2021 3rd International Conference on Process Mining (ICPM) | 978-1-6654-3514-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPM53251.2021.9576856

Probabilistic Trace Alignment

Giacomo Bergami*, Fabrizio Maria Maggi*, Marco Montali*, Rafael Peñaloza†
*Free University of Bozen-Bolzano, Bozen, Italy
Email: gibergami@unibz.it, {maggi,montali}@inf.unibz.it
†University of Milano-Bicocca, Milan, Italy
Email: rafael.penaloza@unimib.it

***Abstract*—Alignments provide sophisticated diagnostics that pinpoint deviations in a trace with respect to a process model. Alignment-based approaches for conformance checking have so far used crisp process models as a reference. Recent probabilistic conformance checking approaches check the degree of conformance of an event log as a whole with respect to a stochastic process model, without providing alignments. For the first time, we introduce a conformance checking approach based on trace alignments using stochastic Workflow nets. This requires to handle the two possibly contrasting forces of the cost of the alignment on the one hand and the likelihood of the model trace with respect to which the alignment is computed on the other.**

***Index Terms*—Stochastic Petri nets, Conformance Checking, Alignments.**

I. Introduction

In the existing literature on conformance checking, a common approach is based on trace alignment [1]. This approach uses crisp process models as reference models. Recently developed probabilistic conformance checking approaches provide a numerical quantification of the degree of conformance of an event log with a stochastic process model by either assessing the distribution discrepancies [2], or by exploiting entropy-based measures [3], [4]. These strategies are not based on trace alignments, and hence cannot be readily used to relate an observed trace with the model traces generated by a stochastic process model. In this paper, we provide, for the first time, an approach for the alignment of a trace and a stochastic reference model. This approach is not comparable with the existing literature on probabilistic conformance checking as its output is not numeric but consists of a ranked list of alignments.

With reference to Figure 1, a user might be interested to align the log trace ⟨close order, archive order⟩ with one of the two possible model traces ⟨close order, accept order, pay order, archive order⟩ or ⟨close order, refuse order, archive order⟩. While the latter trace provides the least alignment cost though the model trace has a low probability (0.1), the former gives a slightly greater alignment cost while providing a higher model trace probability (0.9). Since, depending on the context, analysts might prefer either the former or the latter alignment, providing a selection of the best k alignments among all the distinct model traces empowers the analysts to find their own trade-off between alignment cost and model trace probability.

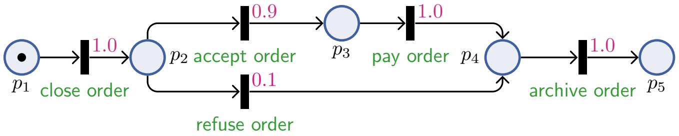


Fig. 1: a simple Stochastic Workflow Net.

To do so, we frame probabilistic trace alignment as a k-Nearest Neighbors (kNN) problem [5], which amounts to find the k nearest data points to a *query* from a set of *data points* via a distance function. We introduce two ranking strategies. The first is based on a brute force approach that reuses existing trace aligners such as [1], [6], where the (optimal) ranking of the top-k alignments is obtained by computing the Levensthein distance of the trace to be aligned to all the model traces. and by multiplying each of these distances by the probability of the corresponding model trace. While this approach returns the best alignment ranking for a query trace, the trace alignments must be computed a-new for all the possible traces to be aligned. For models generating a large number of model traces, this is clearly unfeasible.

To mitigate this, our second strategy produces an approximate ranking where model traces are represented as numerical vectors via embeddings that are independent of the query trace. By exploiting ad-hoc data structures, we can then retrieve the neighborhood of size k containing the traces similar to the given query by pre-ordering (*indexing*) the model traces. Since the embeddings for model traces are independent of the query, this does not require to recompute, for each query, the numeric vector representation of the model traces.

We implemented both strategies, and use a real event log coming from a hospital system to empirically evaluate their properties. Our experiments assess the two strategies as follows: *(i)* first, we evaluate the degree of approximation introduced by the approximate-ranking approach when compared with optimal-ranking; *(ii)* we evaluate the computation time needed by the two strategies to produce the top-k alignments. We observe that, by properly selecting the data structure used to represent the search space, approximate-ranking alignments provide the best trade-off between accuracy and efficiency.

II. Related Work

Recent works on probabilistic conformance checking assess the degree of conformance of a Stochastic Petri net against

This research has been supported by the project IDEE FESR1133 funded by the Eur. Reg. Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020, and by the UNIBZ projects VERBA and CAT.

