Translucent Alignments *

Harry H. Beyel, Christopher T. Schwanen, and Wil M. P. van der Aalst

Chair of Process and Data Science (PADS),
RWTH Aachen University, Aachen, Germany
{beyel, schwanen, wvdaalst}@pads.rwth-aachen.de

Abstract. Event logs, the primary data source in process mining, consist of events with a case identifier, an activity, and a timestamp. Translucent event logs extend this by also recording enabled activities alongside executed ones. Such data can be extracted from user interactions or running information systems. Fitness, a quality measure in conformance checking, assesses how well an event log aligns with a process model. While fitness has been widely studied, no prior work has considered enabled activities. We introduce the first fitness measurement incorporating this information based on the well-established alignment technique. Our generalized, parameterized approach also supports traditional fitness measurement. Through qualitative evaluation, we demonstrate our method's applicability and provide insights into the implications of parameters. Our quantitative assessment shows limited computational overhead when enabled activities are considered and provides insights into the fitness score of the classical and translucent approaches. Our results indicate that incorporating enabled activities yields appropriate fitness scores compared to traditional methods, enhancing conformance checking accuracy.

Keywords: Conformance Checking · Alignments · Fitness · Translucent Event Logs · Enabled Activities

1 Introduction

Processes are executed in various organizations and can be digitally tracked and stored in an *event log*. An event log consists of *events*, each having information on case, activity, and timestamp. Beyond these basics, additional data, like the resource executing an event, can be recorded. Moreover, it is possible to record information on *enabled* activities—besides the executed activity. Such information can be added due to domain knowledge or extraction of activities executed in a desktop environment [11, 13]. Note that extracting these logs from desktop environments provides valuable insights for analyzing human-computer interaction. In addition, when creating such logs from a Graphical User Interface (GUI), they provide a great source for robotic process mining [21] and can be used to create bots for robotic process automation [41]. We call event logs that contain information on enabled activities *translucent event logs*.

* We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

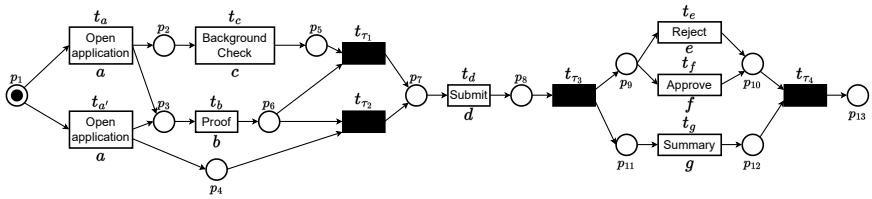


Fig. 1: Example workflow captured within a Petri net.

(a) Open an application. (b) Proof and check. (c) Submit.

Fig. 2: Example GUI for proofing a loan application.

Process mining can be used to analyze event logs to get insights into the execution of processes [1]. Dedicated process-mining techniques can be applied to translucent event logs [2, 12, 15]. One area in process mining is conformance checking, which consists of the fitness quality dimension [18]. Fitness measures how well a process model represents the behavior recorded in an event log. One method to assess fitness is token-based replay [38], which simulates the flow of tokens through the model's transitions according to the observed events. Discrepancies are identified by tracking token movements: missing or remaining tokens indicate deviations between the model and actual execution. However, token-based replay has two main issues: it is non-deterministic when the model contains duplicate tasks, and it can produce misleading diagnostics due to "token flooding". The state-of-the-art technique, alignments [5], overcomes these problems by mapping each event in the log to the corresponding model activities. It finds the optimal sequence of synchronous and asynchronous moves to accurately detect discrepancies and measure conformance.

When computing standard alignments, only executed activities of an event are considered, not the enabled activities. This can lead to false conclusions. To illustrate this, consider a simplified loan application process. In the first step, the application is opened. Afterward, there are two possible actions: either only a proof is conducted, or a proof and a background check are conducted. Subsequently, the results are submitted. In the end, the application is rejected or approved, and a summary is created. The workflow is depicted in Figure 1. A possible application process handling with a GUI is depicted in Figure 2 and relates to the former description. Assume the sequence $\langle a, b, d, e, g \rangle$ of activi-

ties has been executed and is linked to the scenario shown in Figure 2. When measuring fitness between this sequence and the model, the result is a perfect score of 1.0. The sequence of executed transitions is $\langle t_{a'}, t_b, t_{\tau_2}, t_d, t_{\tau_3}, t_e, t_g, t_{\tau_4} \rangle$. However, as denoted in Figure 2b, it is also possible to perform a background check, resulting in the activity being enabled. But when executing $t_{a'}$, b and c are not enabled, thus not fitting the information available on enabled activities extracted from the GUI example. Thus, the score should not be 1.0 but lower.

The information on a mismatch of enabled activities is valuable when evaluating which process model should be used to create a bot for automating tasks in a GUI. The bot may misbehave if a wrong model is chosen based on misleading scores. Moreover, we can verify a system's implementation by utilizing information on enabled activities. Usually, a reference model exists that describes how a system should behave. An implementation should follow this model. By extracting enabled activities from screenshots during the interaction with the software, we can align the recorded behavior with the reference model. When the recording and extraction of activities from screenshots work as intended and a translucent event log is created, translucent alignments can spot context-aware deviations. This provides new insights into system verification, enabling large-scale verification across different user groups and systems. In addition, the information on enabled activities can guide alignments and increase their accuracy. Although a traditional alignment represents an optimal path through a model for a given sequence of activities, there might be multiple optimal paths. Techniques that use alignments as input, for example, the precision method presented in [6], are based on this and are not deterministic, leading to potentially different results in each run. To illustrate this, consider the trace $\langle a, b, d, e, e \rangle$. We assume that for the first e, f and g are also enabled. For the second e, we assume that f is also enabled. Within the classic alignment, a model and log move would be created to execute g, but their position is non-deterministic. However, since we know from the translucent event log that g was possible but not performed, we can create a better alignment using the enabled activity to guide the alignment computation.

In this work, we develop an approach that adjusts the fitness score by considering information on enabled activities, penalizing discrepancies. Our method allows for higher fitness when enabled activities match but executed activities differ, and enables verification of system requirements using enabled activities. At the same time, our approach acts as a generalization of the existing approach.

