailed information to create a CPN model for simulation. Moreover, [23] recommends an approach to simulate an event log based on a given (stochastic) process model. In [24], a simulation approach to simulate business process models that uses information in event logs is proposed. Another direction of the simulation in process mining is an aggregated level simulation, e.g., [25], in which the system dynamics modeling technique is introduced for "what-if" analysis in processes.

III. Preliminaries

In this section, we briefly introduce basic process mining and, specifically, conformance checking terminology and notations that ease the readability of this paper.

Given a set X, a multiset B over X is a function $B\colon X{\rightarrow}\mathbb{N}_{\geq 0}$ that allows certain elements of X to appear multiple times. We write a multiset as $B{=}[e_1^{k_1}, e_2^{k_2}, ..., e_n^{k_n}]$, where for $1{\leq}i{\leq}n$, we have $B(e_i){=}k_i$ with $k_i{\in}\mathbb{N}_{\geq 0}$. If $k_i{=}1$, we omit its superscript, and if for some $e{\in}X$ we have $B(e){=}0$, we omit it from the multiset notation. $\overline{B}{=}\{e{\in}X|B(e){>}0\}$ is the set of unique elements present in the multiset. The set of all multisets over a set X is written as $\mathcal{B}(X)$.

A sequence of length n over members of set X is a function $\sigma{:}\{1, 2, ..., n\}{\rightarrow}X$ which defines the occurrence order of elements of set X. We show a sequence using the notation $\sigma{=}\langle s_1, ..., s_n\rangle$ where $s_i{=}\sigma(i)$, for $1{\leq}i{\leq}n$. We denote with $s_i{\in}\sigma$ that s_i is an element of the sequence σ. Moreover, the set of all possible sequences over set X is shown by X^*. The empty sequence is denoted with ϵ. Furthermore, $|\sigma|$ indicates the length of sequence σ, e.g., $|\langle a, d, d, e\rangle|{=}4$. Function $hd\colon X^*{\times}\mathbb{N}_{\geq 0}{\rightarrow}X^*$, returns the "head" of a sequence, i.e., given a sequence $\sigma{\in}X^*$, $hd(\sigma, k){=}\langle s_1, s_2, .., s_k\rangle$, i.e., the sequence of the first k elements of σ. If $k{\geq}|\sigma|$, then $hd(\sigma, k){=}\sigma$. Symmetrically, $tl\colon X^*{\times}\mathbb{N}_{\geq 0}{\rightarrow}X^*$ returns the "tail" of a sequence and is defined as $tl(\sigma, k){=}\langle s_{n-k+1}, s_{n-k+2}, ..., s_n\rangle$, i.e., the sequence of the last k elements of σ. If $k{\geq}|\sigma|$, $tl(\sigma, k){=}\sigma$. We define that $hd(\sigma, 0){=}tl(\sigma, 0){=}\epsilon$. The concatenation of two sequences σ_1 and σ_2 with length m and n is sequence $\sigma_3{=}\sigma_1{\cdot}\sigma_2$ with length $m{+}n$ where $\sigma_3(i){=}\sigma_1(i)$ for $1{\leq}i{\leq}m$ and $\sigma_3(i){=}\sigma_2(i{-}m)$ for $m{+}1{\leq}i{\leq}m{+}n$. A subsequence σ_s of σ is obtained by removing $|\sigma|{-}|\sigma_s|$ elements of σ where the result equals σ_s. For example, $\langle b, d\rangle$ is a subsequence of $\langle a, b, c, d, e\rangle$. Moreover, σ' is a strict subsequence of sequence σ if there exist σ_1 and σ_2 such that $\sigma{=}\sigma_1{\cdot}\sigma'{\cdot}\sigma_2$ and we denote it by $\sigma'{\sqsubseteq}\sigma$; furthermore, if $\sigma_1{=}\epsilon$, we say that σ' is a prefix of σ.

Having two sequences, function $\omega{:}X^*{\times}X^*{\rightarrow}X^*$ returns one of the longest common subsequence of them. For example, $\omega(\langle a, b, d, e\rangle, \langle c, a, d, f, e\rangle){=}\langle a, d, e\rangle$. Finally, $fq{:}X^*{\times}X^*{\rightarrow}\mathbb{N}_{\geq 0}$ is a function that returns how many times in a sequence a subsequence is present. For example, $fq(\langle d, a, d, e\rangle, \langle d\rangle){=}2$.

Event logs are the starting point of many process mining algorithms. For alignment computation, we just use the control-flow information of the event logs. Therefore, we define an event log as follows.

*Definition 1 (**Event Log**):* Let $\mathcal{A}$ be the universe of activities and $A{\subseteq}\mathcal{A}$ is a set of activities. An event log is a multiset of

sequences over A, i.e., $L{\in}\mathcal{B}(A^*)$. Moreover, we refer to each $\sigma{\in}\overline{L}$ as a "*variant*" whereas $L(\sigma)$ denotes how many traces of the form σ are presented within the event log. Moreover, $|\overline{L}|$ refers to the number of variants in the event log.
For example, in the event log that is presented in Figure 1, $A{=}\{a, b, c, d, e\}$, $|\overline{L}|{=}5$, and $L(\langle a, b, e\rangle){=}2$.

There are many notations to describe the possible behaviors described by a process model. Here, we define a process model using a set of all its possible replayable behaviors as follows.

*Definition 2 (**Process Model**):* Let $\mathcal{A}$ be the universe of activities and $A{\subseteq}\mathcal{A}$ is a set of activities. A process model is a set of sequences over A, i.e., $M{\subseteq}A^*$.
For example, in the process model that is presented in Figure 1 in a Petri net notation, we have $\langle a, b, d, b, c, e\rangle{\in}M$, $\langle a, b, e\rangle{\in}M$, and $\langle a, b, d, b, e\rangle{\in}M$. Note that in case of having an unbounded loop in a process model, Set M is infinite.

As we define event logs and process models as collections of sequences of activities, we are able to use edit distance and the optimal alignment functions to compute their alignment.

