Conformance Checking Approximation using Subset Selection and Edit Distance

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

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

Summary. Conformance checking techniques let us find out to what degree a process model and real execution data correspond to each other. In recent years, alignments have proven extremely useful in calculating conformance statistics. Most techniques to compute alignments provide an exact solution. However, in many applications, it is enough to have an approximation of the conformance value. Specifically, for large event data, the computation time for alignments is considerably long using current techniques which makes them inapplicable in reality. Also, it is no longer feasible to use standard hardware for complex process models. This paper, proposes new approximation techniques to compute approximated conformance checking values close to exact solution values in less time. These methods also provide upper and lower bounds for the approximated alignment value. Our experiments on real event data show that it is possible to improve the performance of conformance checking by using the proposed methods compared to using the state-of-the-art alignment approximation technique. Results show that in most of the cases, we provide tight bounds, accurate approximated alignment values, and similar deviation statistics.

Key words: Process Mining · Conformance Checking Approximation · Alignment· Subset Selection · Edit Distance · Simulation·

1 Introduction

One of the main branches of process mining is conformance checking, aiming at investigating conformity of a discovered/designed process model w.r.t, real process executions [1]. This branch of techniques is beneficial to detect deviations and to measure how accurate a discovered model is. In particular, the techniques in this branch are able to check conformance based on process modeling formalisms that allow for describing concurrency, i.e., the possibility to specify order-independent execution of activities. Early conformance checking techniques, e.g., "token-based replay" [2], often lead to ambiguous and/or unpredictable results. Hence, alignments [3] were developed with the specific goal to explain and quantify deviations in a non-ambiguous manner. Alignments have rapidly turned into the de facto standard conformance checking technique [4]. Moreover, alignments serve as a basis for techniques that link event data to process models, e.g., they support performance analysis, decision mining [5], business process model repair [6] and prediction techniques. However, computing alignments is time consuming on real large event data, which makes it unusable in reality.

In many applications, we need to compute alignment values several times, e.g., if we want to have a suitable process model for an event log, we need to discover many process models using various process discovery algorithms with different settings, and, measure how each process model fits with the event log using alignment techniques. As normal alignment methods require considerable time for large event data, analyzing many candidate process models is impractical. Consequently, by decreasing the alignment computation time, we can consider more candidate models in a limited time. Moreover, in several cases, we do not need to have accurate alignment values, i.e., it is sufficient to have a quick approximated value or a close lower/upper bound for it.

In this paper, we propose several conformance checking approximation methods that provide approximated alignment values plus lower and upper bounds for the actual alignment value. The underlying core idea, is to consider just a subset of the process model behavior, instead of its all behavior. The methods additionally return problematic activities, based on their deviation rates. Using these methods, 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.

We implemented the methods in two open-source process mining tools and applied them on several large real event data and compared them with the state-of-the-art alignment approximation method. The results show that using some of proposed methods, we are able to approximate alignment values faster and at the same time the approximated values are very close to actual alignment values.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. Section 3 defines preliminary notation. We explain the main method in Section 4 and evaluate it in Section 5. Section 6 concludes the paper.

2 Related Work

Several process mining techniques exists, ranging from process discovery to prediction. We limit related work to the field of conformance checking and sampling techniques in the process mining domain. We refer to [1] for an overview of process mining.

In [7], the authors review the conformance checking techniques in process mining domain. In [8] different methods for conformance checking and its applications are covered. Early work in conformance checking uses token-based replay [2]. The techniques replay a trace of executed events in a Petri net and add missing tokens if transitions are not able to fire. After replay, a conformance statistic is computed based on missing and remaining tokens. Alignments were introduced in [9] and have rapidly developed into the standard conformance checking technique. In [10, 11], decomposition techniques are proposed for alignment computation. Moreover, [12] proposes a decomposition method to find an approximation of the alignment in a faster time. Applying decomposition techniques improves computation time, i.e., the techniques successfully use the divide-and-conquer paradigm, however, these techniques are primarily beneficial when there are too many unique activities in the process [13]. Recently, general approximation schemes for alignments, i.e., computation of near-optimal alignments, have been proposed [14]. Finally, the authors in [4] propose to incrementally compute prefix-alignments, i.e., enabling real-time conformance checking for event data streams.

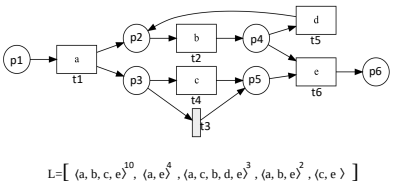

Fig. 1: An example Petri net and an event log in a multiset view.

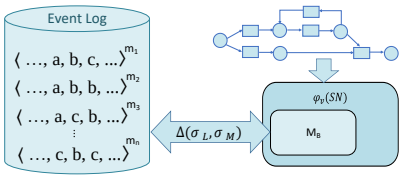

Fig. 2: Overview of the proposed approach. It uses $M_B \subseteq \phi_v(SN)$ to approximate alignment costs.

A limited amount of work considers the use of sampling in process mining. In [15], the authors proposed a sampling approach based on Parikh vectors of traces to detect the behavior in the event log. In [16], the authors recommend a trace-based sampling method to decrease the discovery time and memory footprint. In [17], a trace-based sampling method, specifically for the Heuristic miner [18], is proposed. In these sampling methods, we have no control on the size of the final sampled event data, and, depending on the defined behavioral abstraction, the methods may select almost all the process instances. Finally, all these sampling methods are unbiased and lead to non-deterministic results. In [19], 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 focuses on alignment value approximation. [20] proposes sampling the event log and applying conformance checking on the sampled data. The method increases the sample size until the approximated value is accurate enough. However, the method does not guarantee the accuracy of the approximation, e.g., by providing bounds for it. In Section 5, we show that if there is lot of unique behavior in the event log, using this method, the approximation time exceeds the computation time for finding the alignment value. The authors of [21] propose a conformance approximation method that applies relaxation labeling methods on 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. In this paper, we propose multiple alignment approximation methods that increase the conformance checking performance. The methods also provide bounds for alignment values and a deviation ratio of problematic activities.