2 Related Work

Much work has been done to measure fitness in process mining. Besides the basic idea of footprint comparison between an event log and a process model, advanced techniques were developed. One of them is the aforementioned token-based replay [38]. The idea of token-based replay is to replay a trace in a Petri net. In this process, missing tokens might be added to allow the firing of otherwise impeded transitions. The number of tokens added in this way and the number

of tokens consumed, produced, and remained after finishing a trace are tracked. Considering the different amounts and all traces, a fitness score is computed. This approach has also been adjusted for an online setting [16]. Despite recent advances [8, 9], which solve issues of token-based replay, alignments are still recognized as the state-of-the-art technique.

Alignments, presented in [5], use more advanced concepts than token-based replay. For each trace of an event log, a trace net based on Petri nets is created. Between the process model, which is given as a Petri net, and the trace net, a synchronous product net is created by following predefined rules. The firing of a transition in a synchronous product net is associated with costs. The goal is to find the cheapest sequence from the initial marking to the final marking of the synchronous product net. Over the last years, advances for the consuming computation of an alignment have been proposed [19, 20, 29, 35–37, 39, 40, 42, 43]. Some extensions consider information beyond the control flow, e.g., data-aware alignments [27, 28, 34] or resource-aware alignments [7].

In [17], weighted artificial negative events are presented. Negative events represent information about activities prevented from taking place in a process. These are rarely logged in real-life event logs. The generation algorithm induces these negative events artificially by examining each position in each trace within an event log and determining which activities cannot occur at those positions based on the observed behavior. This contrasts our method since we use a source available in the data and the model to compute our score.

There are also techniques for measuring fitness in a stochastic setting. Stochastic fitness measurements were developed in [24, 25, 33]. In addition, techniques for object-centric event data were developed. A general approach is presented in [4], while alignments for an object-centric setting are presented in [30].

Concerning the usage of translucent event logs in process mining, recent work has shown that the information that translucent event logs convey can improve existing process-discovery techniques [12]. Concerning conformance-checking techniques using translucent event logs, there exists work in measuring precision by considering enabled activities [15]. However, no approach measures fitness between a Petri net and a translucent event log so far.

3 Preliminaries

In this section, we define the basic concepts that we use in the remainder of this work. We define sets, sequences, and tuples and operations on them. Furthermore, we define translucent event logs and traces. Finally, we define Petri nets and related concepts.

Given a set X and a function f, $f(X) = \{f(x) \mid x \in X\}$ denotes applying the function f on all elements of set X. For simplicity, we denote for any set $X^{\gg} = X \cup \{\gg\}$.

Definition 1 (Tversky Index). Given sets A and B, the Tversky index provides an asymmetric similarity measurement between sets defined as

$$S_{\alpha,\beta}(A,B) = \frac{|A \cap B|}{|A \cap B| + \alpha \cdot |A \setminus B| + \beta \cdot |B \setminus A|},$$

with $\alpha, \beta \geq 0$. When α is high relative to β, the index becomes more sensitive to elements in A that are not in B, highlighting the uniqueness of A. A similar situation applies to β. If $\alpha = \beta = 1$, this corresponds to the (symmetric) Jaccard index. We write $D_{\alpha,\beta}(A,B) = 1 - S_{\alpha,\beta}(A,B)$ to denote the Tversky distance.

$\mathcal{B}(X)$ denotes the set of all multisets over set X. E.g., if $X = \{x, y, z\}$, an example multiset is $[x, x, y] = [x^2, y]$. Let A be a set and let $t = (a_1, \ldots, a_n) \in A \times \cdots \times A$ be a tuple of n elements. $\pi_i(t)$ refers to the ith element of the tuple, e.g., $\pi_1((a,b)) = a$. Given a set X, $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle \in X^*$ denotes a sequence over X. σ_i denotes the sequence's ith element. The length of a sequence σ is denoted as $|\sigma|$. The concatenation of two sequences is denoted as $\sigma \cdot \sigma'$. Given a sequence σ and a set X', $\sigma{\restriction_{X'}}$ denotes a sequence projection, e.g., $\langle a,b,c,d \rangle {\restriction_{\{a,c\}}} = \langle a, c \rangle$. $pref_i(\sigma) = \langle \sigma_1, \ldots, \sigma_i \rangle$ refers to the prefix of a sequence containing the first i elements. $pref_0 = \langle\rangle$. For any sequence of tuples $\sigma \in (A \times \cdots \times A)^*$, we define $\pi_i^*(\sigma) = \langle \pi_i(\sigma_1), \ldots, \pi_i(\sigma_{|\sigma|}) \rangle$.

Translucent event logs capture information on enabled activities in addition to executed ones. Hence, the executed activity must also be enabled in the corresponding event. Also, we assume that all enabled activities in an event log are performed at some point. $\mathcal{U}_{case}$ is the universe of case identifiers, $\mathcal{U}_{act}$ is the universe of activity names, and $\mathcal{U}_{time}$ is the universe of timestamps.

Definition 2 (Translucent Event Log, Trace). $\mathcal{U}_{ev}$ is the universe of events. $e \in \mathcal{U}_{ev}$ is an event, $\pi_{case}(e) \in \mathcal{U}_{case}$ is the case of e, $\pi_{time}(e) \in \mathcal{U}_{time}$ is the time of e, $\pi_{en}(e) \subseteq \mathcal{U}_{act}$ are the enabled activities of e, $\pi_{act}(e) \in \pi_{en}(e)$ is the activity of e. A translucent event log L is a set of events $L \subseteq \mathcal{U}_{ev}$. In addition, $\bigcup_{e \in L} \pi_{en}(e) = \bigcup_{e \in L} \{\pi_{act}(e)\}$. For simplicity, we assume that events in L are totally ordered such that for $e_1, e_2 \in L$, $e_1 < e_2$ implies $\pi_{time}(e_1) \leq \pi_{time}(e_2)$. A trace is a sequence of all events of a case ordered from earliest to latest, i.e., $\sigma^{L,c} = \langle e_1, \ldots, e_n \rangle$, such that for $c \in \pi_{case}(L)$, $\{e_1, \ldots, e_n\} = \{e \in L \mid \pi_{case}(e) = c\}$ and $e_1 < \ldots < e_n$. The set of traces of L is denoted as $\Sigma^L = \{\sigma^{L,c} \mid c \in \pi_{case}(L)\}$.