*Definition 3 (**Edit Distance and Optimal Alignment Cost**):* Let A be a set of activities and $\sigma_l, \sigma_m{\in}A^*$ be two traces. $\triangle{:}A^*{\times}A^*{\to}\mathbb{N}_{\geq 0}$ is a function such that $\triangle(\sigma_l, \sigma_m){=}|\sigma_l|{+}|\sigma_m|{-}2{\times}|\omega(\sigma_l, \sigma_m)|$ returns the minimal number of required edits to transform σ_l to σ_m. In addition, we define $\delta{:}A^*{\times}A^*{\to}B(A){\times}B(A)$ that returns two multisets of synchronous and asynchronous moves. Synchronous moves correspond to one of the longest common subsequences of the given sequences and asynchronous moves represents the edited activities that are required to transform subsequences. Moreover, $\Gamma{:}A^*{\times}\mathbb{P}(A^*){\to}\mathbb{N}_{\geq 0}$ is a function that returns the minimum number of edits to transfer a trace to one of the traces in a set of traces, i.e., the optimal alignment cost.
For example, $\triangle(\langle a, b, d, b, e\rangle, \langle a, c, d, e\rangle){=}3$ that corresponds two deletions (i.e., *b*) and one insertion (i.e., *c*). Therefore, $\delta(\langle a, b, d, b, e\rangle, \langle a, c, d, e\rangle){=}([a, d, e], [b^2, c])$. By using $\upharpoonright_1$ and $\upharpoonright_2$, we retrieve the synchronous and asynchronous moves respectively. For example, $([a, d, e], [b^2, c^1]){\upharpoonright_2}{=}[b^2, c]$ are the asynchronous moves. Note that, there may exist more than one longest common subsequence for two sequences and the δ is non-deterministic. Moreover, for the model that is presented in Figure 1, $\Gamma(\langle a, c, e\rangle, M){=}1$ that corresponds to transfer $\langle a, c, e\rangle$ to $\langle a, b, c, e\rangle{\in}M$ or $\langle a, c, b, e\rangle{\in}M$. In [6], it is shown that $\Gamma(\sigma, M)$ equal to the optimal alignment cost.

To compute the fitness value of a trace and a model we use the following equation.

$$fitness_{Trace}(\sigma_L, M) = \frac{\Gamma(\sigma_L, M)}{|\sigma_L| + M_s} \tag{1}$$

In the above equation, $M_s{=}\min\limits_{\sigma\in M}(|\sigma|)$ is the length of the shortest trace in the model. To compute the fitness value of an event log and a process model, we use the following formula.

$$Fitness(L, M) = \frac{\sum\limits_{\sigma\in\overline{L}} L(\sigma){\times}fitness_{Trace}(\sigma, M)}{\sum\limits_{\sigma\in\overline{L}} L(\sigma)} \tag{2}$$

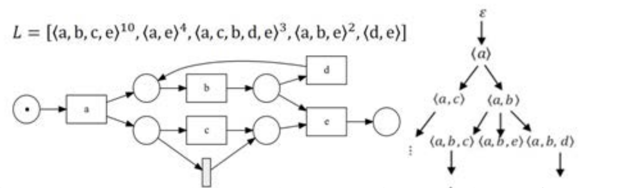
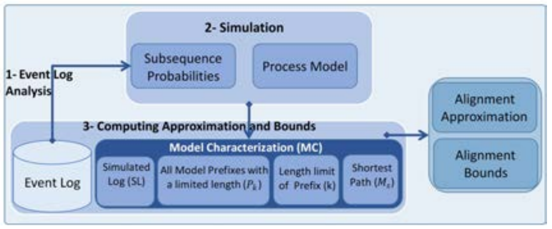


Figure 1: An example event log, a Petri net, and a prefix tree.



2- Simulation
Subsequence Probabilities
Process Model
1- Event Log Analysis
3- Computing Approximation and Bounds
Model Characterization (MC)
Event Log
Simulated Log (SL)
All Model Prefixes with a limited length (P_k)
Length limit of Prefix (k)
Shortest Path (M_s)
Alignment Approximation
Alignment Bounds

Figure 2: A schematic view of the proposed method.

IV. Approximating Alignments using Subset of Model Behavior

In this section, we present the proposed conformance approximation method. We aim to approximate the alignment value without any alignment computation. A schematic view of this method is shown in Figure 2. We first analyze the event log; then, based on the probabilities of subsequences, we provide characteristics of the process model using the simulation method. Afterwards, based on these characteristics, we provide prefixes for the alignment approximation. In the following, each stage is explained in more details.

A. Event log analysis

In this stage, we traverse the event log to find all the activities and variants in the event log. Moreover, we compute the occurrence probability of different sequences with a specific length. These probabilities guide the simulation algorithm to generate model traces close to the available traces in the event log. The occurrence probability of a subsequence $\sigma'{\in}A^*$ is computed as follows.

$$Prob(\sigma', L){=}\frac{\sum\limits_{\sigma\in\overline{L}} L(\sigma) \times fq(\sigma, \sigma')}{\sum\limits_{\sigma\in\overline{L}} (\sum\limits_{\sigma"\sqsubseteq\sigma\wedge\ |\sigma"|=|\sigma'|} (L(\sigma) \times fq(\sigma, \sigma"))} \tag{3}$$

For example, for the event log that is presented in Figure 1, $Prob(\langle a, b, c\rangle, L){=}\frac{10}{10+10+3+3+3+2}$ or $Prob(\langle a, b, c, e\rangle, L){=}\frac{10}{10+3+3}$. By getting the maximum length of subsequence from the user, we are able to compute the probabilities of them.

B. Simulation

The input of the simulation algorithm is a process model and an event log and it provides a *Model Characterization* that is defined as follows.

*Definition 4 (**Model Characterization**):* Let $M{\subseteq}A^*$ be a process model, a 4-tuple $MC{=}(SL, P_k, k, M_s)$ be a model characterization of M, where $SL{\subseteq}M$ is a subset of model traces and $M_s{=}\min\limits_{\sigma\in M}(|\sigma|)$ is the length of the shortest trace in model M. Moreover, $P_k{=}\{hd(\sigma, i)|\sigma{\in}M \wedge i \leq k\}$ is the set of all prefixes in M with length less or equal to k.

