History-based Construction of Log-Process Alignments for Conformance Checking: Discovering What Really Went Wrong*

Mahdi Alizadeh, Massimiliano de Leoni, and Nicola Zannone

Department of Mathematics and Computer Science,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{m.alizadeh,m.d.leoni,n.zannone}@tue.nl

Abstract. Alignments provide a robust approach for conformance checking which has been largely applied in various contexts such as auditing and performance analysis. Alignment-based conformance checking techniques pinpoint the deviations causing nonconformity based on a cost function. However, such a cost function is often manually defined on the basis of human judgment and thus error-prone, leading to alignments that do not provide the most probable explanations of nonconformity. This paper proposes an approach to automatically define the cost function based on information extracted from the past process executions. The cost function only relies on objective factors and thus enables the construction of the most probable alignments, i.e. alignments that provide the most probable explanations of nonconformity. Our approach has been implemented in ProM and assessed using both synthetic and real-life data.

Keywords: Conformance checking, alignments, cost functions

1 Introduction

Modern organizations are centered around the processes needed to deliver products and services in an efficient and effective manner. Organizations that operate at a higher process maturity level use formal/semiformal models (e.g., UML, EPC, BPMN and YAWL models) to document their processes. In some case these models are used to configure process-aware information systems (e.g., WFM or BPM systems). However, in most organizations process models are not used to enforce a particular way of working. Instead, process models are used for discussion, performance analysis (e.g., simulation), certification, process improvement, etc. However, reality may deviate from such models. People tend to focus on idealized process models that have little to do with reality. This illustrates the importance of *conformance checking* [1,2,9].

* This work has been funded by the NWO CyberSecurity programme under the PriCE project and the Dutch national program COMMIT under the THeCS project.

Conformance checking aims to verify whether the observed behavior recorded in an event log matches the intended behavior represented as a process model. The notion of alignments [2] provides a robust approach to conformance checking, which makes it possible to pinpoint the deviations causing nonconformity. An alignment between a recorded process execution and a process model is a pairwise matching between activities recorded in the log and activities allowed by the model. Sometimes, activities as recorded in the event log (events) cannot be matched to any of the activities allowed by the model (process activities). For instance, an activity is executed when not allowed. In this case, we match the event with a special *null* activity (hereafter, denoted as $\gg$), thus resulting in so-called *moves on log*. Other times, an activity should have been executed but is not observed in the event log. This results in a process activity that is matched to a $\gg$ event, thus resulting in a so-called *move on model*.

Alignments are powerful artifacts to detect nonconformity between the observed behavior as recorded in the event log and the prescribed behavior as represented by process models. In fact, when an alignment between a log trace and process model contains at least one move on log or model, it means that such a log trace does not conform the model. As a matter of fact, the moves on log/model indicate where the execution is not conforming by pinpointing the deviations that have caused this nonconformity.

In general, a large number of possible alignments exist between a process model and a log trace, since there may exist manifold explanations why a trace is not conforming. It is clear that one is interested in finding the most probable explanation. Adriansyah et al. [4] have proposed an approach based on the principle of the Occam's razor: the simplest and most parsimonious explanation is preferable. Therefore, one should not aim to find any alignment but, precisely, one of the alignments with the least expensive deviations (one of the so-called *optimal alignments*), according to some function assigning costs to deviations.

Existing alignment-based conformance checking techniques (e.g. [2,4]) require process analysts to manually define a cost function based on their background knowledge and beliefs. The definition of such a cost function is fully based on human judgment and, thus, prone to imperfections. These imperfections ultimately lead to alignments that are optimal, according to the provided cost function, but that do not provide the most probable explanation of nonconformity.

In this paper, we propose an alternative way to define a cost function, where the human judgment is put aside and only objective factors are considered. The cost function is automatically constructed by looking at the logging data and, more specifically, at the past process executions that are compliant with the process model. The intuition behind is that one should look at the past history of process executions and learn from it what is the most probable explanations of nonconformity. We believe that the most probable explanation of nonconformity of a certain process execution can be obtained by analyzing the behavior observed for such a process execution in each and every state and the behavior observed for other confirming traces when they were in the same state. Our approach gives a potentially different cost for each move on model and log (depending on the current state), leading to the definition of a more sensitive cost function.

The approach has been fully implemented as a software plug-in for the open-source process-mining framework *ProM*. To assess the practical relevance of our approach, we performed an evaluation using both synthetic and real event logs and process models. In particular, we tested it on a real-life case study about the management of road-traffic fines by an Italian town. The results show that our approach significantly improves the accuracy in determining the most probable explanation for nonconformity compared to existing techniques.

The paper is organized as follows. Section 2 introduces preliminary concepts. Section 3 presents our approach for constructing optimal alignments. Section 4 presents experiment results, which are discussed in Section 5. Finally, Section 6 discusses related work and concludes the paper with directions for future work.

2 Preliminaries

This section introduces the notation and preliminaries for our work.