An example translucent event log is showcased in Table 1. For e_1, $\pi_{case}(e_1) = 1$, $\pi_{act}(e_1) = a$, $\pi_{en}(e_1) = \{a\}$, and $\pi_{time}(e_1) =$ 2024-06-20 13:37:37. In the remainder of the paper, for simplicity, we denote the enabled activities and underline the executed activity of an event. For example, we can denote for the first trace in Table 1, $\sigma^{L,1} = \langle e_1, e_2, e_3, e_4, e_5 \rangle$, with $\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}fg, \underline{g} \rangle$.

Definition 3 (Petri Net). A Petri net is a tuple $N = (P, T, F, A, l)$, where P is a set of places, T is a set of transitions such that $P \cap T = \emptyset$, and $F \subseteq (T \times P) \cup (P \times T)$ is a set of directed arcs. $A \subseteq \mathcal{U}_{act} \cup \{\tau\}$ is a set of activity labels, and $l \colon T \to A$ is a labeling function where τ denotes the silent activity.

Table 1: Exemplary translucent event log.

Event	Case	Activity	Enabled Activities	Timestamp
e_1	1	a	{a}	2024-06-20 13:37:37
e_2	1	b	{b, c}	2024-06-20 13:37:38
e_3	1	d	{d}	2024-06-20 13:37:39
e_4	1	e	{e, f, g}	2024-06-20 13:37:40
e_5	1	g	{g}	2024-06-20 13:37:41
e_{11}	2	a	{a}	2024-06-20 13:37:51
e_{12}	2	c	{c}	2024-06-20 13:37:52
e_{13}	2	d	{d}	2024-06-20 13:37:53
e_{14}	2	x	{x, e, g}	2024-06-20 13:37:54
e_{15}	2	g	{g}	2024-06-20 13:37:55

$M \in \mathcal{B}(P)$ is a marking and (N, M) refers to the Petri net N in marking M. We define $T_\tau = \{t \in T \mid l(t) = \tau\}$.

We focus on sound workflow nets [3]. Furthermore, for computational reasons, we assume that each transition that is connected to the sink place is a non-silent transition. To ensure that, we introduce a new, non-silent transition with a unique label and a new sink place. This transition, labeled *end*, has an ingoing arc from the old sink place and an outgoing arc to the new sink place. Also, we assume that an artificial end event e_{end} is added at the end of each trace, with $\pi_{act}(e_{end}) = end$ and $\pi_{en}(e_{end}) = \{end\}$. Given the Petri net in Figure 1, $P = \{p_1, p_2, \ldots\}$, $T = \{t_a, t_{a'}, t_b, t_c, t_{\tau_1}, t_{\tau_2}, \ldots\}$, $F = \{(p_1, t_a), (p_1, t_{a'}), \ldots, (t_{\tau_2}, p_7)\}$ and $A = \{a, b, c, \ldots, \tau\}$. The shown Petri net is in marking $[p_1]$. Petri net firing rules can be found in [1].

Definition 4 (Firing Transition and Sequence). Let $N = (P, T, F, A, l)$ be a Petri net. Let $M, M' \in \mathcal{B}(P)$ be two markings. The firing of a single transition $t \in T$ is denoted as $(N, M) \xrightarrow{t} (N, M')$. The successive firing of all transitions in $\sigma \in T^*$ is denoted as $(N, M) \xrightarrow{\sigma} (N, M')$. In addition, $(N, M) \xrightarrow{\langle\rangle} (N, M)$. Let $M_i \in \mathcal{B}(P)$ be the initial marking. We define a marking $M' \in \mathcal{B}(P)$ as reachable in (N, M_i), if there exists $\sigma \in T^*$ such that $(N, M_i) \xrightarrow{\sigma} (N, M')$. The set of all reachable markings starting in (N, M_i) is denoted as $[N, M_i\rangle$.

4 Translucent Fitness

In the classic alignment, moves on the model and log are compared. A cost function assigns costs to the different move types, and the goal is to compute the alignment with the lowest costs while still meeting defined requirements. The costs are used to determine the fitness score. In this section, we show how translucent alignments and a translucent fitness score are computed. We first define translucent alignments and the corresponding moves and costs. We provide a

high-level overview to convey the idea. Second, we show how an optimal translucent alignment is computed. Finally, we show how a fitness score is computed on the trace and log level.

4.1 Defining Translucent Alignments

Moves on the log and model side are essential for alignments. An alignment is computed for each trace of the translucent event log. There are two key requirements for an alignment: first, the sequence of events must yield the original trace; second, the sequence of transitions should represent a firing sequence from a Petri net's initial to final marking. The following definition captures this.

Definition 5 (Translucent Alignment). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, and $N = (P, T, F, A, l)$ be a Petri net with its initial marking $M_i \in \mathcal{B}(P)$ and final marking $M_f \in \mathcal{B}(P)$. An alignment $\gamma \in (L^{\gg} \times T^{\gg})^*$ between σ and N is a sequence such that:

- $\pi_1^*(\gamma)\restriction_L = \sigma$
- $(N, M_i) \xrightarrow{\pi_2^*(\gamma)\restriction_T} (N, M_f)$

For illustration, the following is an alignment between the first case of the event log shown in Table 1 and the Petri net depicted in Figure 1:

$$\gamma = \left|\begin{array}{|c|c|c|c|c|c|c|c|}
e_1 & e_2 & \gg & e_3 & \gg & e_4 & e_5 & \gg \\
\underline{a} & \underline{bc} & \gg & \underline{d} & \gg & \underline{efg} & \underline{g} & \gg \\
\hline
\underline{a} & \underline{b} & \tau & \underline{d} & \tau & \underline{efg} & \underline{g} & \tau \\
t_{a'} & t_b & t_{\tau_2} & t_d & t_{\tau_3} & t_e & t_g & t_{\tau_4}
\end{array}\right|.$$

Given this alignment, $\gamma = \langle (e_1, t_{a'}), (e_2, t_b), \ldots, (e_5, t_g), (\gg, t_{\tau_4}) \rangle$, we denote $\pi_1^*(\gamma)\restriction_L = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and $\pi_2^*(\gamma)\restriction_T = \langle t_{a'}, t_b, t_{\tau_2}, t_d, t_{\tau_3}, t_e, t_g, t_{\tau_4} \rangle$. For our work, information on enabled activities is necessary. While events in a translucent event log have this information, we also need to access this information in a model. Note that silent transitions do not contribute useful information since their execution is not recorded. As they serve as synchronous points in the model, we should seek enabled activities beyond them when possible.