To simulate the process model, we use a prefix tree. In Figure 1 a process model and a part of its prefix tree are shown. We consider sequence σ' as a prefix of model M if $\exists_{\sigma\in M\wedge l\in\mathbb{N}_{\geq 0}}(hd(\sigma,l)=\sigma')$. We use P_k to show the information of the prefix tree that contains all the observed prefixes.

Most process model notations, e.g., BPMN and Petri net, allow us to clarify which activities are executable after the execution of sequence of activities (i.e., a prefix). Thus, we define prefix extension as follows.

*Definition 5 (**Prefix extension**):* Let A be a set of activities and let $\sigma'\in A^*$ be a sequence of activities and let $M\subseteq A^*$ be a process model. The prefix extension function $\alpha{:}A^*\times\mathbb{P}(A^*)\nrightarrow\mathbb{P}(A^*)$ returns the set of all possible extensions of prefix σ'. In other words, $\alpha(\sigma',M)=\{\sigma'\cdot\langle a\rangle|\exists_{\sigma\in M}\,(\sigma'\cdot\langle a\rangle=hd(\sigma,|\sigma'|+1)\}$.
For example, if M is corresponding to the process model that is presented in Figure 1, $\alpha(\langle a,b\rangle,M)=\{\langle a,b,c\rangle,\langle a,b,d\rangle,\langle a,b,e\rangle\}$.

The simulation algorithm starts with initializing $P=\{\epsilon\}$ and $SL=\{\}$. At each step, we select one of the non-extended prefixes in P and discover all its possible extensions, i.e., $\alpha(\sigma',M)$, and append them to P. Moreover, if any of the newly extended prefixes are complete trace, we append them to SL. Therefore, the updated sets are $P{=}P\cup(\alpha(\sigma',M))$ and $SL{=}SL\cup\{\sigma\in\alpha(\sigma',M)|\sigma\in M\}$. Note that a prefix $\sigma'\in P$ is non-extended if $\nexists_{\sigma"\in P\setminus\{\sigma'\}}\,(\sigma'{=}hd(\sigma",|\sigma'|))$.

The simulation method is guided to select the prefix with the highest probability according to Equation 3. In this regard, for all non-extended prefixes in P, the method computes $Prob(tl(\sigma',l),L)$, where l indicates the maximum length of subsequences that is given by the user. For example, in Figure 1, if $P{=}\{\epsilon,\langle a\rangle,\langle a,b\rangle,\langle a,c\rangle\}$, we know that $\langle a,b\rangle$ and $\langle a,c\rangle$ are non-extended prefixes. Therefore, the method chooses $\langle a,b\rangle$ which has a higher probability in the event log. By extending this prefix, we have $P{=}\{\epsilon,\langle a\rangle,\langle a,b\rangle,\langle a,c\rangle\}\cup\{\langle a,b,c\rangle,\langle a,b,d\rangle,\langle a,b,e\rangle\}$, and $SL{=}\{\langle a,b,e\rangle\}$.

By having the k value, we guarantee that all possible prefixes of the model with length less or equal to k are present in P_k. The proposed simulation method finds k based on the length of the shortest non-extended prefix in P.

We continue the simulation procedure until at least one of the following conditions is satisfied: 1) the number of simulated traces, i.e., $|SL|$ reaches to the simulation size, i.e., given by the user 2) there exist no non-extended prefix in P which means we could not simulate any new traces 3) having $k{\geq}2\times\max\limits_{\sigma\in\overline{L}}(|\sigma|){+}M_s$ that indicates all the unseen prefixes are longer than two times of the longest trace in the event log plus the shortest path in the model. In the next subsection, we prove that if we have the third condition, we are able to find the optimal alignment cost using the simulated traces. Moreover, we use P_k notation to show a set of all observed prefixes with length less or equal to k where $P_k{=}\{\sigma'\in P\mid|\sigma'|{\leq}k\}$.

To sum up, the output of the simulation stage is $MC{=}(SL,P_k,k,M_s)$. Note that any other simulation method that generates a model characterization can be used in the proposed approximation method. For example, it is possible to extend the prefix tree in a breadth first traversal order. If the simulation method could not guarantee to have a set of complete prefixes with a specific length, we should use $P_k{=}\{\epsilon\}$ and $k{=}0$ in the corresponding MC.

C. Computing Alignment Cost Bounds and Approximation

In this stage, for each variant $\sigma\in\overline{L}$, we compute the bounds and an approximation for its alignment cost based on a model characterization $MC{=}(SL,P_k,k,M_s)$ which can be produced by any simulation method. Thus, the proposed approach is independent of the process model notation and the simulation method to provide bounds and approximation values. Here, we first explain how to compute the upper and lower bounds; then, the approximation method is described.

Upper bound: For the upper bound of the alignment cost, we compute $UB(MC,\sigma){=}\Gamma(\sigma,SL)$. It is shown in [6] that $\Gamma(\sigma,SL){\geq}\Gamma(\sigma,M)$ as Γ returns the distance of the most similar trace and $SL{\subseteq}M$. In the following, we prove that if $k{\geq}2\times|\sigma|{+}M_s$, then $\Gamma(\sigma,SL)$ returns the optimal alignment. Note that by having P_k, SL contains all the model traces with length less or equal to k.

*Lemma 1 (**Maximum required length of prefixes**):* We know that $\Gamma(\sigma,M){\leq}|\sigma|{+}M_s$. Suppose that $\sigma'_m{\in}M$ corresponds to the optimal alignment of σ, i.e., $\Gamma(\sigma,M){=}\triangle(\sigma,\sigma'_m)$. Therefore, $\triangle(\sigma,\sigma'_m){\leq}|\sigma|{+}M_s$ and according to Definition 3, $|\sigma|{+}|\sigma'_m|{-}2{\times}|\omega(\sigma,\sigma'_m)|{\leq}|\sigma|{+}M_s$. Note that the maximum length of the longest common subsequence of two sequences is at most the length of the shorter sequence. Thus, in the worst case, $|\sigma'_m|{\leq}2{\times}|\sigma|{+}M_s$. Therefore, the length of the model trace corresponds to the optimal alignment of trace σ should be less or equals to $2{\times}|\sigma|{+}M_s$. In other words, if $k{\geq}2\times\max\limits_{\sigma\in\overline{L}}(|\sigma|){+}M_s$, we return the actual fitness value using SL.