2.1 Labeled Petri Nets, Event Logs, and Alignments

Process models describe how processes should be carried out. Many languages exist to model processes. Here, we use a very simple formalism, which however allow one to define all the aspects to take into account for this paper:

Definition 1 (Labeled Petri Net). *A Labeled Petri net is a tuple* $(P, T, F, A, \ell, m_i, m_f)$ *where*

- P *is a set of places;*
- T *is a set of transitions;*
- $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation between places and transitions (and between transitions and places);*
- A *is the set of labels for transitions;*
- $\ell : T \to A$ *is a function that associates a label with every transition in* T*;*
- m_i *is the initial marking;*
- m_f *is the final marking.*

The label of a transition identifies the activity represented by such a transition. Multiple transitions can be associated with the same activity label; this means that the same activity is represented by multiple transitions. This is typically done to make the model simpler. Some transitions can be invisible. Invisible transitions do not correspond to actual activities but are necessary for routing purposes and, as such, their execution is never recorded in event logs. Given a Labeled Petri net N, $\mathsf{Inv}_N \subseteq A$ indicates the set of labels associated with invisible transitions. As a matter of fact, invisible transitions are also associated with labels, though these labels do not represent activities. We assume that a label associated with a visible transition cannot be also associated with invisible ones and vice versa.

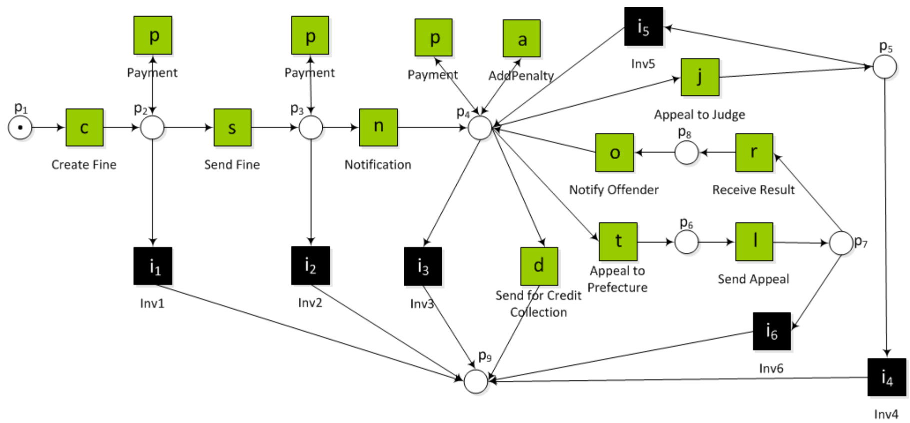


Fig. 1: A process model for managing road traffic fines. The green boxes represent the transitions that are associated with process activities while the black boxes represent invisible transitions. The text below the transitions represents the label, which is shortened with a single letter as indicated inside the transitions.

In the remainder, the simpler term Petri net is used to refer to Labeled Petri nets. The state of a Petri net is represented by a *marking*, i.e. a multiset of tokens on the places of the net. A Petri net has an initial marking m_i and a final marking m_f. When a transition is executed (i.e., *fired*), a token is taken from each of its input places and a token is added to each of its output places. A sequence of transitions σ_M leading from the initial to the final marking is a *complete process trace*. Given a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$, Γ_N indicates the set of all complete process traces.

Example 1. Fig. 1 shows a process model for handling road traffic fines in Italy [14]. A process execution starts with recording a traffic fine in the system and sending it to Italian residents. Traffic fines might be paid before or after they are sent out by police or received by the offenders. If the fine is not paid in 180 days, a penalty is added. In addition, offenders may appeal against fines to prefecture and/or judge. If an appeal is accepted, the fine management is closed. Finally, if the fine is not paid by the offender, eventually the process terminates by handing over the case for credit collection.

Given a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$, a log *trace* $\sigma_L \in A^*$ is a sequence of events where each event records the firing of a transition. In particular, each event records the label of the transition that has fired. An *event log* $\mathcal{L} \in \mathbb{B}(A)$ is a multiset of log traces, where $\mathbb{B}(X)$ is used to represent the set of all multisets over X. Here we assume that no events exist for activities not in A; in practice, this can happen: in such cases, such events are filtered out before the event log is taken into consideration.

Not all log traces can be reproduced by a Petri net, i.e. not all log traces perfectly fit the process description. If a log trace perfectly fits the net, each

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline c & s & n & t & \gg & \gg & o & \gg \\ \hline c & s & n & t & l & r & o & i_3 \\ \hline\end{array} \; \gamma_2 = \begin{array}{|c|c|c|c|c|c|c|}\hline c & s & n & t & o & \gg & \gg \\ \hline c & s & n & t & \gg & l & i_6 \\ \hline\end{array} \; \gamma_3 = \begin{array}{|c|c|c|c|c|c|}\hline c & s & n & t & o & \gg \\ \hline c & s & n & \gg & \gg & d \\ \hline\end{array}$$

Fig. 2: Alignments of $\sigma_1 = \langle c, s, n, t, o\rangle$ and the process model in Fig. 1

"move" in the log trace, i.e. an event observed in the trace, can be mimicked by a "move" in the model, i.e. a transition fired in the net. After all events in the log trace are mimicked, the net reaches its final marking. In cases where deviations occur, some moves in the log trace cannot be mimicked by the net or vice versa. We explicitly denote "no move" by $\gg$.

Definition 2 (Legal move). *Let $N = (P, T, F, A, \ell, m_i, m_f)$ be a Petri net. Let $S_L = (A \setminus \mathsf{Inv}_N) \cup \{\gg\}$ and $S_M = A \cup \{\gg\}$. A legal move is a pair $(m_L, m_M) \in (S_L \times S_M) \setminus (\gg, \gg)$ such that*

- (m_L, m_M) *is a* synchronous move *if $m_L \in S_L$, $m_M \in S_M$ and $m_L = m_M$,*
- (m_L, m_M) *is a* move on log *if $m_L \in S_L$ and $m_M = \gg$,*
- (m_L, m_M) *is a* move on model *if $m_L = \gg$ and $m_M \in S_M$.*

Σ_N denotes the set of legal moves for a Petri net N.