Definition 6 (Enabled Activities in a Model). Let $N = (P, T, F, A, l)$ be a Petri net and $M_i \in \mathcal{B}(P)$ the initial marking. For a marking $M \in [N, M_i\rangle$, we define its set of enabled activities as $en_M^N(M) = \{l(t) \mid \exists_{\sigma \in T_\tau^*, t \in T \setminus T_\tau, M' \in \mathcal{B}(P)} (N, M) \xrightarrow{\sigma \cdot \langle t \rangle} (N, M')\}$. Given a sequence $\sigma \in T^*$, we define $en_\sigma^N(\sigma) = en_M^N(M'')$ if $(N, M_i) \xrightarrow{\sigma} (N, M'')$.

Given the Petri net in Figure 1, we denote for marking $[p_2, p_3]$ activities b and c as enabled, and for marking $[p_8]$ activities e, f, and g as enabled. As denoted, alignments contain a sequence of transitions. This sequence leads to a marking in a Petri net. We use the transition sequence to access the information on enabled activities at certain points in an alignment.

Definition 7 (Enabled Activities in an Alignment). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma = \langle (e_1, t_1), \ldots, (e_{|\gamma|}, t_{|\gamma|}) \rangle \in (L^{\gg} \times T^{\gg})^*$ be an alignment. For any $i \in \{1, \ldots, |\gamma|\}$ and if $t_i \neq \gg$ we define $en^{\gamma}(t_i) = en^N_{\sigma}(\pi^*_2(pref_{i-1}(\gamma))\restriction_T)$.

Given the Petri net in Figure 1 and the alignment shown before, we can denote $en^{\gamma}(t_{a'}) = \{a\}$ and $en^{\gamma}(t_b) = \{b\}$.

In our work, we reinterpret the existing moves (synchronous, visible model, invisible model, and log move) and add three new moves that act as a generalization of the previous moves:

- *execution-change move*: when the enabled activities in the model and the event match, but the executed activities do not match. To illustrate this, consider the model displayed in Figure 1 and the trace $\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}fg, \underline{e}f \rangle$. For the second-to-last event, changing the execution from e to g results in reaching the final marking.
- *enabled-change move*: when the executed activities match, but the enabled activities do not match. As an example, consider the first case of the event log in Table 1. For e_2, b and c are enabled, but in the model, after executing $t_{a'}$, b is enabled, but c is not. Thus, more activities are enabled in the log than in the model. Also, the other way around is possible.
- *execution-enabled-change move*: a mixture of the previous two moves. Consider the second trace shown in Table 1. When executing $t_{a'}$, only b is enabled. In e_{12}, only c is enabled and gets executed. Thus, by changing the set of enabled activities and the execution change, we can align e_{12} with t_b.

The following definition captures the different moves formally.

Definition 8 (Moves). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. For all tuples $(e_i, t_i) \in \gamma$, $i \in \{1, \ldots, |\gamma|\}$, we define the following movements:

- *synchronous move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) = l(t_i)$ and $\pi_{en}(e_i) = en^{\gamma}(t_i)$.
- *log move* if $e_i \in L$ and $t_i = \gg$.
- *invisible move on model* if $e_i = \gg$ and $t_i \in T$, $l(t_i) = \tau$.
- *visible move on model* if $e_i = \gg$ and $t_i \in T$, $l(t_i) \neq \tau$.
- *execution-change move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) \neq l(t_i)$, $l(t_i) \neq \tau$, but $\pi_{en}(e_i) = en^{\gamma}(t_i)$.
- *enabled-change move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) = l(t_i)$, but $\pi_{en}(e_i) \neq en^{\gamma}(t_i)$.
- *execution-enabled-change move* if $e_i \in L$ and $t_i \in T$, $l(t_i) \neq \tau$, $\pi_{act}(e_i) \neq l(t_i)$, and $\pi_{en}(e_i) \neq en^{\gamma}(t_i)$.

Next, we define the costs for the different types of moves. Note that the synchronous move is a special case of the enabled-change move. Also, the execution-change move is a special case of the execution-enabled-change move.

Definition 9 (Costs). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. For all tuples $(e_i, t_i) \in \gamma$, $i \in \{1, \ldots, |\gamma|\}$, we define the following cost function $\delta \colon L^{\gg} \times T^{\gg} \to [0, 2]$:

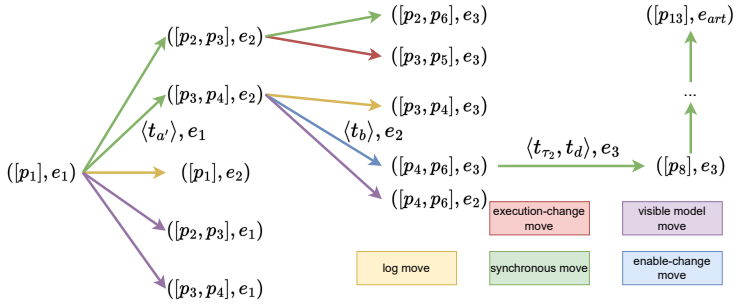


Fig. 3: Part of the synchronous state space between the trace $\langle \underline{a}, \underline{bc}, \underline{d}, \underline{efg}, \underline{g} \rangle$ from the translucent event log shown in Table 1 and the Petri net displayed in Figure 1.