Lower bound: For the lower bound, we use the maximum value of three values, i.e., $LB(MC,\sigma){=}Max(lb_1,lb_2,lb_3)$. let $A'{\subseteq}A$ be the set of all activities in the model, $lb_1(MC,\sigma){=}|\sigma|{-}|\sigma{\upharpoonright_{A'}}|$ where $\sigma\upharpoonright_{A'}$ is a projection of sequence σ on activities in set A'. It means if we have some activities in the trace that are not present in the model, we can easily consider them as asynchronous moves. Moreover, as discussed in [6], $lb_2(MC,\sigma){=}\max(M_s{-}|\sigma\upharpoonright_{A'}|,0)$ is a lower bound for the optimal alignment cost that is useful if the length of the trace is shorter than the shortest path in the model. Note that both lb_1 and lb_2 are independent from the simulation result. Finally, we compute the third lower bound as $lb_3(MC,\sigma){=}\Gamma(hd(\sigma,k),P_k)$. As we have all possible prefixes of model M with length less or equal to k in P_k, lb_3 computes the minimum number of edits that is required to have a valid prefix for trace σ. As $\epsilon{\in}P_k$, we always have $0{\leq}lb_3(MC,\sigma){\leq}\min(k,|\sigma|)$. All lb_1, lb_2 and lb_3 are valid lower bounds for the actual alignment cost; therefore, we use the highest number to have a tighter bound.

Approximation: In both SL and $\overline{L}$, because of the presence of loops in the process, it is possible that a subsequence is

Table I: Result of using the proposed approximation method for the event log that is given in Figure 1, using model characterization $MC{=}(\{\langle a,b,e\rangle\},\{\epsilon,\langle a\rangle,\langle a,b\rangle,\langle a,c\rangle\},2,3\}$.

Trace/ Event Log	$\Gamma(\sigma,M)$	$\Gamma(\sigma,SL)$	Actual Fitness	Lower Bound Fitness	Upper Bound Fitness	Approximated Fitness	Frequency
$\langle a,b,c,e\rangle$	0	1	1.0	0.857	1.0	0.857	10
$\langle a,e\rangle$	1	1	0.8	0.8	0.8	0.8	4
$\langle a,c,b,d,e\rangle$	1	2	0.875	0.75	1	0.75	3
$\langle a,b,e\rangle$	0	0	1	1	1	1	2
$\langle d,e\rangle$	3	3	0.6	0.4	0.6	0.4	1
L	$\sim$	$\sim$	0.921	0.821	0.94	0.821	$\sim$

present several times in a trace. The repetitive patterns increase the number of unique behaviors which leads to inaccurate approximations considering a constant MC. We define the repetitive patterns of a sequence as follows.

*Definition 6 (**Repetitive Patterns**):* Let $\sigma{\in}X^*$ be a sequence. Given a non-empty strict subsequence $\sigma'{\sqsubseteq}\sigma$, we call σ' a *repetitive pattern* if $\sigma'{\cdot}\sigma'{\sqsubseteq}\sigma$. Moreover, we define function $\lambda{:}X^*{\to}\mathbb{P}(X^*)$ that receives a sequence and returns the set of all *repetitive patterns* in it.
For instance, we have $\lambda(\langle a,d,d,d,d,e\rangle){=}\{\langle d\rangle,\langle d,d\rangle\}$ and $\lambda(\langle a,f,g,f,g,e\rangle){=}\{\langle f,g\rangle\}$.

To deal with the problem of the existence of repetitive patterns, we propose to compress sequences and keep only one repetition of repetitive patterns. The sequence compression function is defined as follows.

*Definition 7 (**Sequence Compression**):* Let σ' be a *repetitive pattern* of σ and $\sigma{=}\sigma_1{\cdot}\sigma'..\sigma'{\cdot}\sigma_2$. We say that σ_c is a *compression* of σ by σ' if $\sigma_c{=}\sigma_1{\cdot}\sigma'{\cdot}\sigma_2{\cdot}$. Moreover, we define *sequence compression* function $\theta{:}(X^*\backslash\{\epsilon\}){\times}(X^*\backslash\{\epsilon\}){\to}X^*$ is a function such that $\theta(\sigma,\sigma'){=}\sigma_c$.
Therefore, if $\theta(\sigma,\sigma'){=}\sigma_c$, then $\sigma'{\notin}\lambda(\sigma_c)$. For example, $\theta(\langle a,d,d,d,d,e\rangle,\langle d\rangle){=}\langle a,d,e\rangle$.

To approximate the alignment cost, we consider both original and compressed traces of the simulated log and the original event log. Therefore, we compute $Apx(MC,\sigma){=}\min\limits_{\sigma"\in\{\theta(\sigma,\sigma')|\sigma'\in\lambda(\sigma)\}}(\Gamma(\sigma",SL{\cup}SL_c))$. However, by compressing the traces, it is possible to remove some asynchronous moves which causes to have approximated alignment cost smaller than the lower bound. Therefore, if the computed $Apx(MC,\sigma){<}LB(MC,\sigma)$, we use $Apx(MC,\sigma){=}\frac{LB(MC,\sigma)+UB(MC,\sigma)}{2}$.

To compute the fitness bounds and approximation for traces, UB, Apx, and LB values should be used instead of $\Gamma(\sigma,M)$ in Equation 1, e.g., $fitness_{LB}(MC,\sigma){=}\frac{UB(MC,\sigma)}{|\sigma|+M_s}$. To compute the lower bound for the fitness we need to use UB and to have the fitness upper bound, LB should be used. Moreover, to have fitness bounds and approximation for an event log and a model characterization, we use the weighted average of values for the variants in the event log similar to Equation 2, e.g., $Fitness_{Apx}(L,MC)=\frac{\sum\limits_{\sigma\in L}L(\sigma)\times fitness_{APX}(MC,\sigma)}{\sum\limits_{\sigma\in L}L(\sigma)}$.