In the remainder, we indicate that a sequence σ' is a prefix of a sequence σ'', denoted with $\sigma' \in \mathsf{prefix}(\sigma'')$, if there exists a sequence σ''' such that $\sigma'' = \sigma' \oplus \sigma'''$, where $\oplus$ denotes the concatenation operator.

Definition 3 (Alignment). *Let Σ_N be the set of legal moves. An alignment of a log trace σ_L and a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$ is a sequence $\gamma \in \Sigma_N^*$ such that, ignoring all occurrences of $\gg$, the projection on the first element yields σ_L and the projection on the second element yields a sequence $\langle a_1, \ldots, a_n\rangle$ such that there exists a sequence $\sigma'_P = \langle t_1, \ldots, t_n\rangle \in \mathsf{prefix}(\sigma_P)$ for some $\sigma_P \in \Gamma_N$ where, for each $1 \le i \le n$, $\ell(t_i) = a_i$. If $\sigma'_P \in \Gamma_N$, γ is called a* complete alignment *of σ_L and N.*

Fig. 2 shows three possible complete alignments of a log trace $\sigma_1 = \langle c, s, n, t, o\rangle$ and the net in Fig. 1. The top row of an alignment shows the sequence of events in the log trace, and the bottom row shows the sequence of activities in the net (both ignoring $\gg$). Hereafter, we denote $|_L$ the projection of an alignment over the log trace and $|_P$ the projection over the net.

As shown in Fig. 2, there can be multiple possible alignments for a given log trace and process model. The quality of an alignment is measured based on a provided *cost function* $K : \Sigma_N^* \to \mathbb{R}_0^+$, which assigns a cost to each alignment $\gamma \in \Sigma_N^*$. Typically, the cost of an alignment is defined as the sum of the costs of the individual moves in the alignment. An **optimal alignment** of a log trace and a process trace is one of the alignments with the lowest cost according to the provided cost function.

As an example, consider a cost function that assigns to any alignment a cost equal to the number of moves on log and model for visible transitions. If moves on model for invisible transitions i_k are ignored, γ_1 has two moves on model, γ_2

has one move on model and one move on log, and γ_3 has one move on model and two moves on log. Thus, according to the cost function, γ_1 and γ_2 are two optimal alignments of σ_1 and the process model in Fig. 1.

2.2 State Representation

At any point in time, a sequence of execution of activities leads to some state, and this state depends on which activities have been performed and in which order. Accordingly, any process execution can be mapped onto a state. As discussed in [3], a *state representation function* takes care of this mapping:

Definition 4 (State Representation). *Let $(P, T, F, A, \ell, m_i, m_f)$ be a Petri net. Let R be the set of possible state representations of sequences in A^* A state representation function* $\mathsf{abst} : A^* \rightarrow R$ *produces a state representation* $\mathsf{abst}(\sigma)$ *for each process trace $\sigma \in \Gamma$.*

Several state-representation functions can be defined. Each function leads to a different abstraction, meaning that multiple different traces can be mapped onto the same state, thus abstracting out certain trace's characteristics. The following are examples of state-representation functions:

Sequence abstraction. It is a trivial mapping where the abstraction preserves the order of activities. Each trace is mapped onto a state that is the trace itself, i.e. for each $\sigma \in A^*$, $\mathsf{abst}(\sigma) = \sigma$.

Multi-set abstraction. The abstraction preserves the number of times each activity is executed. This means that, for each $\sigma \in A^*$, $\mathsf{abst}(\sigma) = M \in \mathbb{B}(A)$ such that, for each $a \in A$, M contains all instances of a in σ.

Set abstraction. The abstraction preserves whether each activity has been executed or not. This means that, for each $\sigma \in A^*$, $\mathsf{abst}(\sigma) = M \subseteq A$ such that, for each $a \in A$, M contains a if it ever occurs in σ.

Example 2. Table 1 shows the state representation of some process traces of the net in Fig. 1 using different abstractions. For instance, trace $\langle c, p, p, s, n \rangle$ can be represented as the trace itself using the sequence abstraction, as $\{c(1), p(2), s(1), n(1)\}$ using the multi-set abstraction (in parenthesis the number of occurrences of activities in the trace), and as $\{c, p, s, n\}$ using the set abstraction. Traces $\langle c, p, s, n \rangle$ and $\langle c, p, p, s, n, p \rangle$ are also mapped to state $\{c, p, s, n\}$ using the set abstraction.

3 History-based Construction of the Most Probable Alignments

This section presents our approach to construct alignments that give the most probable explanations of deviations based on objective facts, i.e. the historical logging data, rather than on subjective cost functions manually defined by process analysts. To construct an optimal alignment between a process model and an event log, we use the A-star algorithm, analogously to what proposed in [4].

Section 3.1 discusses how the cost of an alignment is computed, whereas Section 3.2 briefly reports on the use of A-star to compute the most probable alignment.