- synchronous and enabled-change move: $\delta(e_i, t_i) = D_{\alpha,\beta}(\pi_{en}(e_i), en^{\gamma}(t_i))$.
- log move: $\delta(e_i, t_i) = 1$.
- invisible move on model: $\delta(e_i, t_i) = 0$.
- visible move on model: $\delta(e_i, t_i) = 1$.
- execution-change move and execution-enabled-change move: $\delta(e_i, t_i) = 1 + D_{\alpha,\beta}(\pi_{en}(e_i), en^{\gamma}(t_i))$

Note that these moves do not result in higher costs than the classic worst-case scenario. Concerning the execution-enabled-change move, the worst case of a mismatch results in costs of 2, which correspond to a log and model move.

4.2 Computing Optimal Translucent Alignments

The cost of an alignment corresponds to the sum of costs of the moves composing the alignment. An optimal alignment is an alignment with minimal costs among all possible alignments. Compared to the standard alignment approach, we just extended the possible moves and used a customized cost function. This adjustment still allows us to apply the general alignment computation method, which essentially constructs the synchronous state space and solves a shortest-path problem.

Figure 3 illustrates the finding of the shortest path in the synchronous state space for our exemplary trace $\langle \underline{a}, \underline{bc}, \underline{d}, \underline{efg}, \underline{g} \rangle$ and Petri net. This also corresponds to the alignment example introduced previously. The distance between states corresponds to the cost of the move that connects them. We annotated the first three arcs of the shortest path with information on the moves they are representing. Although invisible log moves are not seen directly and, therefore, not given a separate color, they are still contained in the execution sequences of the model and thus implicitly captured by the arcs in the given example of the synchronous state space.

Eventually, any shortest path from the initial to the final state represents an optimal alignment. For the exemplary alignment, we obtain the following costs: $0 + 1 - \frac{|\{b,c\} \cap \{b\}|}{|\{b,c\} \cap \{b\}| + \alpha|\{c\}| + \beta|\emptyset|} + 0 + 0 + 0 + 0 = 1 - \frac{1}{1+\alpha}$. When we want to emphasize that there are enabled activities in the translucent event log that are not represented in the model, we set $\alpha > 0$. When $\alpha = 0$, the alignment has no costs; when $\alpha = 1$, the alignment has costs of 0.5.

4.3 Fitness Score

After defining translucent alignments, their costs, and their computation, we define a translucent fitness score. Similarly to the traditional setting, we thereby take the original approach to compute a trace score.

Definition 10 (Trace Score). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, $M_i \in \mathcal{B}(P)$ be the initial marking, $M_f \in \mathcal{B}(P)$ the final marking, and $\gamma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. Let $\sigma_T \in T^*$ be the shortest sequence of transitions through a Petri net, i.e., $(N, M_i) \xrightarrow{\sigma_T} (N, M_f)$ and $\nexists_{\sigma'_T \in T^*} : (N, M_i) \xrightarrow{\sigma'_T} (N, M_f) \wedge |\sigma'_T| < |\sigma_T|$. The trace score is defined as:

$$\theta(\sigma, N, \gamma) = 1 - \frac{\sum_{(e,t) \in \gamma} \delta(e, t)}{\sum_{t \in \sigma_T} \delta(\gg, t) + \sum_{e \in \sigma} \delta(e, \gg) - 2},$$

where the numerator is the actual costs of the alignment, and the denominator is the worst-case costs (shortest firing sequence plus length of the trace). To compensate for the artificial end activity in each trace and transition in the model, respectively, we subtract their additional cost from the denominator. This is not necessary for the numerator since the move between the artificial ends will always be synchronous, leading to costs of 0. As a result, the actual fitness score is obtained.

For our example alignment, the score is $1 - \frac{0.5}{(7+1)+(5+1)-2} \approx 0.96$ if $\alpha = 1$.

Next, we consider all traces of a translucent event log to compute a translucent log score. The following captures that.

Definition 11 (Log Score). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma_\sigma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. The log score is defined as

$$\Theta(\Sigma^L, N) = \frac{\sum_{\sigma \in \Sigma^L} \theta(\sigma, N, \gamma_\sigma)}{|\Sigma^L|}.$$

5 Evaluation

Our evaluation is divided into two sections. First, we provide a qualitative evaluation, highlighting the implications of the different moves and α and β values. Afterward, we present a quantitative assessment of our method. We provide a large-scale evaluation using various logs to compare the traditional fitness score and the translucent fitness score. Moreover, we compare computation times per (translucent) variant.

Table 2: Qualitative evaluation for different traces and models.

Trace	Traditional Fitness Score	Translucent Fitness Score		
		$\alpha = 1, \beta = 1$	$\alpha = 0, \beta = 2$	$\alpha = 2, \beta = 0$
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}fg, \underline{g} \rangle$	1.00	1.00	1.00	1.00
$\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}fg, \underline{g} \rangle$	1.00	0.95	1.00	0.93
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}g, \underline{g} \rangle$	1.00	0.97	0.95	1.00
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{x}eg, \underline{g} \rangle$	0.80	0.85	0.85	0.85
$\langle \underline{a}, \underline{b}, \underline{e}fg, \underline{g} \rangle$	0.89	0.89	0.89	0.89
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}fg, \underline{e}f \rangle$	0.80	0.90	0.90	0.90

5.1 Qualitative Evaluation

First, we highlight the differences between the traditional method and our approach and investigate the impact of the parameters α and β. Results for different scenarios are shown in Table 2. As model, we use our example in Figure 1.

The first row shows a perfectly fitting trace of the executed and enabled activities within the model. As we can denote, the score for all methods is 1.0. The second row highlights when more activities are enabled in the event log than allowed by the model path (specifically activities b and c, while the best path does not allow for c). Traditional alignment methods still return a perfect score because they do not consider enabled activities. When $\alpha = 0$, we also get a perfect score since we do not penalize additional enabled activities in the translucent event log. As we increase α to 1 or 2, the mismatch is penalized more heavily, reducing the fitness score. This reflects the discrepancy between the model and the event log regarding enabled activities. The third row shows the opposite of the former case: fewer activities are enabled in the data than by the model. As before, the traditional methods still produce a perfectly fitting score. When $\beta = 0$, the score is still perfect since this behavior is not penalized. Adjusting β allows us to penalize the mismatch, reducing the fitness score to reflect fewer enabled activities in the event log compared to the model. The fourth row highlights the results when an activity not contained in the model gets executed. The traditional method returns costs 2 (model and log move) and a score of 0.8. Our method assigns a lower cost of 1.5, resulting in a higher fitness score of 0.85. The costs are based on an execution change (cost 1) and the change in enabled activities. This approach accounts for potential noise in the data, such as false recordings of executed activities, since all activities match the enabled activities in the model. Therefore, information on enabled activities makes the fitness score more robust. Thus, our method provides holistic scoring and can determine the execution sequence more accurately, whereas traditional methods might allow for ambiguous alignments (e.g., permitting a model move before or after activity g). The fifth row is about the missing execution of d. All methods provide the same fitness score. The last row highlights the execution change move and the resulting difference in scoring. Again, false recordings can lead to such data. However, our method seems more robust when this happens.

