Using automatic planning to find the most probable alignment: A history-based approach

Matheus P. Almeida[1], **Karina V. Delgado**[2], **Sarajane M. Peres**[2], **Marcelo Fantinato**[2]

[1]Institute of Mathematics and Statistics – University of São Paulo
São Paulo – SP – Brazil

[2]School of Arts, Sciences and Humanities – University of São Paulo
Ermelino Matarazzo – SP – Brazil.

{matheus.pereira.almeida, kvd, sarajane, m.fantinato}@usp.br

Abstract. *In many organizational contexts, the existence of a normative process model makes it possible to verify if the actual execution of activities of a business process conforms to that model. Non-conforming behavior can be detected by aligning the actions recorded in the event log with a related normative process model. The alignment approach uses a cost-function to build an execution path that shows which actions do not conform to the model, and which are the expected activities for that trace. In this paper, we are interested on finding the optimal probable alignment using a history-base cost-function, i.e. a function based on the process execution history. For that, we use as a base implementation the planning-based approach, previously proposed in the literature, which demonstrates to find large processes faster when compared to A* for standard cost-functions. We incorporate in this tool the automatic generation of an history-base cost-function to find an optimal probable alignment. In addition, we evaluated our approach using data from both synthetic event logs and a real-life event log.*

1. Introduction

Process mining uses a set of techniques for discovering, observing, and improving business processes by learning from event logs [van der Aalst W., 2016]. These techniques can provide insights about business processes based on facts recorded in event logs. The pillars of process mining are process discovery, conformance checking, and enhancement [van der Aalst W., 2016]. Conformance checking uses as input an event log and a business process model; based on this information, it compares how well the event log conforms to the process model and vice versa. This task is used to check exactly where an actual execution differs from the process model. Some methods were proposed in the literature to solve the conformance checking task, for instance, the alignment method [van der Aalst W., 2016].

The idea of alignment offers a reliable method for conformance checking, making it feasible to identify the deviations that results in nonconformity. A pairwise match between actions recorded in the event log and actions permitted by the process model indicates that a recorded process execution and a process model are in alignment. Sometimes the activities recorded in the event log may match none of the activities prescribed by the process model. In such cases, the alignment algorithm needs to execute moves to

continue the conformance checking. In an alignment algorithm, given a certain state of a process instance, there are three types of move: a *synchronous* move occurs when the execution of the activity recorded in the event log corresponds to an allowed transition in the process model; a *model* move occurs when there is no log move; and, finally, a *log* move occurs when there is no model move.

In general, there is a large number of possible alignments between a process model and a specific trace in the event log, as there may be several explanations why a trace is not conforming. In this case, the objective is to find an alignment that has least expensive deviation, that is called **optimal alignment**. For this, a standard cost function is usually used, which assigns unit cost to the deviations. However, if we intend to find the most likely explanation, we must define a different cost function. One option is to use non-fixed costs that may vary depending on the history of previously observed process executions. The most likely alignment using this kind of history-base cost-function is called **optimal probable alignment** [Alizadeh et al., 2015].

According to Dunzer et al. [2019], among the approaches to solve the alignment task are ad-hoc implementations of the A* algorithm to compute optimal alignments [Adriansyah et al., 2013; Alizadeh et al., 2015; Koorneef et al., 2018; Boltenhagen et al., 2021; Bloemen et al., 2022] and an approach that uses PDDL [Aeronautiques et al., 1998] to formulate the alignment as a planning problem [Leoni et al., 2017; Leoni and Marrella, 2017].

The approaches that uses an standard-cost function are [Leoni et al., 2017; Leoni and Marrella, 2017; Boltenhagen et al., 2021] and the approaches that uses a history-base cost-function are [Koorneef et al., 2018; Alizadeh et al., 2015]. Bloemen et al. [2022] and Adriansyah et al. [2013] use a variant of the standard cost-function whose goal is to improve time and memory consumption, respectively.

Leoni and Marrella [2017] showed there is an improvement in terms of processing time for large processes applying the planning-based approach when compared to A*. For this reason, we propose modifications on top of the planning-based approach of Leoni and Marrella. Our proposal is to adopt the automatic generation of the historical base cost function, originally proposed by Koorneef et al. [2018], but slightly modified in our proposal to allow finding an optimal probable alignment. Furthermore, the proposal by Koorneef et al. [2018] has not been empirically evaluated before. In our work, we extended the Leoni and Marrella tool to implement our ideas and performed experiments with synthetic and real-life event log to empirically validate the effectiveness of our proposal.