Table 1: Examples of state representation using different abstractions

Sequence	#	Multi-set	#	Set	#
$\langle c, p\rangle$	25	$\{c(1), p(1)\}$	25	$\{c, p\}$	25
$\langle c, s, n, p\rangle$	15	$\{c(1), p(1), s(1), n(1)\}$	15	$\{c, p, s, n\}$	45
$\langle c, p, p, s, n\rangle$	5	$\{c(1), p(2), s(1), n(1)\}$	5		
$\langle c, p, p, s, n, p\rangle$	25	$\{c(1), p(3), s(1), n(1)\}$	25		
$\langle c, s, n, a, d\rangle$	10	$\{c(1), s(1), n(1), a(1), d(1)\}$	10	$\{c, s, n, a, d\}$	10
$\langle c, s, n, p, a, d\rangle$	10	$\{c(1), s(1), n(1), p(1), a(1), d(1)\}$	10	$\{c, s, n, p, a, d\}$	10
$\langle c, s, n, p, t, l\rangle$	25	$\{c(1), s(1), n(1), p(1), t(1), l(1)\}$	30	$\{c, s, n, p, t, l\}$	60
$\langle c, s, p, n, t, l\rangle$	5				
$\langle c, p, s, n, p, t, l\rangle$	5	$\{c(1), s(1), n(1), p(2), t(1), l(1)\}$	30		
$\langle c, s, p, n, p, t, l\rangle$	25				
$\langle c, s, n, p, t, l, r, o\rangle$	50	$\{c(1), s(1), n(1), p(1), t(1), l(1), r(1), o(1)\}$	50	$\{c, s, n, p, t, l, r, o\}$	50

3.1 Definition of cost functions

The computation of the most probable alignment relies on a cost function that accounts for the probability of an activity to be executed in a certain state. The definition of such a cost function requires an analysis of the past history as recorded in the event log to compute the probability of an activity to immediately occur or to never eventually occur when the process execution is in a certain state.

The cost of moves depends on probabilities. For this purpose, we need to introduce a functions' class $\mathcal{F} \subseteq [0,1] \to \mathbb{R}^+$ such that $f \in \mathcal{F}$ if and only if $f(0) = \infty$ and f is monotonously decreasing between 0 and 1 (with $f(1) > 0$). Hereafter, these functions are called *cost profile*. It is easy to observe that if $f(p)$ is a cost-profile function, then $f(p)^i$ is also a cost-profile function for every $i > 0$. Examples of these functions are:

$$f(p) = \tfrac{1}{p} \qquad f(p) = \tfrac{1}{\sqrt{p}} \qquad f(p) = 1 + \log\left(\tfrac{1}{p}\right) \tag{1}$$

Similarly to what proposed in [4], the cost of an alignment move depends on the move type and the activity involved in the move but, differently from [4], it also depends on the position in which the move is inserted:

Definition 5 (Cost of an alignment move). *Let $N = (P, T, F, A, \ell, m_i, m_f)$ be an Petri net. Let $\gamma \in \Sigma_N^*$ be a sequence of legal moves for N and $f \in \mathcal{F}$ a cost profile. The cost of appending a legal move $(m_L, m_M) \in \Sigma_N$ to γ with state-representation function* abst *is:*

$$\kappa_{\mathsf{abst}}((m_L, m_M), \gamma) = \begin{cases} 0 & m_L = m_M \\ 0 & m_L = \gg \ \text{and}\ m_M \in \mathsf{Inv}_N \\ f\big(P_{\mathsf{abst}}(m_M \text{ occurs after } \gamma|_P)\big) & m_L = \gg \ \text{and}\ m_M \notin \mathsf{Inv}_N \\ f\big(P_{\mathsf{abst}}(m_L \text{ never eventually occurs after } \gamma|_P)\big) & m_M = \gg \end{cases} \tag{2}$$

Readers can observe that the cost of a move on log $(m_L, \gg)$ is not simply the probability of not executing activity m_L immediately after $\gamma\,|_P$; rather, it is the probability of never having activity m_M at the any moment in the future for that execution. This is motivated by the fact that a move on log $(m_L, \gg)$ indicates that m_L is not expected to ever occur in the future. Conversely, if it was expected, a number of moves in model would be introduced until the process model, modeled as a Petri net, reaches a marking that allows m_L to occur (and, thus, a move in both can be appended). Different cost profiles account for the probabilities computed from historical logging data differently. In Section 4, we evaluate the cost profiles in Eq. 1 with different combinations of event logs and process models. The purpose is to verify whether a cost profile universally works better than the others. The following two definitions describe how to compute the probabilities required by Def. 5. For reliability, we only consider the subset of traces $\mathcal{L}_{fit}$ of the original event log $\mathcal{L}$ that comply with the process model.

Definition 6 (Probability that an activity occurs). *Let $\mathcal{L}$ be an event log and $\mathcal{L}_{fit} \subseteq \mathcal{L}$ be the subset of traces that comply with a given process model represented by a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$. The probability that an activity $a \in A$ occurs after executing σ with state-representation function* abst *is the ratio between number of traces in $\mathcal{L}_{fit}$ in which a is executed after reaching state* $\mathsf{abst}(\sigma)$ *and the total number of traces in $\mathcal{L}_{fit}$ that reach state* $\mathsf{abst}(\sigma)$*:*

$$P_{\mathsf{abst}}(a \text{ occurs after } \sigma) = \frac{|\{\sigma' \in \mathcal{L}_{fit} : \exists \sigma'' \in \mathsf{prefix}(\sigma').\ \mathsf{abst}(\sigma'') = \mathsf{abst}(\sigma) \wedge \sigma'' \oplus \langle a \rangle \in \mathsf{prefix}(\sigma')\}|}{|\{\sigma' \in \mathcal{L}_{fit} : \exists \sigma'' \in \mathsf{prefix}(\sigma').\ \mathsf{abst}(\sigma'') = \mathsf{abst}(\sigma)\}|} \tag{3}$$