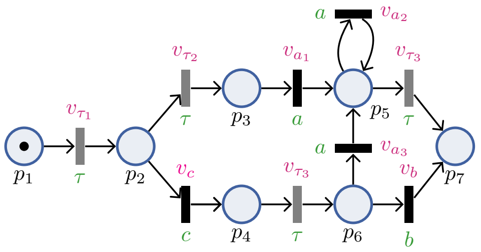
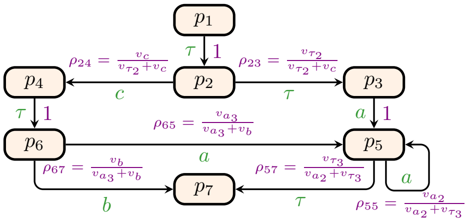


Fig. 2: A sample SWN N. Labels are in green, τ transitions in grey, weights in magenta.



p_1
τ
1
$\rho_{24} = \frac{v_c}{v_{\tau_2}+v_c}$
$\rho_{23} = \frac{v_{\tau_2}}{v_{\tau_2}+v_c}$
p_4
p_2
p_3
c
τ
τ
1
a
1
$\rho_{65} = \frac{v_{a_3}}{v_{a_3}+v_b}$
p_6
p_5
a
$\rho_{67} = \frac{v_b}{v_{a_3}+v_b}$
$\rho_{57} = \frac{v_{\tau_3}}{v_{a_2}+v_{\tau_3}}$
a
p_7
b
τ
$\rho_{55} = \frac{v_{a_2}}{v_{a_2}+v_{\tau_3}}$

Fig. 3: Reachability graph of N. Probabilities are in violet.

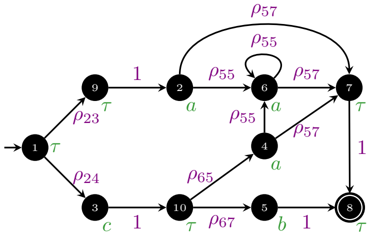


Fig. 4: Preliminary Transition graph encoding N with no τ-closures.

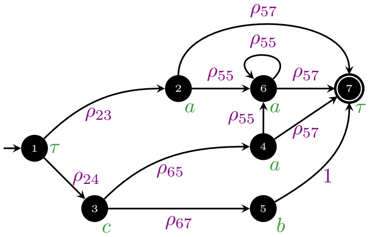


Fig. 5: Transition graph resulting from N after τ-closure.

either one single log trace [3], [4] or against an entire log [7]. The former approaches might be used to rank different stochastic models according to their degree of conformance with respect to a fixed log trace, while our proposed solution ranks a subset of the model traces of the same stochastic model with respect to a given log trace. Albeit the input of such approaches is the same as ours, the problem that we intend to solve is different. Nevertheless, we might exploit our solution to rank stochastic models via the best trace alignment provided by each single model. Furthermore, approaches evaluating the conformance of a stochastic model against a whole log cannot be exploited for aligning a stochastic model and a single log trace, since, apart from the log trace to be aligned, the remaining log traces are not necessarily known, and therefore it is impossible to “earth-move” the probability distribution of a stochastic model towards a set of unknown traces.

A work that combines probabilistic conformance checking and alignments and that is closer to the one described in this paper is [8]. Their approach is based on standard (not stochastic) Petri nets and extends the alignment cost functions by considering the probabilities of activities to be added/deleted when a log trace and a model trace do not match. Such functions always return a zero alignment cost on matching traces, independently of the probability associated to the model trace. Hence, the resulting ranking cannot be used to provide a trade-off between trace probability and alignment cost.

III. Modeling Probabilistic Dynamic Systems

We recall the models and techniques at the basis for representing and computing probabilistic trace alignments.

A. Stochastic Workflow Nets

As customary in probabilistic conformance checking [2]–[4], we adopt stochastic Petri nets [9], [10] to represent processes. More specifically, we consider an interesting class of stochastic Petri nets with immediate (not timed) transitions only, namely untimed Stochastic Workflow Nets (SWNs). We fix a set $\Sigma = \mathcal{A} \cup \{ \tau \}$ of labels, where labels in $\mathcal{A}$ indicate process tasks, whereas τ indicates an invisible execution step (τ-transition). A *trace* is a finite sequence of labels from $\mathcal{A}$.