3 Preliminaries

In this section, we briefly introduce basic process mining and, specifically, conformance checking terminology and notations that ease the readability of this paper.[1]

Given a set X, a multiset $\mathcal{B}$ over X is a function $\mathcal{B}\colon X \to \mathbb{N}_{\geq 0}$ that allows certain elements of X appear multiple times. $\overline{\mathcal{B}} = \{e \in X \mid \mathcal{B}(e) > 0\}$ is the set of elements present in the multiset. The set of all multisets over a set X is written as $\mathcal{B}(X)$.

[1] For some concepts, e.g., labeled Petri net and System net please use the definitions in [10]

Given a system net SN, $\phi_f(SN)$ is the set of all complete firing sequences of SN and $\phi_v(SN)$ is the set of all possible *visible* traces, i.e., complete firing sequences starting its initial marking and ending in its final marking projected onto the set of observable activities (not silent transitions, e.g., t_3 in Fig. 1). To measure how a trace aligns to a process model, we need to define the notation of moves. A *move* is a pair (x, t) where the first element refers to the log and the second element refers to the corresponding transition in the model.

Definition 1 (Legal Moves). *Let $L \in \mathcal{B}(\mathcal{A}^*)$ be an event log, where $\mathcal{A}$ is the set of activities and let T be the set of transitions in the model. Moreover, let l be a function that returns the label of each transition. $A_{LM} = \{(x, (x, t)) \mid x \in \mathcal{A} \wedge t \in T \wedge l(t) = x\} \cup \{(\gg, (x, t)) \mid t \in T \wedge l(t) = x\} \cup \{(x, \gg) \mid x \in \mathcal{A}\}$ is the set of* legal moves.

For example, (a, t_1) means that both log and model make an "a move" and the move in the model is caused by the occurrence of transition t_1 (as the label of t_1 is a). Note that $\gg$ indicates "no move" in log/model trace. Now, we define Alignment as follows [10].

Definition 2 (Alignment). *Let $\sigma_L \in L$ be a log trace and $\sigma_M \in \phi_f(SN)$ a complete firing sequence of a system net SN. An alignment of σ_L and σ_M is a sequence of pairs $\gamma \in A^*_{LM}$ such that the projection on the first element (ignoring $\gg$) yields σ_L and the projection on the second element (ignoring $\gg$ and transition labels) yields σ_M.*

An alignment is a sequence of legal moves such that after removing all $\gg$ symbols, the top row corresponds to a trace in the event log and the bottom row corresponds to a complete firing sequence in $\phi_f(SN)$. The middle row corresponds to a visible path when ignoring the τ steps, i.e., corresponding to silent transitions (e.g., t_3 in Fig. 1). For silent transitions, there is no corresponding recorded event in the log. The following alignments relate to $\sigma_L = \langle a, c, b, d, e\rangle$ and the Petri net in Fig. 1.

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|}\hline a & \gg & c & b & d & e \\ \hline a & \tau & \gg & b & \gg & e \\ \hline t_1 & t_3 & & t_2 & & t_6 \\ \hline\end{array} \quad \gamma_2 = \begin{array}{|c|c|c|c|c|}\hline a & c & b & d & e \\ \hline a & c & b & \gg & e \\ \hline t_1 & t_4 & t_2 & & t_6 \\ \hline\end{array}$$

By considering the label of visible transitions of an alignment, we find the corresponding model trace, e.g., the model trace of γ_1 is $\langle a, b, e\rangle$. To quantify the costs of misalignments we introduce a move cost function δ. *Synchronous moves*, i.e., moves that are similar in the trace and the model, have no costs, i.e., for all $x \in \mathcal{A}$, $\delta((x, (x, t))=0$. Moves in model only have no costs if the transition is invisible, i.e., $\delta(\gg, t) = 0$ if $l(t)=\tau$.

Definition 3 (Cost of Alignment). *Cost function $\delta \in A_{LM} \rightarrow \mathbb{R} \geq 0$ assigns costs to legal moves. The cost of an alignment $\gamma \in A^*_{LM}$ is $\delta(\gamma) = \Sigma_{(x,y)\in\gamma}\delta(x, y)$.*

In this paper, we use a standard cost function δ_S that assigns unit costs: $\delta_S(\gg, t) = \delta_S(x, \gg) = 1$ if $l(t) \neq \tau$. In the above example alignments, $\delta_S(\gamma_1) = 2$ and $\delta_S(\gamma_2) = 1$. Given a log trace and a system net, we may have many alignments. To select the most appropriate one, we select an alignment with the lowest total costs.