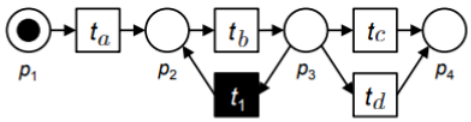
The rest of the paper is organized as follows: Section 2 presents the background; Section 3 reports the technique proposed by Leoni and Marrella [2017] to convert the alignment task in planning problem; Section 4 describes the approach proposed by Alizadeh et al. [2015] and Koorneef et al. [2018] to generate costs using the history; Section 5 describes our approach, which combines both previously mentioned approaches, as well as our contributions to them. Section 6 presents the experiments performed on both real-life and synthetic event logs. Finally, Section 7 concludes the paper.

2. Background

A process model is represented as a labelled Petri net. A labelled Petri net is a directed graph with two types of nodes: place and transition. These nodes are graphically represented by circles and rectangles, respectively. Nodes are connected by arcs and connections between nodes of the same type are not allowed.

Definition 1 (Labelled Petri net [Valk and Vidal-Naquet, 1981]) *A labelled Petri net is a tuple $\langle P, T, F, A, \ell, m_i, m_f \rangle$ that contains: A finite set of P places; a finite set of T transitions such that $T \cap P = \varnothing$; a flow relation $F \subseteq (P \times T) \cup (T \times P)$; a set of activity labels A; a function $\ell : T \rightarrow A$ that associates a label with some transitions in T; an initial marking m_i and a final marking m_f.*

Transitions $Inv \subset T$ that are associated with no labels are called invisible transitions. Figure 1 shows an example of a labelled Petri net and the respective event log used as example in this paper. This labelled Petri net has five transitions and one of them is an invisible transition (t_1), four places and a token on $p1$.



Trace	Frequency
$\langle a, b, c \rangle$	40
$\langle a, b, d \rangle$	10
$\langle a, b, b, c \rangle$	80
$\langle a, b, b, d \rangle$	20
$\langle a, b, b, b, c \rangle$	40
$\langle a, b, b, b, d \rangle$	10
$\langle a, b, b, b, b, b, b, c \rangle$	1

Figure 1. Example of a labelled Petri net and the respective event log, adapted from Koorneef et al. [2018].

We assume an event log contains data related to only one process model. An event log is a multiset of traces and a trace is a sequence of activities. Furthermore, the occurrence of an activity is related to an event that has attributes as ID, timestamp, costs and resources [van der Aalst W., 2016]. For example, the trace $\langle a, b, c \rangle$ appears 40 times in the event log of Figure 1, where a, b and c are activities.

A *noisy trace* is a trace that does not fit into the process model for some reason and it usually has an infrequent occurrence in the event log. Such a non-fitted trace is not expected according to a given process model. Therefore, either the assumed process model needs be adapted to represent that trace, or there was a problem while executing the process, which is reflect in the event log. For example, $\langle a, b \rangle$ is a noisy trace.

Definition 2 (Legal moves [van der Aalst W., 2016]) *Let $N = \langle P, T, F, A, \ell, m_i, m_f \rangle$ be a labelled Petri net and L be an event log. A legal alignment move is represented by a pair (s_L, s_M), where s_L represents an activity in the event log and s_M in the process model. $\gg$ denotes that no move happened. The possible legal moves are:*

- *a log move if $s_L \neq \gg$ and $s_M = \gg$,*
- *a model move if $s_L = \gg$ and $s_M \in T$,*
- *a synchronous move if $s_L \neq \gg$ and $s_M \in T$ for the same activity,*

Definition 3 (Alignment [van der Aalst W., 2016]) *Let $N = \langle P, T, F, A, \ell, m_i, m_f \rangle$ be a labelled Petri net and L be an event log. Let Y be the universe of all legal alignment*

moves for N and L. Let $\sigma \in L$ be a trace. Sequence $y \in Y$ is an alignment of N and σ if, ignoring all occurrences of $\gg$, the projection on the first element results in σ and the projection on the second results in a sequence σ that can be replayed in the labelled Petri net.