Definition 7 (Probability that an activity never eventually occurs). *Let $\mathcal{L}$ be an event log and $\mathcal{L}_{fit} \subseteq \mathcal{L}$ be the subset of traces that comply with a given process model represented by a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$. The probability that an activity $a \in A$ will never eventually occur in a process execution after executing $\sigma \in A^*$ with state-representation function* abst *is the ratio between the number of traces in $\mathcal{L}_{fit}$ in which a is never eventually executed after reaching state* $\mathsf{abst}(\sigma)$ *and the total number of trace in $\mathcal{L}_{fit}$ that reach state* $\mathsf{abst}(\sigma)$*:*

$$P_{\mathsf{abst}}(a \text{ never eventually occurs after } \sigma) = \frac{|\{\sigma' \in \mathcal{L}_{fit} : \exists \sigma'' \in \mathsf{prefix}(\sigma').\ \mathsf{abst}(\sigma'') = \mathsf{abst}(\sigma) \wedge \forall \sigma'''\ \sigma'' \oplus \sigma''' \oplus \langle a' \rangle \in \mathsf{prefix}(\sigma') \wedge a' \neq a\}|}{|\{\sigma' \in \mathcal{L}_{fit} : \exists \sigma'' \in \mathsf{prefix}(\sigma').\ \mathsf{abst}(\sigma'') = \mathsf{abst}(\sigma)\}|} \tag{4}$$

The cost of an alignment is the sum of the cost of all moves in the alignment, which are computed as described in Definition 5:

Definition 8 (Cost of an alignment). *The cost of alignment $\gamma \in \Sigma_N^*$ with state-representation function* abst *is computed as follows:*

$$K_{\mathsf{abst}}(\gamma \oplus (m_L, m_M)) = \begin{cases} \kappa_{\mathsf{abst}}((m_L, m_M), \langle\rangle) & \gamma = \langle\rangle \\ \kappa_{\mathsf{abst}}((m_L, m_M), \gamma) + K_{\mathsf{abst}}(\gamma) & \text{otherwise} \end{cases} \tag{5}$$

Hereafter, the term **most-probable alignment** is used to denote any of the optimal alignments (i.e., with the lowest cost) according to the cost function given in Definition 8.

3.2 The use of the A-star algorithm to construct alignments

The A-star algorithm [10] aims to find a path in a graph V from a given *source* node v_0 to any node $v \in V$ in a target set. Every node v of graph V is associated with a cost determined by an *evaluation* function $f(v) = g(v) + h(v)$, where

- $g : V \to \mathbb{R}_0^+$ is a function that returns the smallest path cost from v_0 to v;
- $h : V \to \mathbb{R}_0^+$ is an heuristic function that estimates the path cost from v to its preferred target node.

Function h is said to be *admissible* if it returns a value that underestimates the distance of a path from a node v' to its preferred target node v'', i.e. $g(v') + h(v') \leq g(v'')$. If h is admissible, A-star finds a path that is guaranteed to have the overall lowest cost.

The A-star algorithm keeps a priority queue of nodes to be visited: higher priority is given to nodes with lower costs. The algorithm works iteratively: at each step, the node v with lowest cost is taken from the priority queue. If v belongs to the target set, the algorithm ends returning node v. Otherwise, v is expanded: every successors v' is added to priority queue with a cost $f(v')$.

We employ A-star to find any of the optimal alignments between a log trace $\sigma_L \in \mathcal{L}$ and a Petri net N. In order to be able to apply A-star, an opportune search space needs to be defined. Every node γ of the search space V is associated to a different alignment that is a prefix of some complete alignment of σ_L and N. Since a different alignment is also associated to every search-space node and vice versa, we use the alignment to refer to the associated state. The source node is an empty alignment $\gamma_0 = \langle\rangle$ and the set of target nodes includes every complete alignment of σ_L and N.

Let us denote the length of a sequence σ with $\|\sigma\|$. Given a node/alignment $\gamma \in V$, the search-space successors of γ include all alignments $\gamma' \in V$ obtained from γ by concatenating exactly one move. Given an alignment $\gamma \in V$, the cost of path from the initial node to node $\gamma \in V$ is:

$$g(\gamma) = \|\gamma_{|L}\| + K(\gamma).$$

where $K(\gamma)$ is the cost of alignment γ according to Definition 8. It is easy to check that, given two complete alignments γ'_C and γ''_C, $K(\gamma'_C) < K(\gamma''_C)$ iff $g(\gamma'_C) < g(\gamma''_C)$ and $K(\gamma'_C) = K(\gamma''_C)$ iff $g(\gamma'_C) = g(\gamma''_C)$. Therefore, an optimal solution returned by A-star coincides with an optimal alignment. To define a more efficient and admissible heuristics, we consider term $\|\sigma_L\|$ in h; this term does not affect optimality. Given an alignment $\gamma \in V$, we employ the heuristics:

$$h(\gamma) = \|\sigma_L\| - \|\gamma_{|L}\|.$$

For alignment γ, the number of steps to add in order to reach a complete alignment is lower bounded by the number of execution steps of trace σ_L that have not been included yet in the alignment, i.e. $\|\sigma_L\| - \|\gamma_{|L}\|$. Since the additional cost to traverse a single node is at least 1, the cost to reach a target node is at least $h(\gamma)$, corresponding to the case where the part of the log trace that still needs to be included in the alignment perfectly fits.

$$\gamma' = \underbrace{\begin{array}{|c|c|c|} c & s & n \\ \hline c & s & n \end{array}}_{\gamma} \quad \oplus \quad \begin{cases} (l, \gg) & \kappa\big((l, \gg), \gamma\big) = 1.49 \\ (\gg, p) & \kappa\big((\gg, p), \gamma\big) = 1.04 \\ (\gg, a) & \kappa\big((\gg, a), \gamma\big) = 2.04 \\ (\gg, d) & \kappa\big((\gg, d), \gamma\big) = \infty \\ \quad \dots & \end{cases}$$

Fig. 3: Construction of the alignment of log trace $\sigma_2 = \langle c, s, n, l, o \rangle$ and the net in Fig. 1. Cost of moves are computed with sequence state-representation function, cost profile $f(p) = 1 + \log\left({}^{1}\!/_{p}\right)$, and $\mathcal{L}_{\mathit{fit}}$ in Table 1.