Definition 4 (Optimal Alignment). *Let $L \in \mathcal{B}(\mathcal{A}^*)$ be an event log and let SN be a system net with $\phi_v(SN) \neq \emptyset$.*

- *For* $\sigma_L \in L$, $\Gamma_{\sigma_L,SN} = \{\gamma \in A^*_{LM} | \exists_{\sigma_M \in \phi_f(SN)}$ *is an alignment of* σ_L *and* $\sigma_M\}$.
- *An alignment* $\gamma \in \Gamma_{\sigma_L,SN}$ *is optimal for trace* $\sigma_L \in L$ *and system net* SN *if for any alignment* $\gamma' \in \Gamma_{\sigma_L,SN} : \delta(\gamma') \geq \delta(\gamma)$.
- $\gamma_{SN} \in \mathcal{A}^* \rightarrow A^*_{LM}$ *is a mapping that assigns any log trace* σ_L *to an optimal alignment, i.e.,* $\gamma_{SN}(\sigma_L) \in \Gamma_{\sigma_L,SN}$ *and* $\gamma_{SN}(\sigma_L)$ *is an optimal alignment.*
- $\lambda_{SN} \in \mathcal{A}^* \rightarrow \mathcal{A}^*$ *is a mapping that assigns any log trace* σ_L *to visible activities of the model trace of the optimal alignment.*

In the running example, $\gamma_{SN}(\langle a, c, b, d, e\rangle)=\gamma_2$ (γ_2 is optimal), and $\lambda(\langle a, c, b, d, e\rangle)=\langle a, c, b, e\rangle$ is the corresponding model trace for the optimal alignment.

We can compute the distance of two traces (or two sequences) faster using the adapted version of Levenshtein distance [22]. Suppose that $\sigma, \sigma' \in \mathcal{A}^*$, Edit Distance function $\triangle(\sigma, \sigma')\rightarrow\mathbb{N}$ returns the minimum number of edits that are needed to transform σ to σ'. As edit operations, we allow deletion/insertion of an activity (or a transition label) in a trace, e.g., $\triangle(\langle a, c, f, e\rangle, \langle a, f, c, a\rangle)=4$, corresponds to two deletions and two insertions. This measure is symmetric, i.e., $\triangle(\sigma, \sigma') = \triangle(\sigma', \sigma)$. It is possible to use the $\triangle$ function instead of the standard cost function. Thus, $\triangle$ and δ_S return same distance values. The $\triangle$ function is expendable from unit cost (i.e., δ_S) to another cost by giving different weights to insertion and deletion of different activities.

In [23], it is explained that the Levenshtein metric before normalization satisfies the triangle inequality. In other words, $\triangle(\sigma, \sigma') \leq \triangle(\sigma, \sigma'') + \triangle(\sigma'', \sigma')$. Moreover, suppose that S is a set of sequences, $\Phi(\sigma_L, S) = \min_{\sigma_M \in S} \triangle(\sigma_L, \sigma_M)$ returns the distance of the most similar sequence in S for σ_L.

Let $\phi_v(SN)$ is a set of all visible firing sequences in SN, and $\gamma_{SN}(\sigma)$ is an optimal alignment for sequence σ. It is possible to use $\Phi(\sigma, \phi_v(SN))$ instead of $\delta_S(\gamma_{SN}(\sigma))$ [2]. Using the edit distance function, we are able to find which activities are required to be deleted or inserted. So, not only the cost of alignment; but, the deviated parts of the process model (except invisible transitions) are also detectable using this function.

It is possible to convert misalignment costs into the fitness value using Equation 1. It normalizes the cost of optimal alignment by one deletion for each activity in the trace and one insertion for each visible transition in the shortest path of model (SPM). The fitness between an event log L and a system net SN (i.e., $Fitness(L, SN)$) is a weighted average of traces' fitness.

$$fitness(\sigma_L, SN) = 1 - \frac{\delta(\gamma_{SN}(\sigma))}{|\sigma_L| + \min_{\sigma_M \in \phi_f} (|\sigma_M|)} \tag{1}$$

4 Approximating Alignments using Subset of Model Behavior

As computational complexity of computing alignment is exponential in the number of states and the number of transitions, it is impractical for larger petri nets and event logs [24]. Considering that the most time consuming part in the conformance checking procedure is finding an optimal alignment for each $\sigma_L \in L$ and the system net SN leads us to propose an approximation approach that requires fewer alignment computations. The overview of the proposed approach is presented in Fig. 2. We suggest to use $M_B \subseteq$

[2] Because of the page limit, we do not provide the proof of this statement here.

$\phi_v(SN)$ instead of the whole $\phi_v(SN)$ and apply the edit distance function instead of δ_S. In the following lemma, we show that using this approach, we have an upper bound for the cost of alignment (i.e., a lower bound for the fitness value).

Lemma 1 (Alignment Cost Upper Bound). *Let $\sigma_L \in \mathcal{A}^*$ is a log trace, and $\sigma_M \in \phi_v(SN)$ is a visible firing sequence of SN. We have $\delta_S(\gamma_{SN}(\sigma_L)) \leq \triangle(\sigma_L, \sigma_M)$ where $\gamma_{SN}(\sigma_L)$ is the optimal alignment.*

Proof: We shown that $\triangle(\sigma_L, \sigma_M) = \delta_S(\gamma)$, so we have $\triangle(\sigma_L, \sigma_M) \geq \delta_S(\gamma_{SN}(\sigma_L))$. Therefore, if $\delta_S(\gamma_{SN}(\sigma_L)) > \triangle(\sigma_L, \sigma_M)$, $\gamma_{SN}(\sigma_L)$ is not an optimal alignment. Consequently, if we use any $M_B \subseteq \phi_v(SN)$, $\Phi(\sigma_L, M_B)$ returns an upper bound for the cost of optimal alignment.

Here, we explain the main components of our proposed approach, i.e., constructing a subset of model behavior (M_B) and computing the approximation.

4.1 Constructing Model Behavior (M_B)

As explained, we propose to use M_B i.e., a subset of visible model traces to have an approximated alignment. An important question is how to construct M_B. In this regard, we propose two approaches, i.e., *simulation* and *candidate selection*.