Figure 2 shows two possible alignments for the Petri net and the event log given in Figure 1. The first row represents the trace $\langle a, b \rangle$ that is being aligned, and the second row represents the movements in the labelled Petri net. In the first alignment y_1, there were two synchronous moves and one model move where $s_L = \gg$ and $s_M = d$. In the second alignment y_2, there were also two synchronous moves and one model move where $s_L = \gg$ and $s_M = c$.

$y_1 =$

a	b	$\gg$
a	b	d

$y_2 =$

a	b	$\gg$
a	b	c

Figure 2. Example of alignments.

Considering the example in Figure 2, identifying which alignment is optimal depends on the cost function in use. The standard cost-function assigns a cost 1 to log moves and model moves (except for invisible transitions) and 0 to synchronous moves. In this case, both alignments are optimal. In the example shown in Table 2, we raise a second issue related to which of the two alignments should be considered the **most likely explanation**. We should adjust a way of defining costs in order to display the most likely explanation for the alignment. For that, we consider the most likely explanation must be related to what has already been observed in the event log and that conforms to the process model. This alignment is referred as **probable alignment** [Alizadeh et al., 2015]. In order to find this type of alignments, we use the cost function of Koorneef et al. [2018] that takes into account an event log to assign probabilities to activities (cf. Section 4.1).

3. Encoding the Alignment Task as a Planning Problem

Leoni and Marrella [2017] encode the alignment task as an observable, static and deterministic planning problem. For each trace to be aligned, a problem and domain description files are created using the PDDL language [Aeronautiques et al., 1998].

3.1. Problem description

In the problem file, there are three types of objects: place, transition and event. These types represent the places, the transitions of the labelled Petri net and finally, the type event is used to record exactly in which activity we are while aligning the trace.

Additionally, two boolean predicates were defined: *token* and *tracePointer*.: *token* holds iff the current place contains a token; *tracePointer* keeps track of the current activity that is being aligned, and holds iff the activity received as a parameter is the next to be aligned. The final objective is to minimize the total cost of the alignment.

For example, for trace $\sigma_L = \langle a, b, b, c \rangle$ and the labelled Petri net in Figure 1, the objects are:

```
{: objects a b c d inv - transition
           p1 p2 p3 p4 - place
           ev1 ev2 ev3 ev4 evEND - event
}.
```

In this example, five objects of type event were defined because the trace σ_L has four elements. For instance, ev4 corresponds to the activity `c` in σ_L and the `evEND` was included to indicate the end of the trace. Notice that the invisible transition `inv` was explicitly introduced. For this example, the initial state is represented as:

```
(: init
 (tracePointer ev1)
 (token p1)
)
```

and the desired goal state G is:

```
(: goal (and (tracePointer evEND) (token p4)
        (not (token p1)) (not (token p2))
        (not (token p3))
))
```

The goal is to complete the alignment, i.e., to have one token in `p4` and zero tokens in any other places in the Petri net. Additionally, the *tracePointer* has reached the end of the trace σ_L under analysis.

3.2. Domain description

The main component of the domain description is the set of actions. The three possible legal moves given in Definition 2 are represented as actions [Leoni and Marrella, 2017]:

- **Synchronous move action:** An action of this type is created for each pair of transition $t \in T \backslash inv$ and event $e \in \sigma_L$ to represent a synchronous move.
- **Model move action:** An action of this type is created for each transition $t \in \mathcal{T}$.
- **Log move action:** An action of this type is created for each event $e \in \sigma_L$ to represent a log move.

The quantity of actions will vary depending on the trace being aligned as well as the process model. For trace $\sigma_L = \langle a, b, b, c \rangle$ and the Petri net in Figure 1 there are four synchronous move actions, five model move actions and four log move actions. For example, the synchronous move for c is represented by the following action:

```
(:action moveSync#c#ev4-evEND
    :parameters (?ev4 - event ?evEND - event)
    :precondition (and (token p3)
                  (tracePointer ev4))
    :effect (and (not (token p3))
                 (token p4)
                 (not (tracePointer ev4))
                 (tracePointer evEND)
            ))
```

Suppose we are in the state where `(token p3)` and `(tracePointer ev4)` are true, i.e. the place `p3` has a token in the Petri net and `ev4` (that corresponds to `c`) is the actual event of the trace σ_L under analysis. Since the preconditions of action `moveSync#c#ev4` are satisfied, after applying this action, the effects are: (i) the token in place `p3` is consumed and one token is produced in place `p4`; and (ii) the trace pointer moves from `ev4` to `evEND`. Thus, in the next state `(token p4)` and `(tracePointer evEND)` are true and then we have reached the desired goal state.