In Table I, the computed bounds and the approximated fitness value for each variant and the overall event log of Figure 1 is given based on having only one simulated trace and k=2, i.e., $MC{=}(\{\langle a,b,e\rangle\},\{\epsilon,\langle a\rangle,\langle a,b\rangle,\langle a,c\rangle\},2,3\}$. The approximated fitness is 0.821 and the proposed fitness bounds are 0.94 and 0.821. By increasing $|SL|$, we expect to have more accurate approximations and bounds. Furthermore, using the δ function, we find the number of asynchronous moves for each activity which is 1 for *a*, 2 for *b*, *c*, and *d*, and 0 for *e*.

Table II: The real event logs that are used in the experiment. Artificial *start* and *end* activities are inserted in all the traces.

Event Log	**Activities#**	**Traces#**	**Variants#**
BPIC-2012 [26]	25	13087	4336
BPIC-2017 *[27]*	28	31509	15930
BPIC-2018-*Inspection* [28]	17	5485	3190
BPIC-2019 [29]	44	251734	11973
Hospital-Billing [30]	20	100000	1020
RTFM [31]	13	150370	231
Sepsis [32]	18	1050	846

V. Evaluation

In this section, we aim to explore the accuracy and the performance of the proposed method. We first explain the implementation and the experimental setting. Thereafter, the experimental results and a discussion of results are provided.

A. Implementation

To apply the proposed conformance approximation method, we implemented the *Conformance Approximation using Simulation* plug-in in the `ProM` framework[1] [7]. It takes an event log and a process model as inputs and returns the conformance approximation, its bounds, and the deviation rates of different activities. In this implementation, we consider a Petri net representation. However, other notations can be supported using our proposed method. The given Petri net can have silent transitions or duplicate labels, however, it should be sound. The user is able to adjust the maximum number of the simulated traces. Another parameter of this plug-in is the maximum length of subsequence in computation of probabilities according to Equation 3.

To apply the method on various event logs with different parameters, we ported the developed plug-in to `RapidProM`, i.e., an extension of `RapidMiner` and combines scientific work-flows with a several process mining algorithms [8].

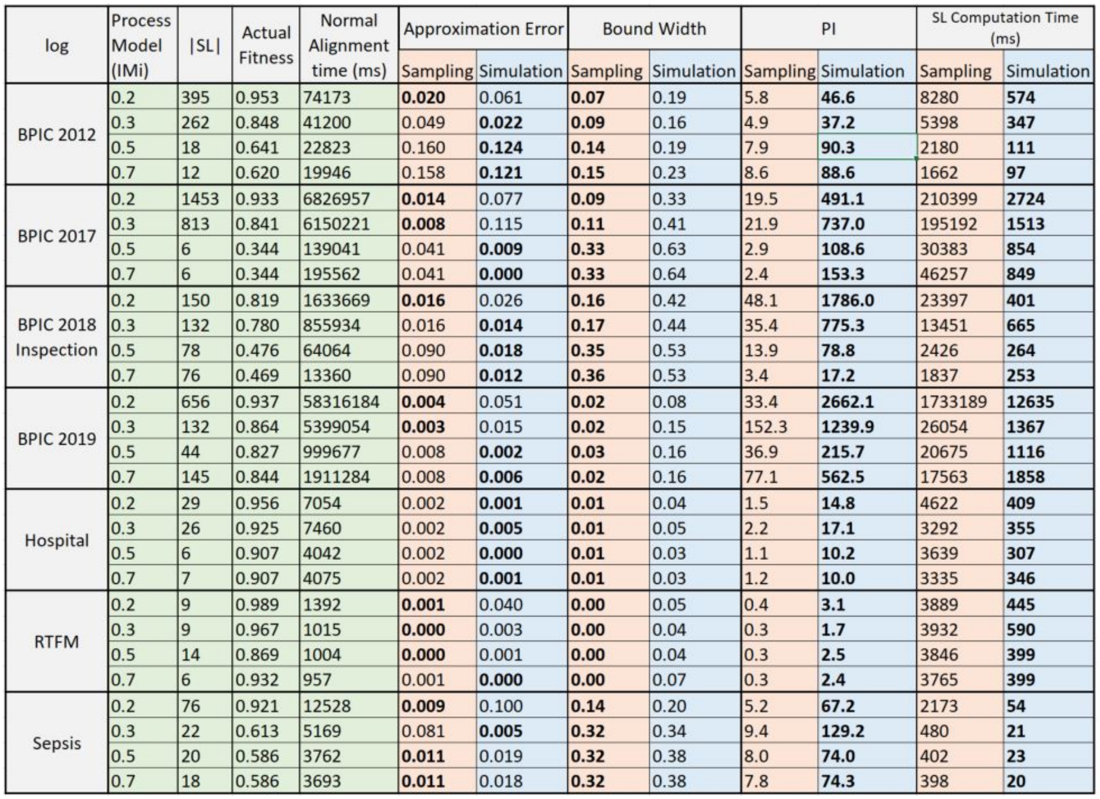
B. Experimental Setup

We applied the proposed methods on seven different real event logs. Some information about these logs is given in Table II. To discover models, we used the Inductive miner [33] with infrequent thresholds equal to 0.2, 0.3, 0.5, and 0.7.