1) Simulation: The subset of model traces can be constructed by simulating the process model. In this regard, having a system net and the initial and final markings, we simulate some complete firing sequences. Note that we keep only the visible firing sequences in M_B. It is possible to replay the Petri net randomly or by using more advanced methods, e.g., stochastic petri net simulation techniques. This approach is fast; but, we are not able to guarantee that by increasing the size of M_B we will obtain the perfect alignment (or fitness) value, because the model traces are able to be infinite. Another potential problem of this method is that the generated subset may be far from traces in the event log that leads to have an inaccurate approximation.

2) Candidate Selection: The second method to construct M_B is computing the optimal alignments of selected traces in the event log and finding the corresponding model traces for these alignments. In this regard, we first select some traces (i.e., candidates) from the event log L and put them in L_C. Then for each $\sigma_L \in L_C$ we find the optimal alignment and insert $\lambda_{SN}(\sigma_L)$ to M_B. Thereafter, for other traces $\sigma'_L \in L'_C$ (i.e., $L'_C = L - L_C$), we will use M_B and compute $\Phi(\sigma'_L, M_B)$.

As the triangle inequality property holds for the edit distance function, it is better to insert $\lambda_{SN}(\sigma_L)$ in M_B instead of considering σ_L. To make it more clear, let σ_L be a log trace, SN is a system net, and $\sigma_M = \lambda_{SN}(\sigma_L)$ is the corresponding visible model trace for an optimal alignment of σ_L and SN. According to the triangle inequality property, for any trace $\sigma \in L$, we have $\triangle(\sigma, \sigma_M) \leq \triangle(\sigma, \sigma_L) + \triangle(\sigma_L, \sigma_M)$. So, the cost of transforming σ_L to σ_M is less than the cost of transforming it to σ_L and then to σ_M. As $\Phi(\sigma_L, M_B)$ returns the minimum cost of the most similar sequence in M_B to σ_L, putting directly the alignments of traces M_B causes to have a smaller upper bound for alignment cost. Moreover, it is possible to have $\lambda_{SN}(\gamma_{SN}(\sigma_1)) = \lambda_{SN}(\gamma_{SN}(\sigma_2))$ for $\sigma_1 \neq \sigma_2$. Therefore, by inserting $\lambda_{SN}(\sigma_1)$ instead of σ_1 in M_B, we will have M_B with fewer members that increases the performance of the approximation.

Table 1: Result of using the proposed approximation method for the event log that is given in Fig. 1, using $M_B = \{\langle a, b, e\rangle, \langle a, b, c, e\rangle\}$.

Trace	$\delta_S(\gamma_{SN})$	$\Phi(\sigma, M_B)$	Actual Fitness	LBoundFitness	UBoundFitness	AppxFitness	Freq
$\langle a, b, c, e\rangle$	0	0	1	1	1	1	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	0.875	3
$\langle a, b, e\rangle$	0	0	1	1	1	1	2
$\langle c, e\rangle$	2	2	0.5	0.5	0.8	0.65	1
L	$\sim$	$\sim$	0.916	0.898	0.95	0.924	$\sim$

To select the candidate traces in L_C, we propose three different methods. We can select these traces *randomly* or based on their *frequency* in the event log (i.e, $L(\sigma_L)$). The third possible method is to apply a *clustering* algorithm on the event log and put the traces in K different clusters based on their control flow information. We then select one trace, i.e., medoid for each cluster that represents all cluster's members. It is expected that by using this approach, the detected bounds will be more accurate.

4.2 Computing Alignment Approximation

After constructing M_B, we use it for all traces in the L'_C. Note that for the *simulation* method, $L_C = \emptyset$ and $L'_C = L$. Moreover, for the *candidate selection* method, we use the alignment values that already computed by in constructing M_B. To compute the lower bound for the fitness value, we compute the fitness value of all of the $\sigma \in L'_C$ using $\Phi(\sigma, M_B)$. Afterwards, based on the weighted average of this fitness and alignments that are computed in the previous part, the lower bound for the fitness value is computed.

For the upper bound of fitness value, we compare the length of each trace in L'_C with the shortest path in the model (i.e., SPM). To find SPM, we compute the cost of the optimal alignment for an empty trace (i.e., $\langle\rangle$) and the system net. In the example that is given in Fig. 1, $SPM = 3$. If the length of a trace is shorter than the ***SPM***, we know that it needs at least $SPM - \sigma_L$ insertions to transform to one of model traces in $\phi_v(SN)$. Otherwise, we consider at least 0 edit operation for that trace. Because, it is possible that there is a model trace s.t. $\sigma_M \in \phi_v(SN)$ and $\sigma_M \notin M_B$ and it perfectly fits to the log trace. After computing the upper bound values for all traces in L'_C, based on the weighted average of them and the computed fitness values of traces in L_C, we compute the upper bound value for fitness.

To compute the approximation values, for each trace in $\sigma \in L'_C$, we compute the $\Phi(\sigma, M_B)$ and compare it to the average fitness value of L_C. If the fitness value of the new trace is higher than $Fitness(L_C, SN)$, we consider $\Phi(\sigma, M_B)$ as the approximated fitness value; otherwise, $Fitness(L_C, SN)$ will be considered for the approximation. Similar to the bounds, we use the weighted averages of fitness values of L_C and L'_C to compute the approximated fitness value of whole event log. Note that for the simulation method that $L_C = \emptyset$, the approximated fitness value for each trace (and for the whole event log) is equal to the lower bound.

Finally, the proposed method returns the number of asynchronous (i.e., deletions and insertions) and synchronous moves for each activity in the event log. This information helps the data analyst to find out the source of deviations.