In the proposal of Leoni and Marrella [2017], a standard cost-function was used. Additionally, it is possible to associate manually different and non-unitary costs for each action.

4. Approaches to generate history-base cost-function to find probable alignments

The computation of probable alignments is based on a history-base cost-function which considers the probability of an activity to be executed [Alizadeh et al., 2015]. Koorneef et al. [2018] and Alizadeh et al. [2015] focus on generating this cost-function taking into account the history executions of the process model. In section 4.1, we specially describe how Koorneef et al. [2018] compute the probabilities of log and model move. These probabilities are used in our proposal. In Section 4.2 , we describe functions that are called *cost-profile*, responsible for transforming the probabilities into costs. One of them is used in our proposal.

4.1. Probabilities of a log and model moves

Next, we formally present the definitions of the probabilities of log and model moves proposed by Koorneef et al. [2018]. The probability of a log move is based on the probability of an activity in the event log. In the last one, Koorneef et al. [2018] also takes into account unobserved activities.

Definition 4 (Probability of an activity in the event log [Koorneef et al., 2018]) *Let an activity X be a discrete random variable with an outcome in the set of activities $\mathcal{A}$. Let us extend the set of activities to allow for unobserved activities $\mathcal{B}$, for example $\mathcal{B} = \{*\}$. The probability of seeing outcome i is $P(X = i) = \theta_i, \forall i \in \mathcal{A} \cup \mathcal{B}$, where $0 \leq \theta_i \leq 1$ and $\sum_{i \in \mathcal{A} \cup \mathcal{B}} \theta_i = 1$. θ could be estimated based on the event log. Let $\hat{\theta}$ be the estimate of the true probability θ. Given the observations, $\hat{\theta}$ can be computed by $\hat{\theta} = E(|X = i|) = \frac{|X=i|}{\sum_{j \in \mathcal{A}} |X=j|} = \frac{|X=i|}{n}, \forall i \in \mathcal{A} \cup \mathcal{B}$. Where the $|.|$, is the number of times the outcome appears.*

Definition 5 (Probability of a log move [Koorneef et al., 2018]) *The estimated probability of a log move is:*

$$\hat{\theta_i^L} = E(|X^L = i|) = \frac{1 - \hat{\theta}_i}{k - 1}, \forall i \in \mathcal{A} \cup \mathcal{B}, \tag{1}$$

where k is the number of possible outcomes in $\mathcal{A} \cup \mathcal{B}$, $0 \leq \hat{\theta_i^L} \leq 1$ and $\sum_{i \in \mathcal{A} \cup \mathcal{B}} \hat{\theta_i^L} = 1$.

According to Definition 5, the most likely log move should be the least frequent activity in an event log.

Definition 6 (Probability of a model move [Koorneef et al., 2018]) *The probability of a model move is equal to the probability of a transition given a marking in a labelled Petri net N. Let marking M be the state of a labelled Petri net. Given N, a marking M is a discrete random variable with outcomes in the set of markings. To compute this probability, N is represented by a Markov Chain where states are markings. The probability of seeing outcome i given $M = m$ is $P(\mathcal{T} = i|M = m) = \phi_{i|m}, \forall t \in T$, where $0 \leq \phi_{i|m} \leq 1$ and $\sum_{i \in T} \phi_{i|m} = 1$. The estimate of the true probability of ϕ, $\hat{\phi}$ is computed by:*

$$\hat{\phi}_{i|m} = E(|t = i|m|) = \frac{|t = i|m|}{\sum_{j \in T} |t = j|m|}, \forall t \in T. \tag{2}$$

Note that in Definitions 5 and 6, the independence of the events is guaranteed and, consequently, we only have one cost associated with each type of activity.