In the first experiment, we compared the proposed method with the *sampling* method [6] that generates a subset of model traces by alignment computation of some traces in the event log. Here, we use its default setting, i.e., computing alignment

[1]svn.win.tue.nl/repos/prom/Packages/LogFiltering and http://www.promtools.org/doku.php

Table III: Comparison of approximating the conformance checking using the proposed simulation method and the sampling method that is proposed in [6]. For the sampling method, we selected the 10% of the most frequent variants in the event logs and for the simulation method, we used $|SL|$ equal to the generated model traces by the sampling method.



log	Process Model (IMi)	|SL|	Actual Fitness	Normal Alignment time (ms)	Approximation Error		Bound Width		PI		SL Computation Time (ms)	
					Sampling	Simulation	Sampling	Simulation	Sampling	Simulation	Sampling	Simulation
BPIC 2012	0.2	395	0.953	74173	**0.020**	0.061	**0.07**	0.19	5.8	**46.6**	8280	**574**
	0.3	262	0.848	41200	0.049	**0.022**	**0.09**	0.16	4.9	**37.2**	5398	**347**
	0.5	18	0.641	22823	0.160	**0.124**	**0.14**	0.19	7.9	**90.3**	2180	**111**
	0.7	12	0.620	19946	0.158	**0.121**	**0.15**	0.23	8.6	**88.6**	1662	**97**
BPIC 2017	0.2	1453	0.933	6826957	**0.014**	0.077	**0.09**	0.33	19.5	**491.1**	210399	**2724**
	0.3	813	0.841	6150221	**0.008**	0.115	**0.11**	0.41	21.9	**737.0**	195192	**1513**
	0.5	6	0.344	139041	0.041	**0.009**	**0.33**	0.63	2.9	**108.6**	30383	**854**
	0.7	6	0.344	195562	0.041	**0.000**	**0.33**	0.64	2.4	**153.3**	46257	**849**
BPIC 2018 Inspection	0.2	150	0.819	1633669	**0.016**	0.026	**0.16**	0.42	48.1	**1786.0**	23397	**401**
	0.3	132	0.780	855934	0.016	**0.014**	**0.17**	0.44	35.4	**775.3**	13451	**665**
	0.5	78	0.476	64064	0.090	**0.018**	**0.35**	0.53	13.9	**78.8**	2426	**264**
	0.7	76	0.469	13360	0.090	**0.012**	**0.36**	0.53	3.4	**17.2**	1837	**253**
BPIC 2019	0.2	656	0.937	58316184	**0.004**	0.051	**0.02**	0.08	33.4	**2662.1**	1733189	**12635**
	0.3	132	0.864	5399054	**0.003**	0.015	**0.02**	0.15	152.3	**1239.9**	26054	**1367**
	0.5	44	0.827	999677	0.008	**0.002**	**0.03**	0.16	36.9	**215.7**	20675	**1116**
	0.7	145	0.844	1911284	0.008	**0.006**	**0.02**	0.16	77.1	**562.5**	17563	**1858**
Hospital	0.2	29	0.956	7054	0.002	**0.001**	**0.01**	0.04	1.5	**14.8**	4622	**409**
	0.3	26	0.925	7460	0.002	**0.005**	**0.01**	0.05	2.2	**17.1**	3292	**355**
	0.5	6	0.907	4042	0.002	**0.000**	**0.01**	0.03	1.1	**10.2**	3639	**307**
	0.7	7	0.907	4075	0.002	**0.001**	**0.01**	0.03	1.2	**10.0**	3335	**346**
RTFM	0.2	9	0.989	1392	**0.001**	0.040	**0.00**	0.05	0.4	**3.1**	3889	**445**
	0.3	9	0.967	1015	**0.000**	0.003	**0.00**	0.04	0.3	**1.7**	3932	**590**
	0.5	14	0.869	1004	**0.000**	0.001	**0.00**	0.04	0.3	**2.5**	3846	**399**
	0.7	6	0.932	957	0.001	**0.000**	**0.00**	0.07	0.3	**2.4**	3765	**399**
Sepsis	0.2	76	0.921	12528	**0.009**	0.100	**0.14**	0.20	5.2	**67.2**	2173	**54**
	0.3	22	0.613	5169	0.081	**0.005**	**0.32**	0.34	9.4	**129.2**	480	**21**
	0.5	20	0.586	3762	**0.011**	0.019	**0.32**	0.38	8.0	**74.0**	402	**23**
	0.7	18	0.586	3693	**0.011**	0.018	**0.32**	0.38	7.8	**74.3**	398	**20**

of 10% of the most frequent variants. However, the generated model traces using this method could be less than the number of computed alignments. Therefore, we use the simulation method with $|SL|$ equal to the number of unique model behaviors that are generated by sampling method.

In the second experiment, we analyse the effect of changing the simulation parameters on the accuracy and performance of the proposed method. Therefore, we used the simulation method with $|SL|$ equals to $100, 500$, and 1000 and the subsequence length equals $1, 2, 3$, and 4. Moreover, in the last experiment, we compared the proposed simulation method, with a random simulation method. In this regard, we used the subsequence length equal to 2 and $|SL|$ equal to $10, 50, 100, 500, 1000$, and 10000 for Sepsis event log.

In all the experiments and for all methods, we used one thread of CPU. Moreover, each experiment was repeated four times, since the conformance checking and simulation times are not deterministic, and the average values are shown. For computing the normal conformance checking, we used the method that is proposed in [34].

To evaluate whether the proposed simulation method is able to improve the performance of the conformance checking process, we measure $PI{=}\frac{\text{Normal Conformance Time}}{\text{Approximated Conformance Time}}$. In this formula, a higher PI value means conformance is computed in less time. As both approximation methods need a preprocessing phase (e.g., traversing the event log and computing a subset of model behaviors), we compute PI considering the preprocessing time.

The approximation error, i.e., the difference between approximated fitness value and the actual fitness value shows how the accuracy of approximation that is computed by $AppxErr{=}|ActualFitness{-}AppxFitness|$. Also, we measure the bound width of an approximation by computing $BoundWidth{=}UBFitness{-}LBFitness$. A tighter bound width means we have more accurate bounds.

C. Experimental Result and Discussion

In Table III, we show how different approximation methods improve the performance of conformance checking. For both sampling and simulation methods, in most of the cases, we have improvement in the performance of conformance checking (i.e., $PI{>}1$). This improvement for the simulation method is much higher in all the the cases. It is mainly because, the required time to generate model traces is less than using alignments. When we have complex process models and large event logs, the improvement is higher. However, in these cases, we have a higher approximation error and the provided bounds are not accurate enough. However, the provided bounds for the conformance value are always worse when the simulation method is used. It is because, the sampling method knows the exact conformance values of the 10% of the most frequent variants. Due to a Pareto-line distribution a small number of variants may account for a large number of traces.