Definition 1. *An* untimed Stochastic Workflow Net (SWN) *is a tuple $N = (P, T, F, \ell, W)$ where: (i) (P, T, F) is a standard* Workflow net *with places P, transitions T, and flow relation F such that there is exactly one* input place *with no incoming arc, and exactly one* output place *with no outgoing arcs; (ii) $\ell : T \to \Sigma$ is a* labeling function *mapping each transition $t \in T$ into a label $\ell(t) \in \Sigma$ (if $\ell(t) = \tau$, then t is a silent transition); (iii) $W : T \to \mathbb{R}^+$ is a* weight function *assigning a positive firing weight to each transition of the net.* ◁

For an SWN N, we use dot notation to get its components (e.g., $N.P$ denotes its places). *We do the same for the other structures introduced in the paper*. We use $N.in$ and $N.out$ to respectively denote the input and output places of N.

As usual, the current state of execution for N is captured as a marking of the net: a multiset over places $N.P$ indicating how many tokens populate each place. The notions of transition enablement and firing are also standard. Given a marking m over SWN N, we denote by $E_N(m)$ the set of enabled transitions in m; given transition $t \in E_N(m)$, we write $m \xrightarrow{t}_N m'$ to capture that, within N, firing t in m results in the new marking m'. A *firing sequence of N starting from marking* m_0 is a sequence $t_1 \cdots t_n$ of transitions from $N.T$ so that, for every $i \in \{1, \ldots, n\}$, we have that $m_{i-1} \xrightarrow{t_i}_N m_i$. We say that the firing sequence results in m_n.

As customary in Workflow nets, we consider two special markings: the *input* (resp. *output*) marking m_{in}^N (resp. m_{out}^N) that assigns a single token to the input (resp. output) place $N.in$ (resp. $N.out$) of N, and no token elsewhere. A *valid* *sequence* $\eta = t_1 \cdots t_n$ of N is a firing sequence of N starting from m_{in}^N and resulting in m_{out}^N. A sequence $\xi = \alpha_1 \cdots \alpha_n$ of labels from Σ is a *run* of N if there exists a valid underlying sequence $\eta = t_1 \cdots t_n$ of N such that, for every $i \in \{1, \ldots, n\}$, we have $N.\ell(t_i) = \alpha_i$. Run ξ may have different underlying valid sequences in N, which we collectively refer to as $seqs_N(\xi)$. A trace σ is a *model trace* of N (or N-trace for short) if there exists an underlying run ξ of N that corresponds to σ once all occurrences of τ are removed. There may be multiple runs underlying an N-trace σ, and we collectively refer to them as $runs_N(\sigma)$. Finally, we denote the (possibly infinite) set of N-traces as $traces(N)$.

For an SWN, the key change to the standard execution semantics is that, being the net stochastic, the set of enabled transitions in a marking is associated to a discrete probability distribution. This is defined using the weight function of N: given marking m of N and an enabled transition $t \in E_N(m)$, the *firing probability* of t in m is $\mathbb{P}_{m,N}(t) = \frac{N.W(t)}{\sum_{t' \in E_N(m)} N.W(t')}$. We use this to define the probability $\mathbb{P}_N(\eta)$ of a valid sequence $\eta = t_1 \cdots t_n$ of N as the product of the probabilities associated to each transition: $\mathbb{P}_N(\eta) = \prod_{i \in \{1,\ldots,n\}} \mathbb{P}_{m_{i-1},N}(t_i)$. For a trace σ of N, its probability $\mathbb{P}_N(\sigma)$ is then obtained by collecting all its underlying runs, in turn collecting all their underlying valid sequences, and summing up their respective probabilities: $\mathbb{P}_N(\sigma) = \sum_{\xi \in runs_N(\sigma)} \sum_{\eta \in seqs_N(\xi)} \mathbb{P}_N(\eta)$. This captures that, to observe σ, one can equivalently pick any of its underlying valid sequences. Notably, if a trace is not an N-trace (i.e., it does not conform with N), then its probability is 0. For convenience, when needed, we represent an N-trace as a pair $\langle \sigma, \mathbb{P}_N(\sigma) \rangle$.

Remark 1. The sum of the probabilities of all runs of an SWN, up to a certain maximum length n, is between 0 and 1. When n tends to ∞, this sum tends to 1. This is a direct consequence of how transition probabilities are computed, paired with the fact that runs of a Workflow net are maximal.$\triangleleft$

By interpreting concurrency by interleaving, firing sequences and their probabilities are compactly represented in a reachability graph.