In Table 1, the computed bounds and the approximated fitness value for each trace and the overall event log of Fig. 1 is given based on $M_B = \{\langle a, b, e\rangle, \langle a, b, c, e\rangle\}$. This M_B is possible to be gained by computing the alignment of the two most frequent traces in the event log or by simulation. Moreover, *LBoundFitness*, *UBoundFitness* and *ApxFitness* show the lower bound, the upper bound and the approximation of the fitness value respectively. The approximated fitness will be 0.924 that its accuracy equals to 0.008. The proposed bounds are 0.95 and 0.898. Furthermore, the method returns the number of insertion and deletions that are 1 insertion for *a*, 5 insertion for *b*, 3 deletions for *c*, 3 deletions for *d*, and nothing for *e*.

By increasing $|M_B|$, we expect to have more accurate approximations and bounds. But, increasing the $|M_B|$ for the candidate selection approach increases the number of required alignments computations and consequently increases the computation time.

5 Evaluation

In this section, we aim to explore the accuracy and the performance of our methods. We first explain the implementation, and, subsequently, we explain the experimental setting. Finally, the experimental results and some discussions will be provided.

5.1 Implementation

To apply the proposed conformance approximation method, we implemented the *Conformance Approximation* plug-in in the `ProM` [25] framework[3]. It takes an event log and a Petri net as inputs and returns the conformance approximation, its bounds, and the deviation rates of different activities. In this implementation, we let the user adjusts the size of M_B and the method to select and insert model traces in it (i.e., *simulation* and alignment of selected candidates). If the user decides to use alignments for creating model behavior, she can select candidates based on their *frequency*, *random*, or using the *clustering* algorithm. For finding the distance of a log trace and a model trace, we used the *edit distance* function, which is an adapted version of the Levenshtein distance [22]. To cluster traces, we implement the K-Medoids algorithm that returns one trace as a candidate for each cluster [26] based on their edit distance.

To apply the methods 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 [27].

5.2 Experimental Setup

We applied the proposed methods on eight different real event logs. Some information about these event logs is given in Table 2. Here, *uniqueness* refers to $\frac{Variant\#}{Trace\#}$.

For process discovery, we used the Inductive Miner [34] with infrequent thresholds equal to 0.3, 0.5, and 0.7. We applied conformance approximation methods with different settings. In this regard, an approximation parameter is used with values equal to 1, 2, 3, 5, 10, 15, 20, 25, and 30. This value for the *Simulation* method is the number of

[3] `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#	Uniqueness
BPIC-2012 [28]	23	13087	4336	138	0.33
BPIC-2018-*Department* [29]	6	29297	349	19	0.01
BPIC-2018-*Inspection* [29]	15	5485	3190	67	0.58
BPIC-2018-*Reference* [29]	6	43802	515	15	0.01
BPIC-2019 [30]	42	251734	11973	498	0.05
Hospital-Billing [31]	18	100000	1020	143	0.01
Road [32]	11	150370	231	70	~ 0
$Sepsis$ [33]	16	1050	846	115	0.81

simulated traces times $|L|$, and for the *candidate selection* methods (i.e., *clustering*, *frequency*, and *random*), it shows the relative number of selected candidates, i.e., $\frac{|L_C|}{|L|}$. We also compared our proposed method with the *statistical* sampling method [20]. The approximation parameter for this method determines the size and the accuracy of sampling and we consider $\epsilon = \delta =$ approximation parameter $\times\ 0.001$. We did not consider [12] in the experiments, as it does not improve the performance of normal computation of alignment [35] for event logs which have few unique activities using the default setting. Even for some event logs with lots of unique activities in [12], the performance improvement of our methods is higher. Because of the page limit, we do not show results of this experiment here.

In all experiments and for all methods, we used eight threads of CPU. Moreover, each experiment was repeated four times, since the conformance checking time is not deterministic, and the average values are shown.

To evaluate how the conformance approximation is able to improve the performance of the conformance checking process, we used the $PI = \frac{\text{Normal Conformance Time}}{\text{Approximated Conformance Time}}$. In this formula, a higher PI value means conformance is computed in less time. As all our proposed methods need a preprocessing phase (e.g., for clustering the traces), we compute the PI with and without the preprocessing phase.

The accuracy of the approximation, i.e., the difference between approximated conformance value and the actual fitness value shows how close is the approximated fitness to the actual fitness value that is computed by $Accuracy = |AccFitness - ApproxFitness|$. Also, we measure the distance of the provided upper and lower bounds. The bound width of an approximation is computed by $BoundWidth = UBFitness - LBFitness$. Tighter bound widthes means that we have more accurate bounds.

5.3 Experimental Result and Discussion

In Fig. 3, we show how different approximation methods improve the performance of conformance checking. For most of the cases, the improvement is higher for the *simulation* method. It is because, the most time consuming part in conformance checking is computing the optimal alignment. As in the *simulation* method, there is no need to do any alignment computation, it is faster than any other method. For some event logs, the *statistical* sampling method [20] is not able to provide the approximation faster than the normal conformance checking (i.e., $PI < 1$). It happens because, this method is not able to benefit from the parallel computing of alignment and after each alignment computation it needs to check if it needs to do more alignment or not. For the *statistical* method, decreasing approximation parameter leads to more precise approximations;

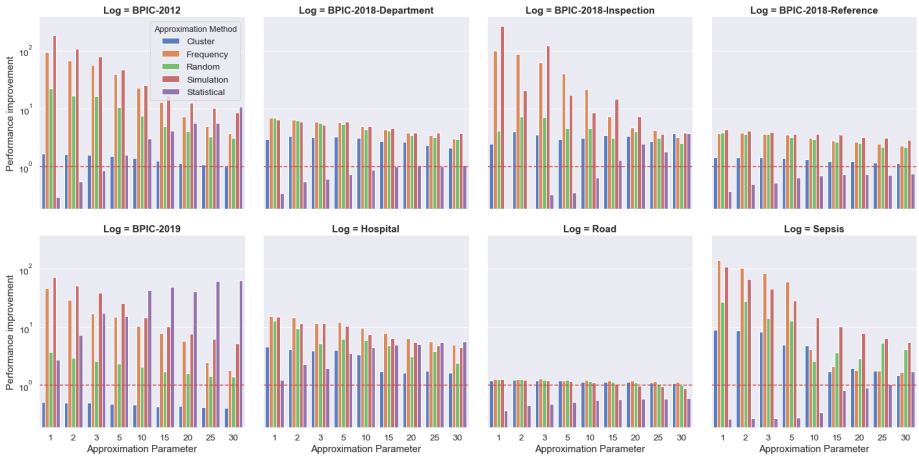

Fig. 3: Performance improvement with consideration of preprocessing time.