Table IV: Analysing the effect of the simulation size on the approximation time and the accuracy results. Here, the average value are shown when the subsequence length for computing probabilities equals to 2 and IMi=0.2.

Event Log	|SL|	Simulation Time (ms)	Approximation Time (ms)	PI	Approximation Error	Bound Width
BPIC 2019	10	40	2173	6409	0.046	0.164
	100	598	4313	3129	0.028	0.148
	1000	5400	14214	931	0.016	0.136
BPIC 2018 Inspection	10	6	393	16	0.002	0.038
	100	225	640	10	0.000	0.020
	1000	437	881	8	0.000	0.020
Hospital	10	3	376	1881	0.069	0.546
	100	218	860	876	0.018	0.475
	1000	2384	3291	196	0.004	0.426
RTFM	10	1	509	2	0.008	0.046
	100	18	491	2	0.000	0.021
	1000	230	692	2	0.000	0.011
Sepsis	10	2	46	138	0.061	0.351
	100	26	97	63	0.025	0.313
	1000	208	341	23	0.012	0.208

The result shows that we are able to have more accurate approximations using the simulation method specifically when we have more precise process models (that are discovered using a higher threshold in the Inductive miner) even with just few simulated traces. Note that the corresponding model behavior for alignment of several variants may be similar. Therefore, the sampling method usually generates fewer model traces compared to the number of computed alignments. For example, for RFTM event log and process model that is discovered using IMi=0.7, it computed alignments of 24 variants to generate 6 model traces.

For some logs, none of the methods are able to provide accurate bounds (with the applied setting). Specifically, when the process model is imprecise, the output of the proposed method is not accurate. The simulation of an imprecise process model usually generates many behaviors that may be far from variants that are in the event log. Thus, the provided bounds are far from each other which is a limitation of our method. To have more accurate results in these cases, we should simulate more traces that decrease the performance improvement.

In the next experiment, we analyze the effect of the number of simulated traces on the simulation and approximation times and the accuracy of the approximation. In this experiment, we considered subsequences in Equation 3 equals to 2 and used the process models with threshold equals to $0,2$.

The results of this experiment are shown in Table IV. The results indicate that increasing the size of $|SL|$ increases the required time for simulation and finding the distance of the most similar trace (i.e., the Approximation time). Results show that for some event logs which have simple structured process models, e.g., *RTFM*, even with 100 simulated traces, we are able to detect accurate conformance value. Moreover, for some event logs, it is not possible to generate the specified number of unique model traces, which is the reason that PI does not decrease as expected. Also, by increasing the size of the simulated traces, we provide more accurate approximations and bounds. Therefore, the user is able to trade-off between the performance improvement and accuracy of the approximated value by adjusting this parameter. We also conducted a similar experiment for analyzing the effects of changing the length of subsequences when we compute probabilities. We found that this parameter has no significant impact on the performance of the proposed method. However, when $|SL|$ is low, the length of subsequence affects the accuracy of approximation value and the provided bounds.

Table V: Comparison of using the proposed simulation method with random simulation technique [6] for Sepsis event log when the Inductive miner with threshold 0.2 is used.

	Approximation Error		Bound Width		Total Approximation Time (ms)	
|SL|	Random Simulation	Proposed Method	Random Simulation	Proposed Method	Random Simulation	Proposed Method
10	0.408	0.186	0.437	0.308	75	82
50	0.304	0.122	0.332	0.224	104	164
100	0.284	0.104	0.312	0.214	122	219
500	0.235	0.065	0.263	0.152	197	573
1000	0.219	0.053	0.248	0.133	355	1010
10000	0.171	0.023	0.200	0.092	1805	8846

Finally, we analyzed the effect of considering log behaviors in the simulation process on the accuracy of provided conformance approximation and its bounds. As the base line, we used the *random* simulation method that was originally proposed by [35] and used for conformance checking approximation in [6]. The results of this experiment are presented in Table V which show that the proposed method detects more accurate approximation and bounds. Also, the random sampling method is faster than our method. It is because there is no need to have a probability computation in this method and also it may generate traces that are already presented in SL.

VI. Conclusion

In this paper, we proposed an approximation method for computing conformance values including providing upper and lower bounds based on process model simulation. We consider a process model as a set of all possible behaviors than can be executed by the process. This assumption allows us to obtain conformance checking results even for the cases that we do not have a descriptive process model. Using the simulation method, we generate a subset of the model's behaviors. We guide the simulation method to generate traces that are more similar to the recorded behaviors in the event log. We apply these simulated traces for approximating the conformance value using the edit distance function. Moreover, the method provides lower and upper bounds for the approximated value based on the seen simulated behaviors.

To evaluate the proposed method, we implemented our technique using both ProM and RapidProM and applied our implementation to seven real event logs and 28 discovered process models. The results show that the proposed method is able to decrease the conformance checking computation time and simultaneously find approximated values close to the actual alignment value. We found that when the process model is imprecise, the accuracy of the approximation degrades. Furthermore, experiments show that considering behaviors in the event log in simulation improves the accuracy of the approximation value and bounds compared to the case that a process model is simulated randomly.

As future work, we aim to provide a platform in which gives us qualitative feedback in a faster way. Moreover, it is possible to provide an incremental approximation tool that increases the number of simulated traces incrementally and lets the end-user decide when the accuracy is enough.

Acknowledgment

We thank the Alexander von Humboldt (AvH) stiftung for funding this research.

References

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer Berlin Heidelberg, 2016.

[2] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. M. P. van der Aalst, "Alignment based Precision Checking," in *International Conference on Business Process Management*. Springer, 2012, pp. 137–149.