4.2. Cost-profile functions

Note that, as mentioned, a probability is assigned to activities, but it is necessary to transform this probability into a cost. The *cost-profile* functions are responsible for these transformation. In this functions, a low cost must be assigned for a movement associated to activities whose execution are more probable, and a high cost for movements associated with unlikely activities.

In [Koorneef et al., 2018], the probabilities p are transformed into costs and then a minimization of the sum of costs is performed using the A^* algorithm. The *cost-profile* function used in [Koorneef et al., 2018] is $f_1(p) = -log(p)$.

Alizadeh et al. [2015] propose a different form to compute costs for each model and log move that is also based on a event log. However, an activity can have more than one cost, if this activity occurs more than 1 time in the trace. This feature prevents its use together with the proposal of Leoni and Marrella [2017] because in this case we only need to assign one cost for each action. In [Alizadeh et al., 2015] the A* algorithm is also used and three different types of *cost-profile* were defined:

$$f_2(p) = \frac{1}{p} \qquad f_3(p) = \frac{1}{\sqrt{p}} \qquad f_4(p) = 1 + \log \frac{1}{p}. \tag{3}$$

5. The history-based approach applied in Automated Planning to find probable alignments

In order to find probable alignments taking into account the history executions, we combine the proposal of Leoni et al. [2017] and Leoni and Marrella [2017] with the history-base cost-function proposed by Koorneef et al. [2018] that has been slightly modified. In section 5.1 we show an overview of our approach. In section 5.2 we describe the modifications made to the methods presented in the approach presented in Section 4 to compute the probabilities of both log and model moves. We also present a formal definition of the *history-base cost-function* used in this paper.

5.1. Overview of the histoy-based approach applied in Automated Planning

Figure 3 shows an overview of our approach to generate the most probable alignment. Given a real-life process execution recorded in a **training event log** which includes only the traces in accordance with the process model and that are used to automatically calculate the probabilities of the activities and the costs. These costs are included in the planning domains. The next parts are equal to the proposal of Leoni et al. [2017] and Leoni and Marrella [2017]. The planning domains and the planning problems files are generated from a Petri net and the event log that includes the original traces that will be aligned, called **event log to align** [1]. Given these two files, an off-the-shelf planner could be use to return a sequence of actions that are then transformed to the aligned trace.

5.2. Computing a history-base cost-function

We use history executions of the process model, which were recorded in the training event log and are in *conformance with the model*, to generate costs automatically. The

[1] Notice that a planning domain and a planning problem are generated for each trace of the event log to align.

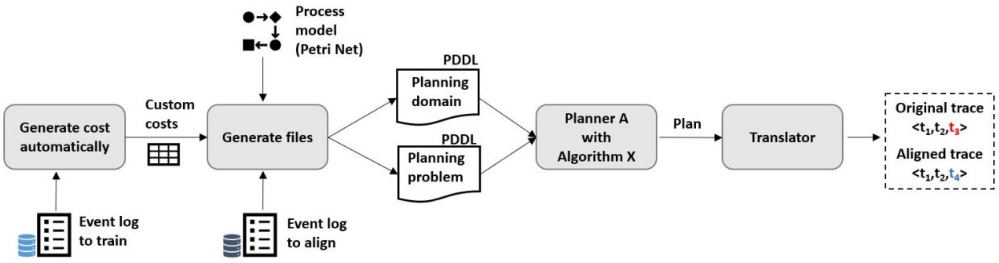


Figure 3. An overview of the history-based approach applied in Automated Planning to generate the most probable alignment.

method used to calculate the log move probabilities is very similar to that proposed in equation 1, however, we will consider that the entire set of activities is known, so, taking this as an assumption, there is no need to add the set $\mathcal{B}$ for unobserved activities. For example, considering the activity a, the number of observations of a is 201, and the sum of all observed activities is 809. Therefore the estimated probability of a log move ($\hat{\theta_a^L}$) is equals to $\frac{1-201/809}{4-1}$. For the calculation of model moves, we use equation 2, but we do not calculate the probability of invisible transitions and assign cost 0 to it.