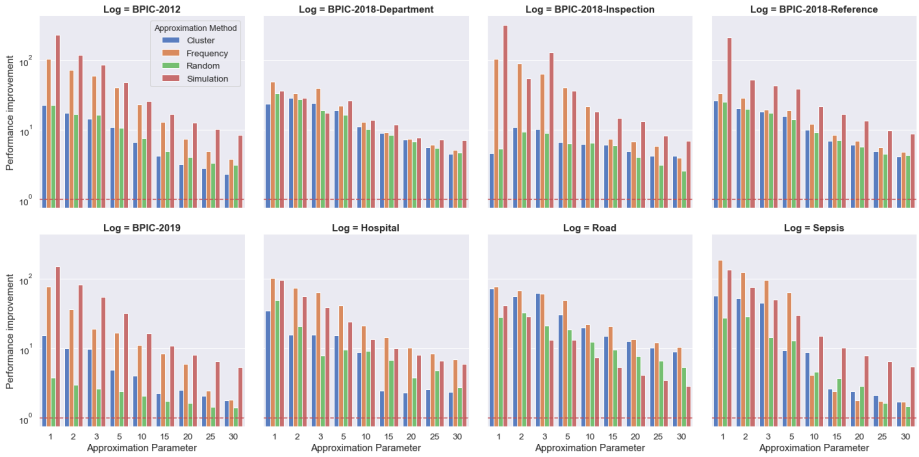

Fig. 4: Performance improvement without consideration of preprocessing time.

however, it causes to have less *PI* value. Among the *candidate selection* methods, using the *frequency* method usually leads to a higher *PI* value.

For some event logs, e.g., *Road*, none of the method has a high *PI* value. It happens because in Fig. 3, we consider the preprocessing time. The preprocessing time corresponds to choosing the candidate traces and simulating the process model behaviors that needs to be done once per each event log or process model. For the candidate selection methods, this phase is independent of process models and for doing that we do not need to consider any process model. For the simulation method, this phase is independent of the given event log. Thus, we are able to do the preprocessing step before conformance approximation. If we use some event log standards such as MXML and Parquet, we do not need to preprocess the event log for the *frequency* and *random* method because we know the number of variants and their frequency beforehand.

Table 3: The average accuracy of approximation for conformance values when we use different approximation methods. Here we used different Inductive miner thresholds.

Approximation Method			Cluster	Frequency	Random	Simulation	Statistical
Event Log	IMi	Fitness					
BPIC-2012	0.3	0.874	0.001	0.001	0.073	0.360	0.002
	0.5	0.813	0.006	0.022	0.011	0.302	0.004
	0.7	0.755	0.008	0.032	0.037	0.271	0.005
BPIC-2018-Department	0.3	0.962	0.005	0.006	0.016	0.000	0.004
	0.5	0.962	0.005	0.006	0.013	0.000	0.005
	0.7	0.962	0.005	0.006	0.018	0.000	0.003
BPIC-2018-Inspection	0.3	0.886	0.003	0.008	0.007	0.446	0.008
	0.5	0.853	0.006	0.012	0.013	0.429	0.005
	0.7	0.800	0.007	0.021	0.027	0.370	0.003
BPIC-2018-Reference	0.3	0.943	0.006	0.006	0.059	0.000	0.004
	0.5	0.943	0.006	0.006	0.051	0.000	0.003
	0.7	0.943	0.006	0.006	0.048	0.000	0.005
BPIC-2019	0.3	0.905	0.014	0.001	0.031	0.408	0.004
	0.5	0.930	0.015	0.001	0.018	0.419	0.003
	0.7	0.930	0.015	0.001	0.016	0.418	0.002
Hospital	0.3	0.991	0.002	0.002	0.031	0.324	0.002
	0.5	0.747	0.001	0.001	0.055	0.117	0.003
	0.7	0.573	0.003	0.001	0.017	0.013	0.002
Road	0.3	0.992	0.003	0.002	0.069	0.082	0.003
	0.5	0.758	0.002	0.002	0.023	0.046	0.007
	0.7	0.717	0.001	0.002	0.004	0.069	0.005
Sepsis	0.3	0.993	0.001	0.004	0.006	0.456	0.001
	0.5	0.988	0.003	0.006	0.003	0.414	0.003
	0.7	0.891	0.011	0.015	0.010	0.260	0.003

In Fig. 4, we show the performance improvement without considering the preprocessing time. As the *statistical* sampling method does not have preprocessing phase, it is not shown in this figure. It is shown that there is a linear decrement in improvement of the *candidate selection* methods by increasing the approximation parameter. It is expectable, as increasing in this parameter for candidate selection methods means more optimal alignment computations that requires more time. For example, by considering 5 for this parameter, means that we need to compute 5% of all optimal alignments of the normal conformance checking. Therefore, it is expected that the approximated conformance value will be computed in 20 times faster than using normal alignment.