Definition 2. *The* Reachability Graph $RG(N)$ *of SWN* N*is a triple* (M, E, P) *where: (i)* M *is the set of reachable markings from* m_0^N *(*m_0^N *included); (ii)* $E \subseteq M \times \Sigma \times M$ *is a* Σ-labeled transition relation *induced by* N*, that is, for* $m, m' \in M$*, we have edge* $(m, a, m') \in E$ *if and only if there exists transition* t *in* N *with label* $\ell(t) = a$ *and such that* $m \xrightarrow{t}_N m'$*; (iii)* $P : E \to [0, 1]$ *is the* transition probability *function assigning to each transition* $(m, a, m') \in E$ *its probability, obtained from the firing probability of the SWN transition(s) that lead from* m *to* m' *and are labeled by* a*:* $P(m, a, m') = \sum_{t_i \in E_N(m) \text{ s.t. } N.\ell(t)=a \text{ and } m \xrightarrow{t}_N m'} \mathbb{P}_{m,N}(t)$. $\triangleleft$

The definition also handles the special case where, in a given marking, distinct net transitions with the same label produce the same consequent marking: they are indistinguishable when observing the execution traces of the net, hence they collapse into a single edge of the reachability graph, where all firing probabilities are aggregated into a single value. Fig. 3 shows an example of a reachability graph.

Every SWN N handled in our framework is assumed to satisfy two natural properties:

1) N is *bounded*, that is, every marking in $RG(N)$ assigns at most a pre-defined number of tokens to each place;
2) no loop in $RG(N)$ has all edges labeled by τ.

Property 1) states that process instances of N do not generate unboundedly many concurrent threads, i.e., that $RG(N)$ has finitely many states. Property 2) naturally corresponds to how τ-transitions are used when modeling processes: they are essential to model gateways (such as exclusive and parallel splits/joins), cascaded gateways without tasks in between, and skippable tasks; such constructs require τ-transitions, but never used in fully invisible loops.[1] Property 2) implies a very interesting property: given a trace σ, there are only boundedly many valid sequences that can produce it. Hence, the probability of σ can be computed by: *(i)* exhaustively enumerating all its valid sequences; *(ii)* calculating the probability of each such sequence; *(iii)* summing up all the so-obtained probabilities.

Remark 2 (From [11]). For an SWN N with at most b consecutive τ transitions, there are boundedly many runs of N yielding a given N-trace σ: $seqs_N(\sigma)$ contains runs whose maximum length is bounded by the length of σ and b. $\triangleleft$

By combining Remarks 1 and 2, we get a direct way of computing the trace probability $\mathbb{P}_N(\sigma)$. To handle probabilistic trace alignment, we need to relate an arbitrary log trace σ' over $\mathcal{A}^*$ with the closest N-traces that balance their distance from σ' and their probability. Fortunately:

Remark 3. By increasing the length of the N-traces, we reach a point where their probability and distance with respect to any log trace σ' *both* decrease. Intuitively, this is because executing too many loop iterations within N at once decreases the overall run probability and increments the distance from σ'. $\triangleleft$

This implies that probabilistic trace alignment operates over a finite space of traces/runs, hence being a combinatorial problem that can be tackled with techniques such as kNN.

B. Transition Graphs

The graph and trace embedding techniques at the core of our approach cannot be directly defined over reachability graphs: they rely on graphs where edges are decorated by probabilities, and where labels are attached to nodes. In addition, to enable efficient algorithmic techniques, such graphs are compactly defined via transition matrixes. We hence take inspiration from [12] and introduce *probabilistic transition graphs*, used later to encode SWNs via their reachability graphs.

Definition 3. *A* (Probabilistic) Transition Graph *is a tuple* (V, s, t, L, R) *where: (i)* $V \subset \mathbb{N}$ *is a set of* nodes*; (ii)* $s \in V$ *is the* initial node*; (iii)* $e \in V$ *is the* accepting node*; (iv)* $L : \Sigma \times V \to \{0, 1\}$ *is a* label matrix *associating each*

[1] A more thorough discussion on this can be found in [11].

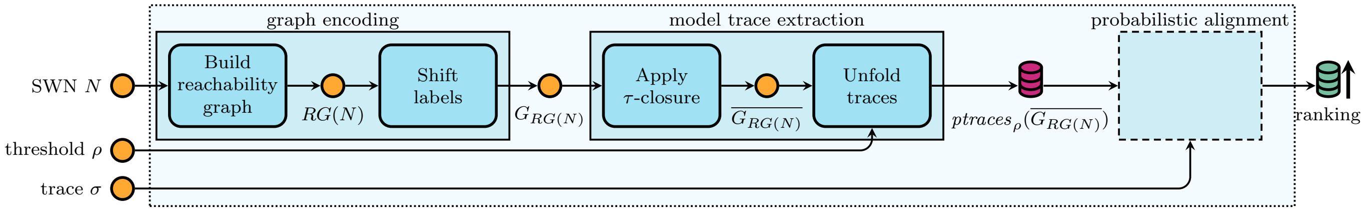


Fig. 6: Proposed pipeline to assess the probabilistic trace alignment.