Definition 7 (history-base cost-function of an alignment) *The history-base cost-function κ assigns a non-negative cost to each legal move:*

$$\kappa(s_L, s_M) = \begin{cases} 0 & s_L = s_M \\ 0 & s_L =\gg \ and \ s_M \in Inv \\ f(\hat{\phi}_{i|m}) & s_L =\gg \ and \ s_M \notin Inv \\ f(\hat{\theta}) & s_M =\gg \end{cases} \tag{4}$$

Note that a cost 0 is assigned to a synchronous move or a move of an invisible transition, otherwise, a non-negative cost is assigned for a log move or a model move. The cost of an alignment between the event log and the labelled Petri net is computed as the sum of the cost of all constituent moves.

We adopted $f_4(p) = 1 + log(1/p)$ as a cost-profile, considering the good results obtained with this function by Alizadeh et al. [2015]. The costs generated using Equation 4 are decimal numbers and must be natural numbers to be used in the PDDL, so we multiplied all results by a factor of 1000 and then truncated this result.

In the first alignment y_1 shown in Figure 2, the two synchronous moves have cost 0, the probability of the model move where $s_L =\gg$ and $s_M = d$ is 20%, and the cost associated with the cost-profile f_4 is 1.698, then after multiplying with 1000 we obtain 1698. The total cost of y_1 is $0 + 0 + 1698$. For y_2 the two synchronous moves have cost 0, the probability of the model move where $s_L =\gg$ and $s_M = c$ is 80%, and the cost associated with the cost-profile f_4 is 1.096, then after multiplying with 1000 we obtain 1096. The total cost of y_2 is $0 + 0 + 1096$. Thus, the most probable alignment is y_2.

6. Implementation and Validation

The implementation[2] was based on the GUI software provided by Leoni et al. [2017]. This tool is integrated with the Fast-Downward planning framework [Helmert, 2006] to find optimal alignments. We extend this tool with our approach for automatic generation of the costs based on the history. To use the history-base cost-function, we import a training event log, used to generate the costs automatically. Thus, the tool computes the most probable alignment for each trace in the event log being aligned, with base on the frequency of the traces in the training event log.

Experiments were performed on synthetic and real-life event logs and process models. We compare the standard cost-function and a history-base cost-function. A machine with an Intel(R) Core(TM) i5-2430M CPU 2.80GHz and 4GB RAM was used.

The evaluation method is similar to that used by Alizadeh et al. [2015]. Initially, we separated the event log in training event log (used to calculate the probabilities of the activities) and testing event log (used to do the actual alignment). The split ratio was 80% for the training event log and 20% for the testing event log. Only conforming traces were included in the training event log, the real-life event log was filtered to ensure only conforming traces. After that, we inserted noise into the testing event log, in an increasing proportion of 10%, 20%, 30% and 40% of noise. The traces in the initial event log, i.e, the one before splitting and inserting noise, is what we call **original or correct traces**.

To measure the ability of the approaches to reconstruct the original traces, we analyzed two metrics: the correct alignment metric (CA) and the Levenshtein distance (LD). The CA metric is calculated by comparing the correct trace (which conforms to the model) with both traces, the one reconstructed using the standard cost-function and the one reconstructed using the history-base cost-function. Therefore, if the reconstructed trace is equals to the correct trace, we increase CA in 1. The metric LD [Levenshtein, 1966] computes the minimum number of activities edits (insertions, deletions or substitutions) required to change the reconstructed trace to obtain the correct trace. Finally, we compute the gain results that show how much better our approach was compared to the standard cost-function. This is done by using the values of CA and LD obtained from both cost-functions and applying the formula $1 - \frac{history\_base\_cost\_function}{standard\_cost\_function} * 100$.

6.1. Synthetic Event Log and Process Model

For the experiments with synthetic data, we used the PLG2 [Burattin, 2015] tool to generate the event logs conforming to their respective model. We also generate four process models, with increasing sizes, according to the method used in [Leoni et al., 2018]. By size, we are talking about visible and invisible transitions. So, we generated process models of 23, 35, 50 and 96 transitions. For each process model, we created an event log with 20,000 traces. The training event logs and testing event logs contain 16000 and 4000 traces, respectively. The results are shown in Table 1.

As the model complexity increased, the CA metric decreased, both for the standard cost-function and for the history-base cost-function. Despite that, in the first two models, we verify our approach obtained better results than the approach using the stan-

[2]Detalhes omitidos por double-blind reviewing

Table 1. Results of experiments on synthetic data.