After analyzing the performance improvement capabilities of the proposed methods, in Table 3, we compare the accuracy of their approximations. In this regard, the average accuracy values of the approximated conformance values are shown in this table. The lower value means a higher accuracy or in other words, the approximated fitness value is closer to the actual fitness value. In this table, *Fitness* shows the actual fitness value when the normal conformance checking method is used. We used different values for the approximation parameter as explained in Section 5.2. The results show that for most of the event logs the accuracy of the *simulation* method is not good enough. However, for *BPIC*-2018-*Reference* and *BPIC*-2018-*Department*, that have simpler process models, using this method, we generated almost all the model behavior (i.e., $M_B = \phi_v$) and obtain perfect accuracy. Results show that if we use the *statistical*, and *frequency* methods, we usually obtain accuracy value below 0.01 which is acceptable for many applications. Among the above methods, results of the statistical sampling method are more stable and accurate. However, the accuracy of candidate selection methods is usually improved by using a higher approximation parameter.

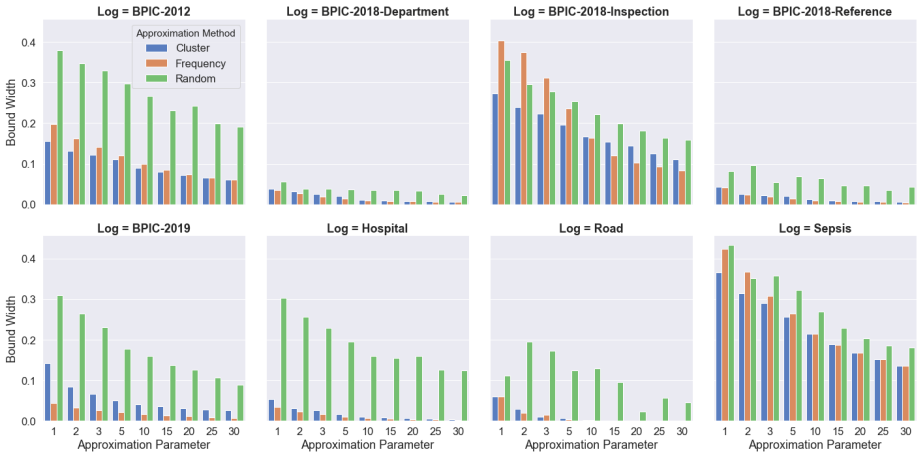

Fig. 5: The average of bound width using different approximation methods.

In the next experiment, we aim to evaluate the provided bounds for the approximation. Fig. 5 shows how increasing the value of the approximation parameter increases the accuracy of the provided lower and upper bounds. As the *statistical* method does not provide any bounds, we do not consider it in this experiment. The *simulation* method is not able to provide tight bound widths for most of the event logs. For most of the event logs, the *frequency* method results in tighter bounds. However, for event logs like *Sepsis* which there is no high frequent trace-variant, the *clustering* method provides more accurate bounds. If there are high frequent variants in the event log, it is recommended to use the *frequency* approximation method. Note that, for all methods, by increasing the value of approximation parameter, we decrease the bound width.

Considering both Fig. 4 and Fig. 5, we observe that there is a trade-off between the performance and the accuracy of the approximation methods. By increasing the number of visible traces in M_B, we need more time to approximate the fitness value; but, we will provide more accurate bounds. In the case that we set the approximation parameter to 100, the bound width will be zero; however, there will not any improvement in performance of the conformance checking. By adjusting the approximation parameter, the end user is able to specify the performance improvement.

Fig. 5 shows that for some event logs like *Sepsis* and *BPIC*-2018-*Inspection*, none of the approximation methods are able to provide tight bounds. That happens because in these event logs not only do we have lots of unique traces; but, also these traces are not close to each other. In Table 4, we show the average of edit distance of the most similar trace in the event logs that equals to $Average_{\sigma \in L} \Phi(\sigma, L - \sigma)$. If the traces in an event log are similar to each other, we are able to provide tight bounds by the approximation methods. This characteristic of the event log can be analyzed without any process model before the approximation. Therefore, it is expected to use more traces in M_B when the traces are not similar. Using this preprocessing step, user is able to adjust the approximation parameter easier.

Table 4: The average similarity of traces in different event logs.

BPIC-2012	Department	Inspection	References	BPIC-2019	Hospital	Road	Sepsis
3.686	1.224	3.269	1.481	5.108	1.745	1.113	3.956

Table 5: Comparison of deviation ratio of the six most problematic activities using normal alignment (*Real*) and the *frequency* based approximation method (*Appx*).

	BPIC-2012		Department		Inspection		References		BPIC-2019		Hospital		Road		Sepsis	
	Appx	Real	Appx	Real	Appx	Real	Appx	Real	Appx	Real	Appx	Real	Appx	Real	Appx	Real
Activity 1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.96	1.00	1.00
Activity 2	1.00	1.00	0.53	0.53	1.00	1.00	1.00	0.50	1.00	1.00	1.00	0.95	1.00	0.96	1.00	1.00
Activity 3	1.00	0.94	0.37	0.37	1.00	1.00	0.31	0.28	1.00	0.98	0.96	0.95	1.00	0.83	0.59	0.48
Activity 4	0.64	0.45	0.06	0.06	1.00	0.85	0.00	0.00	1.00	0.91	1.00	0.88	1.00	0.82	0.43	0.32
Activity 5	0.16	0.01	0.00	0.00	0.58	0.40	0.00	0.00	0.13	0.11	0.83	0.82	1.00	0.72	0.20	0.25
Activity 6	0.67	0.00	1.00	0.00	0.29	0.16	0.01	0.00	0.13	0.11	0.82	0.82	0.10	0.10	0.27	0.22

Finally, we analyze the accuracy of the provided information about deviations. We first analyze the normal alignments of event logs and process models. Thereafter, for each alignment, we determine the six most problematic activities based on their deviation ratio that is computed based on the following formula.

$$DeviationRatio = \frac{AsynchronousMoves}{AsynchronousMoves + SynchronousMoves} \tag{2}$$

Afterwards, we compare the deviation ratio of these problematic activities with the case that the approximation method was used. The result of this experiment is given in Table 5. Here, we used the *frequency* selection method with an approximation parameter equal to 10. We did not compare the result with the *statistical* method as the goal of this method is either the fitness value or the number of asynchronous moves; but, could not return both of them at the same time[4]. Results show that using the *frequency* method, we find the problematic activities that have high deviation rates.