Example 3. Consider a log trace $\sigma_2 = \langle c, s, n, l, o \rangle$ and the net N in Fig. 1. An analyst wants to determine the most probable explanations for nonconformity by constructing the most probable alignment of σ_2 and N, based on historical logging data. In particular, $\mathcal{L}_{\mathit{fit}}$ consists of the traces in Table 1 (the first column shows the traces, and the second the number of occurrences of a trace in the history). Assume that the algorithm has constructed an optimal alignment γ of trace $\langle c, s, n \rangle \in \mathsf{prefix}(\sigma_2)$ and N (left part of Fig. 3). The next event in the log trace (i.e., l) cannot be replayed in the net. Therefore, the algorithm should determine which move is the most likely to have occurred. Different moves are possible; for instance, a move on log for l, a move on model for p, a move on model for t, etc. The algorithm computes the cost for these moves using Eq. 5 (right part of Fig. 3). As move on model $(\gg, p)$ is the move with the least cost (and no other alignments have lower cost), alignment $\gamma' = \gamma \oplus (\gg, p)$ is selected for the next iteration. It is worth noting that activity d never occurs after $\langle c, s, n \rangle$ in $\mathcal{L}_{\mathit{fit}}$; consequently, the cost of move $(\gg, d)$ is equal to ∞.

4 Implementation and Experiments

We have implemented our approach for history-based construction of alignments as a plugin of the open-source ProM framework (`http://www.promtools.org`). The plug-in takes as inputs a process model and two event logs. It computes the most probable alignments for each trace in the first event log based on the frequency of the traces in the second event log (historical logging data).

To assess the practical feasibility and accuracy of the approach, we performed a number of experiments using both synthetic and real-life logs. In the experiments with synthetic logs, we assumed that the execution of an activity depends on the activities that were performed in the past. In the experiments with real-life logs, we tested if this assumption holds in real applications. Accordingly, the real-life logs were used as historical logging data. To evaluate the approach, we artificially added noise to the traces used for the experiments. This was necessary to assess the ability of the approach to reconstruct the original traces.

4.1 Synthetic Data

For the experiments with synthetic data, we used the process for handling credit requests in [14]. Based on this model, we generated 10000 traces consisting of

Table 2: Results of experiments on synthetic data. CA indicates the percentage of correct alignments, and LD indicates the overall Levenshtein distance between the original traces and the projection of the alignments over the process. For comparison with existing approaches, the standard cost function as defined in [4] was used. In bold the best result is highlighted for each amount of noise.

	1/p						$1/\sqrt{p}$						1 + log(1/p)						Existing approach	
	Seq		Multi-set		Set		Seq		Multi-set		Set		Seq		Multi-set		Set			
Noise	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD
10%	93	259	93	258	87	514	**95**	164	**95**	164	88	430	**95**	**153**	**95**	154	88	409	92	233
20%	85	569	85	561	78	968	**87**	426	**87**	431	79	852	**87**	**410**	**87**	415	79	823	83	534
30%	74	1084	74	1077	65	1653	**76**	950	75	963	66	1509	**76**	**944**	75	958	67	1474	71	1110
40%	63	1658	62	1659	55	2285	**64**	1519	**64**	1537	56	2148	**64**	**1512**	**64**	1535	56	2118	60	1685

69504 events using the CPN Tools (`http://cpntools.org`). To assess the accuracy of the approach, we manipulated 20% of these traces by introducing different percentages of noise. In particular, given a trace, we added and removed a number of activities to/from the trace equal to the same percentage of the trace length. The other traces were used as historical logging data. We computed the most probable alignments of the manipulated traces and process model, and evaluated the ability of the approach to reconstruct the original traces. To this end, we measured the percentage of correct alignments (i.e., the cases where a projection of an alignment over the process coincides with the original trace) and compute the overall Levenshtein distance [12] between the original traces and the projection of the computed alignments over the process. This string metric measures the distance between two sequences, i.e. the minimal number of changes required to transform one sequence into the other. In our setting, it provides an indication of how much the projection of the computed alignments over the process is close to the original traces.

We tested our approach with different amounts of noise (i.e., 10%, 20%, 30% and 40% of the trace length), with different cost profiles (i.e., $^1/_p$, $^1/_{\sqrt{p}}$, and $1 + log(^1/_p)$), and with different state-representation functions (i.e., *sequence*, *multi-set*, and *set*). Moreover, we compared our approach with existing alignment-based conformance checking techniques. In particular, we used the standard cost function introduced in [4]. We repeated each experiment five times. Table 2 shows the results where every entry reports the average over the five runs.