PN Length	*Noise 10%*				*Noise 20%*			
	Standard Cost-function		*History-base Cost-function*		*Standard Cost-function*		*History-base Cost-function*	
	CA	**LD**	**CA**	**LD**	**CA**	**LD**	**CA**	**LD**
23	382	10024	396	9781	658	7383	676	6831
35	539	14762	541	14625	534	18809	548	18306
50	0	92424	0	92027	0	108579	0	107676
96	0	121068	0	120525	0	137203	0	135158
	Noise 30%				*Noise 40%*			
PN Length	*Standard Cost-function*		*History-base Cost-function*		*Standard Cost-function*		*History-base Cost-function*	
	CA	**LD**	**CA**	**LD**	**CA**	**LD**	**CA**	**LD**
23	616	7801	648	6940	560	9335	613	8261
35	467	23639	504	22768	398	28391	448	27468
50	0	119276	0	118015	0	128879	0	127478
96	0	153515	0	147135	0	169759	0	159849

dard cost-function. Regarding the LD metric, we noticed that in all traces, our approach managed to significantly improve the quality of the alignment found, for all scenarios.

In Table 2 we show the gain, i.e., how much our approach had a better result than the approach using the standard cost-function. For example, for the Petri net of length 23 and noise 10%, our approach had an improvement in CA of 3.7% and LD of 2.5%. An interesting pattern that we can see is that, in most cases, as the amount of noise increased, the gain of LD got with our approach increased as well.

Table 2. Gain of our approach over the standard cost-function on synthetic data.

	Noise 10%		*Noise 20%*		*Noise 30%*		*Noise 40%*	
	CA	**LD**	**CA**	**LD**	**CA**	**LD**	**CA**	**LD**
23	3,7%	2,5%	2,7%	8,0%	5,1%	12,4%	9,5%	13,0%
35	0,4%	0,9%	2,6%	2,7%	7,9%	3,8%	12,6%	3,4%
50	0,0%	0,4%	0,0%	0,8%	0,0%	1,0%	0,0%	1,1%
96	0,0%	0,4%	0,0%	1,5%	0,0%	4,3%	0,0%	6,2%

6.2. Real-life Event Log and Process Model

We used an event record obtained from a fine management system of the Italian police [Mannhardt et al., 2016] to assess the applicability of our approach to real-life scenarios. Figure 4 shows the labelled Petri net for this process model. This is the same real-life event log and process model used by Leoni and Marrella [2017].

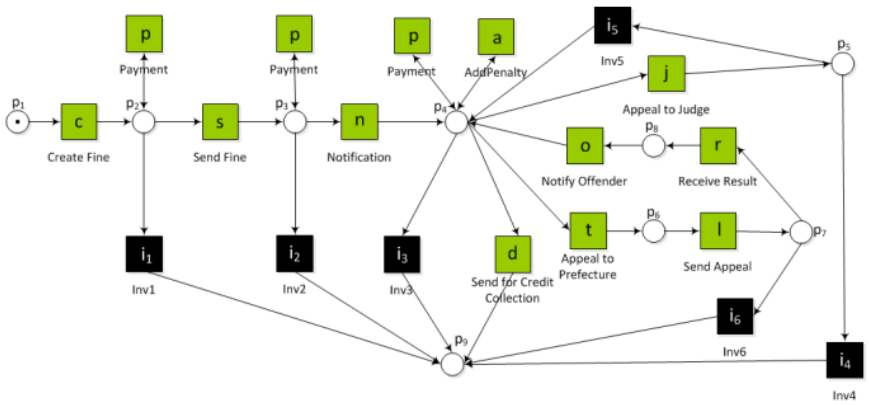


Figure 4. A process model for managing road traffic fines extracted from [Mannhardt et al., 2016].

The event log consists of 150,370 traces, after which we filtered to obtain traces that are in conformance with the process model presented. After that, we obtained a filtered event log with 146,773 traces. In these traces, we applied the same method used to evaluate the experiment results with synthetic data. The result are in Table 3.

Table 3. Results of experiments on real-life data.