Considering all the experiments, we conclude that using frequency of traces for selecting candidates is more practical. Moreover, the candidate selection methods give more flexibility to users to trade off between the performance and the accuracy of approximations compared to the *statistical* method that sometimes could not improve the performance and has nondeterministic results. In addition, the proposed methods provide bounds for the approximated alignment value and deviation rates for activities that is useful for many diagnostic applications. Finally, the proposed methods are able to use parallel computation and benefit from adjusted computational resources.

6 Conclusion

In this paper, we proposed approximation methods for conformance value including providing upper and lower bounds. Instead of computing the accurate alignment between the process model and all the traces available in the event log, we propose to

[4] Approximating deviations is required much more time using the *statistical* method.

just consider a subset of possible behavior in the process model and use it for approximating the conformance value using the edit distance function. We can find this subset by computing the optimal alignments of some candidate traces in the event log or by simulating the process model. To evaluate our proposed methods, we developed them in ProM framework and also imported them to RapidProM and applied them on several real event logs. Results show that these methods decrease the conformance checking time and at the same time find approximated values close to the actual alignment value. We found that the *simulation* method is suitable to be used when the given process model is simple. We also show that using the *frequency* method is more applicable to select the candidate traces and have accurate results. Results also indicate that although the *statistical* method is able to approximate accurately, it takes more time and for some event logs, it is slower than the normal conformance checking.

As future work, we aim to find out what the best subset selection method is due to the available time and event data. Also, it is possible to provide an incremental approximation tool that increases the M_B during the time and let the end user decide when the accuracy is enough. Here, we did not use the probabilities for the simulation method, we think that by using the distribution in the event log, we enhance the *simulation* method.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer Berlin Heidelberg (2016)
2. Rozinat, A., Van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1) (2008) 64–95
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. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: Online conformance checking: relating event streams to process models using prefix-alignments. International Journal of Data Science and Analytics (2017) 1–16
5. De Leoni, M., van der Aalst, W.M.: Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the 28th annual ACM symposium on applied computing, ACM (2013) 1454–1461
6. Fahland, D., van der Aalst, W.M.P.: Model Repair—Aligning Process Models to Reality. Information Systems **47** (2015) 220–243
7. Elhagaly, M., Drvoderić, K., Kippers, R.G., Bukhsh, F.A.: Evolution of compliance checking in process mining discipline. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), IEEE (2019) 1–6
8. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer (2018)
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.: 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.: Single-entry single-exit decomposed conformance checking. Information Systems **46** (2014) 102–122

12. Lee, W.L.J., Verbeek, H.M.W., Munoz-Gama, J., van der Aalst, W.M.P., Sepúlveda, M.: Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. Inf. Sci. **466** (2018) 55–91
13. 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
14. 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
15. Carmona, J., Cortadella, J.: Process mining meets abstract interpretation. In: Machine Learning and Knowledge Discovery in Databases, Springer (2010) 184–199
16. 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
17. Berti, A.: Statistical sampling in process mining discovery. In: The 9th International Conference on Information, Process, and Knowledge Management. (2017) 41–43
18. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible Heuristics Miner (FHM). In: CIDM. (2011)
19. Fani Sani, M., van Zelst, S., van der Aalst, W.M.P.: Repairing Outlier Behaviour in Event Logs. In: Business Information Systems, Springer (2018) 115–131
20. Bauer, M., van der Aa, H., Weidlich, M.: Estimating process conformance by trace sampling and result approximation. (2019) 179–197
21. 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
22. Sellers, P.H.: On the theory and computation of evolutionary distances. SIAM Journal on Applied Mathematics **26**(4) (1974) 787–793
23. Marzal, A., Vidal, E.: Computation of normalized edit distance and applications. IEEE transactions on pattern analysis and machine intelligence **15**(9) (1993) 926–932
24. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.: Measuring precision of modeled behavior. Information systems and e-Business Management **13**(1) (2015) 37–67
25. 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)
26. De Amorim, R.C., Zampieri, M.: Effective spell checking methods using clustering algorithms. In: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013. (2013) 172–178
27. van der Aalst, W.M.P., Bolt, A., van Zelst, S.: RapidProM: Mine Your Processes and Not Just Your Data. CoRR **abs/1703.03740** (2017)
28. Van Dongen, B.F. (Boudewijn): Bpi challenge 2012 (2012)
29. Van Dongen, B.F. (Boudewijn), Borchert, F. (Florian): Bpi challenge 2018 (2018)
30. Van Dongen, B.F. (Boudewijn): Bpi challenge 2019 (2019)
31. Mannhardt, F.: Hospital billing-event log. Eindhoven University of Technology. Dataset (2017) 326–347
32. De Leoni, M., Mannhardt, F.: Road traffic fine management process. Eindhoven University of Technology. Dataset (2015)
33. Mannhardt, F.: Sepsis cases-event log. eindhoven university of technology (2016)
34. 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
35. van Dongen, B.F.: 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) 44–55