[3] S. J. van Zelst, A. Bolt, M. Hassani, B. F. van Dongen, and W. M. van der Aalst, "Online conformance checking: relating event streams to process models using prefix-alignments," *International Journal of Data Science and Analytics*, pp. 1–16, 2017.

[4] M. De Leoni and W. M. van der Aalst, "Data-aware process mining: discovering decisions in processes using alignments," in *Proceedings of the 28th annual ACM symposium on applied computing*. ACM, 2013, pp. 1454–1461.

[5] D. Fahland and W. M. P. van der Aalst, "Model Repair—Aligning Process Models to Reality," *Information Systems*, vol. 47, pp. 220–243, 2015.

[6] M. Fani Sani, S. J. van Zelst, and W. M. P. van der Aalst, "Conformance Checking Approximation Using Subset Selection and Edit Distance," in *Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings*, 2020, pp. 234–251. [Online]. Available: https://doi.org/10.1007/978-3-030-49435-3_15

[7] W. M. P. van der Aalst, B. van Dongen, C. W. Günther, A. Rozinat, E. Verbeek, and T. Weijters, "ProM: The Process Mining Toolkit," *BPM (Demos)*, vol. 489, no. 31, 2009.

[8] W. M. P. van der Aalst, A. Bolt, and S. van Zelst, "RapidProM: Mine Your Processes and Not Just Your Data," *CoRR*, vol. abs/1703.03740, 2017.

[9] M. Elhagaly, K. Drvoderić, R. G. Kippers, and F. A. Bukhsh, "Evolution of Compliance Checking in Process Mining Discipline," in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2019, pp. 1–6.

[10] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking*. Springer, 2018.

[11] A. Rozinat and W. M. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[12] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 182–192, 2012. [Online]. Available: https://doi.org/10.1002/widm.1045

[13] W. M. van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[14] J. Munoz-Gama, J. Carmona, and W. M. van der Aalst, "Single-entry single-exit decomposed conformance checking," *Information Systems*, vol. 46, pp. 102–122, 2014.

[15] H. M. W. Verbeek, W. M. P. van der Aalst, and J. Munoz-Gama, "Divide and Conquer: A Tool Framework for Supporting Decomposed Discovery in Process Mining," *Comput. J.*, vol. 60, no. 11, pp. 1649–1674, 2017. [Online]. Available: https://doi.org/10.1093/comjnl/bxx040

[16] D. Schuster and S. J. van Zelst, "Online Process Monitoring Using Incremental State-Space Expansion: An Exact Algorithm," in *BPM 2020, Proceedings*, ser. Lecture Notes in Computer Science. Springer, 2020. [Online]. Available: https://sebastiaanvanzelst.com/wp-content/uploads/2020/06/2020_bpm_schuster_zelst_incremental_alignments.pdf

[17] S. J. J. Leemans and A. Polyvyanyy, "stochastic-aware conformance checking: An entropy-based approach."

[18] T. Nolle, A. Seeliger, N. Thoma, and M. Mühlhäuser, "Deepalign: Alignment-based process anomaly correction using recurrent neural networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2020, pp. 319–333.

[19] F. Taymouri and J. Carmona, "A recursive paradigm for aligning observed behavior of large structured process models," in *International Conference on Business Process Management*. Springer, 2016, pp. 197–214.

[20] M. Bauer, H. van der Aa, and M. Weidlich, "Estimating Process Conformance by Trace Sampling and Result Approximation," pp. 179–197, 2019.

[21] L. Padró and J. Carmona, "Approximate Computation of Alignments of Business Processes Through Relaxation Labelling," in *International Conference on Business Process Management*. Springer, 2019, pp. 250–267.

[22] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support," *Data Knowl. Eng.*, vol. 68, no. 9, pp. 834–850, 2009. [Online]. Available: https://doi.org/10.1016/j.datak.2009.02.014

[23] A. Rogge-Solti, R. S. Mans, W. M. P. van der Aalst, and M. Weske, "Improving Documentation by Repairing Event Logs," in *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 2013, pp. 129–144.

[24] M. Camargo, M. Dumas, and O. G. Rojas, "Automated discovery of business process simulation models from event logs," vol. abs/1910.05404, 2019. [Online]. Available: http://arxiv.org/abs/1910.05404

[25] M. Pourbafrani, S. J. van Zelst, and W. M. P. van der Aalst, "Scenario-Based Prediction of Business Processes Using System Dynamics," in *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, 2019, pp. 422–439. [Online]. Available: https://doi.org/10.1007/978-3-030-33246-4_27

[26] Van Dongen, B.F. (Boudewijn), "BPI Challenge 2012," 2012. [Online]. Available: https://data.4tu.nl/repository/uuid:3926db30-f712-4394-aebc-75976070e91f

[27] Van Dongen, B.F., "BPIC 2017. Eindhoven University of Technology," 2017. [Online]. Available: https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

[28] Van Dongen, B.F. (Boudewijn) and Borchert, F. (Florian), "BPI Challenge 2018," 2018. [Online]. Available: https://data.4tu.nl/repository/uuid:3301445f-95e8-4ff0-98a4-901f1f204972

[29] Van Dongen, B.F. (Boudewijn), "BPI Challenge 2019," 2019. [Online]. Available: https://data.4tu.nl/repository/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1

[30] F. Mannhardt, "Hospital billing-event log," *Eindhoven University of Technology. Dataset*, pp. 326–347, 2017.

[31] M. De Leoni and F. Mannhardt, "Road traffic fine management process," *Eindhoven University of Technology. Dataset*, 2015.

[32] F. Mannhardt, "Sepsis cases-event log. Eindhoven University of Technology," 2016. [Online]. Available: https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[33] S. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour," in *BPI*, 2014, pp. 66–78.

[34] B. F. van Dongen, "Efficiently Computing Alignments - Algorithm and Datastructures," in *Business Process Management Workshops - BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers*, 2018, pp. 44–55.

[35] A. Rogge-Solti and M. Weske, "Prediction of business process durations using non-Markovian stochastic Petri nets," vol. 54, 2015, pp. 1–14. [Online]. Available: https://doi.org/10.1016/j.is.2015.04.004