The results show that cost profiles $^1/_{\sqrt{p}}$ and $1 + log(^1/_p)$ in combination with *sequence* and *multi-set* abstractions are able to better identify what really happened, i.e. they align the manipulated traces with the corresponding original traces in more cases (CA). In all cases, cost profile $1 + log(^1/_p)$ with *sequence* state-representation function provides more accurate diagnostics (LD): even if log traces are not aligned to the original traces, the projection over the process of alignments constructed using this cost profile and abstraction are closer to the original traces. Compared to the cost function used in [4], our approach computed the correct alignment for 4.4% more traces when cost profile $1 + log(^1/_p)$ and *sequence* state-representation function are used. In particular, our approach correctly reconstructed the original trace for 18.4% of the traces that were not

Table 3: Results of experiments on real-life data. Notation analogous to Table 2.

	1/p						$1/\sqrt{p}$						1 + log(1/p)						Existing approach	
	Seq		Multi-set		Set		Seq		Multi-set		Set		Seq		Multi-set		Set			
Noise	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD	CA	LD
10%	**99**	397	**99**	397	**99**	415	**99**	384	**99**	389	**99**	408	**99**	**366**	**99**	371	**99**	389	98	1274
20%	**99**	585	**99**	585	**99**	602	**99**	570	**99**	575	**99**	592	**99**	**554**	**99**	559	**99**	576	97	1448
30%	**89**	3349	**89**	3349	**89**	3371	**89**	3300	**89**	3341	**89**	3362	**89**	**3281**	**89**	3322	**89**	3344	87	4284
40%	**76**	9160	**76**	9160	75	9238	**76**	**9091**	**76**	9152	75	9230	**76**	9103	75	9165	75	9243	74	9861

correctly reconstructed using the cost function used in [4]. Moreover, an analysis of LD shows that, on average, the traces reconstructed using our approach have 0.37 deviations, while the traces reconstructed using the cost function used in [4] have 0.45 deviation. This corresponds to an improvement of LD of about 15.2%.

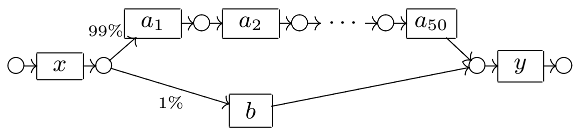
4.2 Real-life Logs

To evaluate the applicability of our approach to real-life scenarios, we used an event log obtained from a fine management system of the Italian police [14]. The process model in form of Petri net is presented in Fig. 1. We extracted a log consisting of 142408 traces and 527549 events, where all traces are conforming with the net. To these traces, we applied the same methodology used for the experiments reported in Section 4.1. We repeated the experiments five times. Table 3 shows the results where every entry reports the average over five runs.

The results confirm that cost profiles $1/\sqrt{p}$ and $1+log(1/p)$ in combination with *sequence* and *multi-set* state-representation functions provide the more accurate diagnostics (both CA and LD). Moreover, the results show that our approach (regardless of the used cost profile and state-representation function) performs better than the cost function in [4] on real-life logs. In particular, using *sequence* state-representation function and cost profile $1 + log(1/p)$, our approaches computed the correct alignment for 1.8% more traces than what the cost function in [4] did. In particular, our approach correctly reconstructed the original trace for 19.3% of the traces that were not correctly reconstructed using the cost function used in [4]. Moreover, our approach improves LD by 21.1% compared to the cost function used in [4]. Such an improvement shows that when the original trace is not reconstructed correctly, our approach returns an explanation that is significantly closer to the actual explanation.

5 Discussion

The A-star algorithm requires a cost function to penalize nonconformity. In our experiments, we have considered a number of cost profiles to compute the cost of moves on log/model based on the probability of a given activity to occur in historical logging data. The selection of the cost profile has a significant impact on the results as they penalize deviations differently. For instance, cost profile $1/p$ penalizes less probable moves much more than $1 + log(1/p)$. To illustrate



(a) Process model

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|} x & \gg & \gg & \gg & \gg & y \\ \hline x & a_1 & a_2 & \ldots & a_{50} & y \end{array}$$

$$\gamma_2 = \begin{array}{|c|c|c|} x & \gg & y \\ \hline x & b & y \end{array}$$

(b) Alignments

Fig. 4: Process model including two paths formed by a (sub)sequence of 50 activities and 1 activity respectively. The first path is executed in 99% of the cases; the second in 1% of the cases. γ_1 and γ_2 are two possible alignments of trace $\sigma = \langle x, y \rangle$ and the process model.

this, consider a trace $\sigma = \langle x, y \rangle$ and the process model in Fig. 4a. Two possible alignments, namely γ_1 and γ_2, are conceivable (Fig. 4b). γ_1 contains a large number of deviations compared to γ_2 (50 moves on log vs. 1 move on log). The use of cost profile $1/p$ yields γ_1 as the most probable alignment, while the use of cost profile $1 + log(1/p)$ yields γ_2 as the most probable alignment. Tables 2 and 3 show that cost profile $1 + log(1/p)$ usually provides more accurate results. Cost profile $1/p$ penalizes less probable moves excessively, and thus tends to construct alignments with more frequent traces in the historical logging data even if those alignments contain a significantly larger number of deviations. Our experiments suggest that the construction of the most probable alignments requires a trade-off between the frequency of the traces in historical logging data and the number of deviations in alignments, which is better captured by cost profile $1 + log(1/p)$.