5.2 Quantitative Evaluation

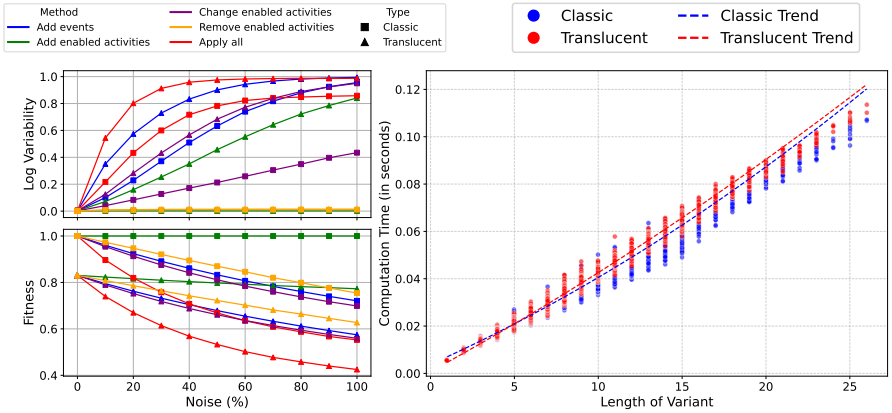
We provide an implementation in Python[1] using *PM4Py* [10]. Our method also computes classic alignments. Thus, we used our implementation for this part. Our data are available [14].

Although there are tools to create translucent event logs (see [11, 13]), we manipulated existing event logs to allow for a large-scale comparison and different complexities. The logs we use are the road traffic fine management [26], the sepsis [31], and the hospital billing log [32]. For each log, we used the *Inductive Miner - infrequent* [23] with a threshold of 40 %, to discover a process model. Using the traditional alignment, we checked for fitting traces and kept them. We converted the Petri nets of each model into a Deterministic Finite Automaton (DFA) using the reachability graph and automata theory. By replaying each trace of a log on a DFA, we added information on enabled activities, thus creating a translucent event log. For more details, we refer to [11, 12, 15]. For each log's fitting and enhanced traces, we applied the *Inductive Miner* [22]. In the next step, we randomly modified the created event logs by either adding, changing, or removing enabled activities of events. If the selected enabled activity of an event is the executed activity, we also modify the executed activity or remove the event. Also, we introduced new events. In addition, we also applied all mentioned modifications to the logs. We added noise with respect to the number of events per log. Besides the classic definition of trace variants, we further introduce translucent variants. Translucent variants also consider the information of enabled activities. For example, $\langle \underline{a}, \underline{b}, \underline{c} \rangle$ and $\langle \underline{a}b, \underline{b}, \underline{c} \rangle$ belong to the same classic variant, but are different translucent variants. For the alignment computation, we used the standard setting of $\alpha = \beta = 1$ and the models discovered by the Inductive Miner [22] on the translucent logs without noise as reference models.

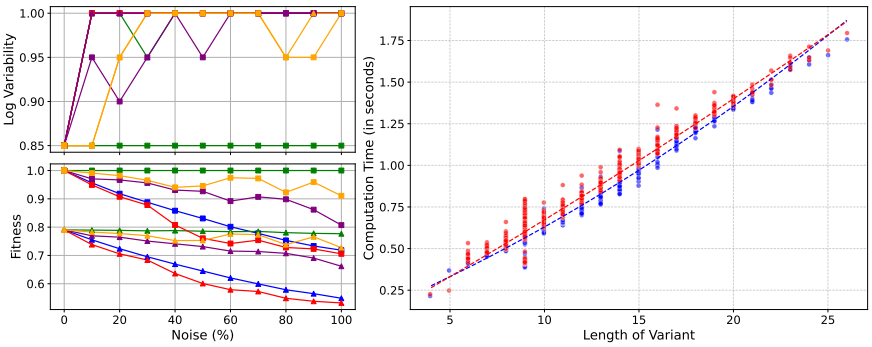
In our evaluation, we checked the log variability for each modified event log, i.e., the ratio between variants and traces. A value of 1 means that each variant has exactly one trace. Alignments are usually computed per trace variant. To measure the computation time, we took the median out of five repetitions of the computation per variant. Since the number of translucent variants is at least the same as for classic variants, we do not display the computation time per log but the computation time for all variants per log for the different logs. This allows for a fairer comparison. Our results are depicted in Figure 4.

Regarding log variability, we generally observe more translucent variants than classic ones. In general, applying all modifications leads to the highest variability. This is followed by adding events, changing enabled activities, adding enabled activities, and removing enabled activities. Regarding the fitness scores, we denote that the translucent fitness score with no induced noise leads to lower scores than the classic approach. The reason is the enabled activities, which were extracted from a different model. This again highlights the value of considering enabled activities for fitness measurement. As before, applying all noise methods had the worst impact on fitness values. Overall, adding enabled activities had no impact

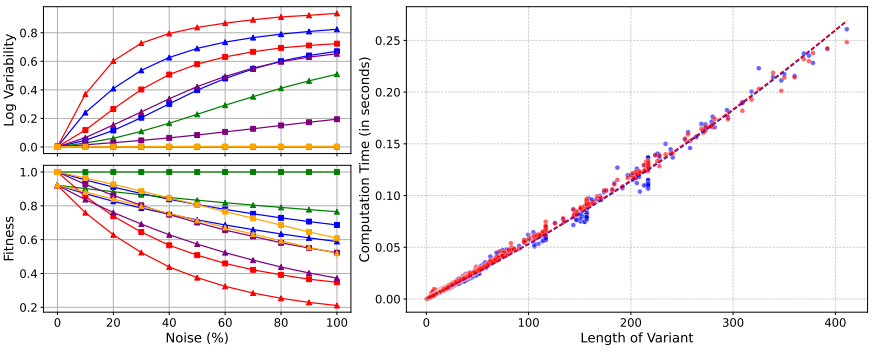
[1] `https://github.com/hherbertb/translucent_alignment`



(a) Results for the road traffic fine management log [26].