	Standard cost-function		*History-base cost-function*		*gain (%)*	
Noise(%)	**CA**	**LD**	**CA**	**LD**	**CA**	**LD**
10	11656	41924	11770	40949	0,97%	2,33%
20	2884	63162	3030	61661	4,82%	2,38%
30	5929	50220	6086	48174	2,58%	4,07%
40	3529	63672	3619	61679	2,49%	3,13%

We can see that there was a significant improvement in all scenarios. For the CA metric, for instance, the less significant result was 0,97% for 10% of noise and the biggest improvement 4,82% for 20% of noise. For the LD metric, the biggest improvement was 4,07% for 30% of noise and the less significant result was 2,33% for 10% of noise.

7. Conclusion

Techniques for conformance checking attempt to see whether process executions truly differ from the process model. A reliable method of conformance checking that allows to identify the deviations that result in nonconformity is provided by the notion of alignment [Van der Aalst et al., 2012]. In this work, we propose a slight modification to the automatic generation of history-base cost-function proposed by Koorneef et al. [2018] and use our version of such a function on top of the planning-based tool of Leoni et al. [2017] to find an optimal probable alignment. A training event log is used to assign probabilities to the occurrence of activities and to the log and model moves. Different from [Koorneef et al., 2018] activities not observed in the event log were not considered. In addition, a cost-profile described in [Alizadeh et al., 2015] was used to generate the costs from the

probabilities. Some manipulations were made in the values of the costs in order to be used in the PDDL. The empirical evaluation with synthetic and real-life event logs and process models of growing complexity showed that our approach improves upon the study of Leoni and Marrella [2017].

References

Adriansyah, A., van Dongen, B. F., and van der Aalst, W. M. (2013). Memory-efficient alignment of observed and modeled behavior. Technical Report 3, BPM Center Report.

Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., SRI, D. W., Barrett, A., et al. (1998). PDDL – the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Alizadeh, M., de Leoni, M., and Zannone, N. (2015). History-based construction of alignments for conformance checking: Formalization and implementation. In *Data-Driven Process Discovery and Analysis*, pages 58–78. Springer International Publishing.

Bloemen, V., van Zelst, S., van der Aalst, W., van Dongen, B., and van de Pol, J. (2022). Aligning observed and modelled behaviour by maximizing synchronous moves and using milestones. *Information Systems*, 103:101456.

Boltenhagen, M., Chatain, T., and Carmona, J. (2021). A discounted cost function for fast alignments of business processes. In Polyvyanyy, A., Wynn, M. T., Van Looy, A., and Reichert, M., editors, *Business Process Management*, pages 252–269. Springer International Publishing.

Burattin, A. (2015). PLG2: Multiperspective processes randomization and simulation for online and offline settings. *preprint arXiv:1506.08415*.

Dunzer, S., Stierle, M., Matzner, M., and Baier, S. (2019). Conformance checking: A state-of-the-art literature review. In *11th Int'l Conf on Subject-Oriented Business Process Management*, pages 1–10.

Helmert, M. (2006). The fast downward planning system. *J Artif Intell Res*, 26:191–246.

Koorneef, M., Solti, A., Leopold, H., and Reijers, H. A. (2018). Automatic root cause identification using most probable alignments. *13th Int'l Workshop on Business Process Intelligence*, pages 204–215.

Leoni, M., Lanciano, G., and Marrella, A. (2017). A tool for aligning event logs and prescriptive process models through automated planning. In *BPM (Demos)*.

Leoni, M., Lanciano, G., and Marrella, A. (2018). Aligning partially-ordered process-execution traces and models using automated planning. *Int'l Conf on Automated Planning and Scheduling*, 28(1):321–329.

Leoni, M. and Marrella, A. (2017). Aligning real process executions and prescriptive process models through automated planning. *Expert Syst Appl*, 82:162–183.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

Mannhardt, F., Leoni, M., Reijers, H., and Aalst, W. V. D. (2016). Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437.

Valk, R. and Vidal-Naquet, G. (1981). Petri nets and regular languages. *Journal of Computer and System Sciences*, 23(3):299–325.

Van der Aalst, W., Adriansyah, A., and van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining and Knowl Discov*, 2(2):182–192.

van der Aalst W. (2016). *Process Mining – Data Science in Action*. Springer.