Different state-representation functions can be used to characterize the state of a process execution. In this work, we have considered three state-representation functions: *sequence*, *multi-set*, and *set*. The experiments show that in general the sequence abstraction produces more accurate results compared to the other abstractions. The set abstraction provides the least accurate results, especially when applied to the process for handling credit requests (Table 2). The main reason is that this abstraction is not able to accurately characterize the state, especially in presence of loops: after each loop iteration the process execution yields the same state. Therefore, the cost function constructed using the set abstraction is not able to account for the fact that the probability of executing certain activities can increase after every loop iteration, thus leading to alignments in which loops are not captured properly.

To conclude, the experiments show that our technique tends to build alignments that give better explanations of deviations. It is easy to see that, when nonconformity is injected in fitting traces and alignments are subsequently built, the resulting alignments yield perfect explanations if the respective process projections coincide with the respective fitting traces before the injections of nonconformity. Tables 2 and 3 have shown that, basing the construction of the cost function on the analysis of historical logging data our technique tends to build alignments whose process projection is closer to the original fitting traces and, hence, the explanations of deviations are closer to the correct ones.

6 Related Work and Conclusions

In process mining, a number of approaches have been proposed to check conformance of process models and the actual behavior recorded in event logs. Some approaches [7,8,13,15,16] check conformance by verifying whether traces satisfies rules encoding properties expected from the process. Petkovič et al. [17] verify whether a log trace is a valid trace of the transition system generated by the process model. Token-based approaches [6,18] use the number of missing and added tokens obtained by replaying traces over the process model to measure the conformance between the log and the process. However, these approaches only give a boolean answers diagnosing whether traces conform to a process model or not. When they are able to provide diagnostic information, such information is often imprecise. For instance, token-based approaches may allow behavior that is not allowed by the model due to the used heuristics and thus may provide incorrect diagnostic information.

Recently, the construction of alignments has been proposed as a robust approach for checking the conformance of event logs with a given process model [4]. Alignments have proven to be very powerful artifacts to perform conformance checking. By constructing alignments, analysts can be provided with richer and more accurate diagnostic information. In fact, alignments are also used as the main enablers for a number of techniques for process analytics, auditing, and process improvement, such as for performance analysis [2], privacy compliance [5] and process-model repairing [11].

To our knowledge, the main problem of existing techniques for constructing optimal alignments is related to the fact that process analysts need to provide a function which associates a cost to every possible deviation. These cost functions are only based on human judgment and, hence, prone to imperfections. If these techniques are fed with imprecise cost functions, they create imperfect alignments, which ultimately leads to unlikely or, even, incorrect diagnostics.

In this paper, we have proposed a different approach where the cost function is automatically computed based on real facts: historical logging data recorded in event logs. In particular, the cost function is computed based on the probability of activities to be executed or not in a certain state (representing which activities have been executed and their order). Experiments have shown that, indeed, our approach can provide more probable explanations of nonconformity of process executions, if compared with existing techniques.

We acknowledge that the evaluation is far from being completed. We aim to perform more extensive experiments to verify whether certain cost-profile functions provide more probable alignments than others or, at least, to give some guidelines to determine in which settings a given cost-profile function is preferable. We also aim to develop a technique that, given a model and log, allow for (semi-)automatic tuning of the cost-profile function and state abstraction.

In this paper, we only considered the control-flow, i.e. the name of the activities and their ordering, to construct the cost function and, hence, to compute the most probable alignment. However, the choice in a process execution is often driven by other aspects. For instance, when instances are running late, the execution of certain fast activities are more probable; or, if a certain process attribute takes on a given value, certain activities are more likely to be executed. We expect that our approach can be significantly improved if the other business process perspectives (i.e., data, time and resources) are taken into account.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Data Min. Knowl. Discov. 2(2), 182–192 (2012)
3. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems 36(2), 450–475 (2011)
4. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Memory-efficient alignment of observed and modeled behavior. BPM Center Report 03-03, BPMcenter.org (2013)
5. Adriansyah, A., van Dongen, B.F., Zannone, N.: Privacy analysis of user behavior using alignments. it–Information Technology 55(6), 255–260 (2013)
6. Banescu, S., Petkovic, M., Zannone, N.: Measuring privacy compliance using fitness metrics. In: Business Process Management. pp. 114–119. LNCS 7481, Springer (2012)
7. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst. Appl. 41(11), 5340–5352 (2014)
8. Caron, F., Vanthienen, J., Baesens, B.: Comprehensive rule-based compliance checking and risk management with process mining. Decision Support Systems 54(3), 1357–1369 (2013)
9. Cook, J.E., Wolf, A.L.: Software process validation: quantitatively measuring the correspondence of a process to a model. TOSEM 8(2), 147–176 (1999)
10. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A*. Journal of the ACM 32, 505–536 (1985)
11. Fahland, D., van der Aalst, W.M.P.: Model repair - aligning process models to reality. Information Systems (2014)
12. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
13. Ly, L.T., Rinderle-Ma, S., Göser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems - requirements, challenges, solutions. Information Systems Frontiers 14(2), 195–219 (2012)
14. Mannhardt, F., de Leoni, M., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. BPM Center Report 14-08, BPMcenter.org (2014)
15. de Medeiros, A.K.A., van der Aalst. Wil M. P., Pedrinaci, C.: Semantic Process Mining Tools: Core Building Blocks. In: Proc. ECIS. pp. 1953–1964. AIS (2008)
16. Montali, M.: Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach. LNBIP 56, Springer (2010)
17. Petković, M., Prandi, D., Zannone, N.: Purpose control: Did you process the data for the intended purpose? In: Secure Data Management. pp. 145–168. LNCS 6933, Springer (2011)
18. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)