(b) Results for the sepsis log [31].



(c) Results for the hospital billing log [32].

Fig. 4: Results of different translucent event logs based on real-life data.

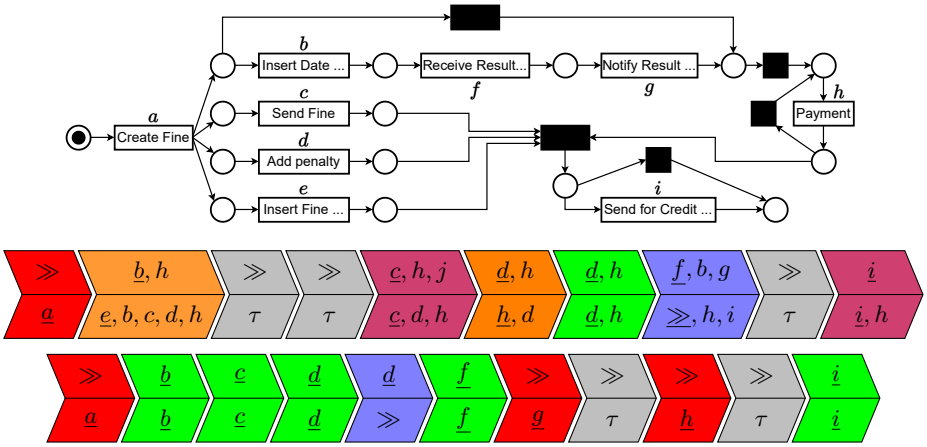


Fig. 5: Petri net with an exemplary translucent alignment of the translucent trace $\langle \underline{b}h, \underline{c}h\underline{j}, \underline{d}h, \underline{d}h, \underline{f}\underline{b}\underline{g}, \underline{i} \rangle$ where j represents the activity "Appeal to Judge" which does not appear in the model. The translucent alignment begins with a model move (red), followed by an execution-enabled-change move (orange, cost of 1.6), and two silent moves (gray). The next two moves are an enabled-change move (purple, cost of 0.5) and an execution-change move (orange, cost of 1). The color saturation represents the cost and, thereby, the degree of changes in the enabled activities. The alignment continues with a synchronous move (green) and a log move (blue) before ending with another silent move (gray) and enabled-change move (purple, cost of 0.5). For comparison, the classic alignment ignoring translucent information is shown underneath.

on the classical view, as expected. For the other methods of inducing noise, the difference in the score exists, up to roughly 0.2. Considering the computation time, we observe that it is nearly quadratic. Moreover, considering enabled activities in the computation has a negligible computational overhead. One of the main reasons is that the Tversky index can be quickly computed. Nonetheless, alignments are usually computed per variant, and since the number of translucent variants is at least the same as classic variants, the computation time for translucent event logs is at least roughly the same as for classic logs.

A translucent alignment from the enhanced road traffic fine management log with all methods and 20 % noise is shown in Figure 5. Enabled-change moves occur, indicating that the model allows for behavior different from the data. With the execution-change move, we may conclude that there is noise in the data. Comparing the optimal translucent alignment with the classic optimal alignment, we observe that the latter contains more synchronous moves. The reason is that it does not account for deviations within the enabled activities. In practice, this information can help determine whether the desktop application works as intended. The translucent alignment has costs of 5.6, the classic alignment costs of 4. Hence, the fitness score of the classic alignment is higher.

6 Conclusion

We provided the first method that considers information on enabled activities in a log and a model when evaluating fitness by introducing three moves that also generalize existing ones. The first deals with situations where the information on enabled activities matches but not the information on executed activities. The second deals with situations where the information on executed activities matches but not the information on enabled activities. The third combines the former two. We can verify system requirements using translucent logs from capturing desktop environments. Our method builds on state-of-the-art techniques and provides holistic scoring. In future work, we plan to extend token-based replay to consider enabled activities, making it suitable for industrial-scale tools due to its faster runtime and relevance to our applications. Additionally, translucent event logs should be considered when assessing generalization and evaluating different models, such as process trees.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.P.: Lucent process models and translucent event logs. Fundam. Informaticae **169**(1-2), 151–177 (2019). https://doi.org/10.3233/FI-2019-1842
3. van der Aalst, W.M.P., Stahl, C.: Modeling Business Processes - A Petri Net-Oriented Approach. MIT Press (2011)
4. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM. pp. 128–135. IEEE (2021). https://doi.org/10.1109/ICPM53251.2021.9576886
5. Adriansyah, A.: Aligning observed and modeled behavior. Phd thesis, Mathematics and Computer Science (2014). https://doi.org/10.6100/IR770080
6. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: BPM - International Workshops. pp. 137–149. Springer (2012). https://doi.org/10.1007/978-3-642-36285-9_15
7. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Comput. Secur. **73**, 172–193 (2018). https://doi.org/10.1016/J.COSE.2017.10.010
8. Berti, A., van der Aalst, W.M.P.: Reviving token-based replay: Increasing speed while improving diagnostics. In: ATAED@Petri Nets/ACSD 2019. pp. 87–103. CEUR-WS.org (2019)
9. Berti, A., van der Aalst, W.M.P.: A novel token-based replay technique to speed up conformance checking and process enhancement. Trans. Petri Nets Other Model. Concurr. **15**, 1–26 (2021). https://doi.org/10.1007/978-3-662-63079-2_1
10. Berti, A., van Zelst, S., Schuster, D.: Pm4py: A process mining library for python. Software Impacts **17**, 100556 (2023). https://doi.org/10.1016/j.simpa.2023.100556
11. Beyel, H.H., van der Aalst, W.M.P.: Creating translucent event logs to improve process discovery. In: ICPM Workshops. pp. 435–447. Springer (2022). https://doi.org/10.1007/978-3-031-27815-0_32

12. Beyel, H.H., van der Aalst, W.M.P.: Improving process discovery using translucent activity relationships. In: BPM. pp. 146–163. Springer (2024). https://doi.org/10.1007/978-3-031-70396-6_9
13. Beyel, H.H., Manuel, S., van der Aalst, W.M.P.: Activitygen: Extracting enabled activities from screenshots. In: ECAI. pp. 712–720. IOS Press (2024). https://doi.org/10.3233/FAIA240553
14. Beyel, H.H., Schwanen, C.T., van der Aalst, W.M.P.: Data for "Translucent Alignments" (2025). https://doi.org/10.5281/zenodo.15210626
15. Beyel, H.H., van der Aalst, W.M.P.: Translucent precision: Exploiting enabling information to evaluate the quality of process models. In: RCIS. pp. 29–37. Springer (2024). https://doi.org/10.1007/978-3-031-59468-7_4
16. vanden Broucke, S.K.L.M., Munoz-Gama, J., Carmona, J., Baesens, B., Vanthienen, J.: Event-based real-time decomposed conformance analysis. In: OTM. Springer (2014). https://doi.org/10.1007/978-3-662-45563-0_20
17. vanden Broucke, S.K.L.M., Weerdt, J.D., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Trans. Knowl. Data Eng. **26**(8), 1877–1889 (2014). https://doi.org/10.1109/TKDE.2013.130
18. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018). https://doi.org/10.1007/978-3-319-99414-7
19. van Dongen, B.F.: Efficiently computing alignments - using the extended marking equation. In: BPM. pp. 197–214. Springer (2018). https://doi.org/10.1007/978-3-319-98648-7_12
20. van Dongen, B.F., Carmona, J., Chatain, T., Taymouri, F.: Aligning modeled and observed behavior: A compromise between computation complexity and quality. In: CAiSE. pp. 94–109. Springer (2017). https://doi.org/10.1007/978-3-319-59536-8_7
21. Dumas, M., Rosa, M.L., Leno, V., Polyvyanyy, A., Maggi, F.M.: Robotic process mining. In: Process Mining Handbook, pp. 468–491. Springer (2022)
22. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - A constructive approach. In: PETRI NETS. pp. 311–329. Springer (2013). https://doi.org/10.1007/978-3-642-38697-8_17
23. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops. pp. 66–78. Springer (2013). https://doi.org/10.1007/978-3-319-06257-0_6
24. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware conformance checking: An entropy-based approach. In: CAiSE. pp. 217–233. Springer (2020). https://doi.org/10.1007/978-3-030-49435-3_14
25. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware precision and recall measures for conformance checking in process mining. Inf. Syst. **115**, 102197 (2023). https://doi.org/10.1016/J.IS.2023.102197
26. de Leoni, M.M., Mannhardt, F.: Road traffic fine management process (2015). https://doi.org/10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5
27. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: BPM. pp. 113–129. Springer (2013). https://doi.org/10.1007/978-3-642-40176-3_10

28. de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and resource-aware conformance checking of business processes. In: BIS. pp. 48–59. Springer (2012). https://doi.org/10.1007/978-3-642-30359-3_5
29. de Leoni, M., Marrella, A.: Aligning real process executions and prescriptive process models through automated planning. Expert Syst. Appl. **82**, 162–183 (2017). https://doi.org/10.1016/J.ESWA.2017.03.047
30. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Object-centric alignments. In: ER. pp. 201–219. Springer (2023). https://doi.org/10.1007/978-3-031-47262-6_11
31. Mannhardt, F.: Sepsis cases - event log (2016). https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
32. Mannhardt, F.: Hospital billing - event log (2017). https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
33. Mannhardt, F., Leemans, S.J.J., Schwanen, C.T., de Leoni, M.: Modelling data-aware stochastic processes - discovery and conformance checking. In: PETRI NETS. pp. 77–98. Springer (2023). https://doi.org/10.1007/978-3-031-33620-1_5
34. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016). https://doi.org/10.1007/S00607-015-0441-1
35. Padró, L., Carmona, J.: Computation of alignments of business processes through relaxation labeling and local optimal search. Inf. Syst. **104**, 101703 (2022). https://doi.org/10.1016/J.IS.2020.101703
36. Reißner, D., Armas-Cervantes, A., Conforti, R., Dumas, M., Fahland, D., Rosa, M.L.: Scalable alignment of process models and event logs: An approach based on automata and s-components. Inf. Syst. **94**, 101561 (2020). https://doi.org/10.1016/J.IS.2020.101561
37. Reißner, D., Conforti, R., Dumas, M., Rosa, M.L., Armas-Cervantes, A.: Scalable conformance checking of business processes. In: OTM. pp. 607–627. Springer (2017). https://doi.org/10.1007/978-3-319-69462-7_38
38. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008). https://doi.org/10.1016/J.IS.2007.07.001
39. Schwanen, C.T., Pakusa, W., van der Aalst, W.M.P.: A dynamic programming approach for alignments on process trees. In: ICPM Workshops (2025). https://doi.org/10.1007/978-3-031-82225-4_7
40. Schwanen, C.T., Pakusa, W., van der Aalst, W.M.P.: Process tree alignments. In: EDOC (2025). https://doi.org/10.1007/978-3-031-78338-8_16
41. Syed, R., Suriadi, S., Adams, M., Bandara, W., Leemans, S.J.J., Ouyang, C., ter Hofstede, A.H.M., van de Weerd, I., Wynn, M.T., Reijers, H.A.: Robotic process automation: Contemporary themes and challenges. Comput. Ind. **115**, 103162 (2020). https://doi.org/10.1016/j.compind.2019.103162
42. Taymouri, F., Carmona, J.: An evolutionary technique to approximate multiple optimal alignments. In: BPM. pp. 215–232. Springer (2018). https://doi.org/10.1007/978-3-319-98648-7_13
43. Taymouri, F., Carmona, J.: Structural computation of alignments of business processes over partial orders. In: ACSD. pp. 73–81. IEEE (2019). https://doi.org/10.1109/ACSD.2019.00012