node in V to a single label in Σ, where for label $\alpha \in \Sigma$ and node $\mathtt{i} \in V$, $[L]_{\alpha\mathtt{i}}$ gives 1 *if* i *is labeled by* α, 0 *otherwise; (v) $R : V \times V \rightarrow [0,1]$ is a* (probabilistic) transition matrix *indicating, for each pair of nodes, the probability of executing a transition from the first node leads to the second node. L and R satisfy the following well-formedness conditions: (i) for every $i \in V$ there is one and only one label $\alpha \in \Sigma$ so that $[L]_{\alpha\mathtt{i}} = 1$; (ii) for every $\mathtt{i} \in V$, $\sum_{\mathtt{j}\in V}[R]_{\mathtt{ij}} = 1$.* ◁

The condition for L indicates that each node is mapped by L to a single label, while the same label may be used for multiple nodes. The condition for R ensures that the values contained therein can be interpreted as a probability distribution when choosing which next node to pick upon executing a transition. Matrices L and R can be exploited to determine the probability of reaching a node labeled by $\beta \in \Sigma$ from any node labeled $\alpha \in \Sigma$ in n steps with $[LR^nL^\top]_{\alpha\beta}/[LL^\top]_{\alpha\alpha}$ that we shorthand as $[G.\Lambda^n]_{\alpha\beta}$ [12].

A transition graph G can be visualized as shown in Fig. 4 (Fig. 5 after τ-closure, see below). There, the elements have the obvious interpretation, with the only important consideration that an edge from node i to node j is only depicted if the transition probability $[G.R]_{\mathtt{ij}}$ is positive.

We mirror the definitions of SWNs considering that now labels are on nodes. A *valid sequence* of G is a sequence $\mathtt{i}_0 \dots \mathtt{i}_n$ of nodes in $G.V$ from the initial to the accepting node that only traverses transitions with nonzero probability: *(i)* $\mathtt{i}_0 = G.s$; *(ii)* $\mathtt{i}_n = G.e$; *(iii)* if the sequence contains at least two nodes, each two consecutive nodes are connected by a positive transition probability, i.e., for every $j \in \{1, \dots, n\}$ we have $[R]_{\mathtt{i}_{j-1}\mathtt{i}_j} > 0$. Runs and model traces of transition graphs are then defined as in SWNs, and we employ the same notation to indicate the runs underlying a model trace, and the valid sequences underlying a run. The computation of probabilities for runs and traces is hence defined equivalently.

C. Graph and String Kernels

As a foundational basis to compute trace alignments, we adapt similarity measures from the database literature. Given a set of data examples $\mathcal{X}$, (e.g., strings or traces, transition graphs) a (positive definite) *kernel* function $k\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ denotes the similarity of elements in $\mathcal{X}$. If $\mathcal{X}$ is the d-dimensional Euclidean space $\mathbb{R}^d$, the simplest kernel function is the inner product $\langle \mathbf{x}, \mathbf{x}' \rangle = \sum_{1\leq i\leq d} \mathbf{x}_i\mathbf{x}'_i$. A kernel is said to *perform ideally* [13] when $k(x, x') = 1$ whenever x and x' are the same object (*strong equality*) and $k(x, x') = 0$ whenever x and x' are distinct objects (*strong dissimilarity*). A kernel is also said to be *appropriate* when similar elements $x, x' \in \mathcal{X}$ are also close in the feature space. Notice that appropriateness can only be assessed empirically [13]. A positive definite kernel induces a distance metric as $d_k(\mathbf{x}, \mathbf{x}') := \sqrt{k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{x}') + k(\mathbf{x}', \mathbf{x}')}$. When the kernel of choice is the inner product, the resulting distance is the Euclidean distance $\|\mathbf{x} - \mathbf{x}'\|_2$. A normalized vector $\hat{\mathbf{x}}$ is defined as $\mathbf{x}/\|\mathbf{x}\|_2$. For a normalized vector, we can easily prove that: $\|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|_2^2 = 2(1 - \langle \hat{\mathbf{x}}, \hat{\mathbf{x}}' \rangle)$. When $\mathcal{X}$ does not represent directly the d-dimensional Euclidean space $\mathbb{R}^d$, we can use an *embedding* $\phi\colon \mathcal{X} \rightarrow \mathbb{R}^d$ to define a kernel $k_\phi\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ as $k_\phi(x, x') := \langle \phi(x), \phi(x') \rangle$.

The literature also provides a kernel representation for strings [12], which we can directly employ for traces. Specifically, if we associate each dimension in $\mathbb{R}^d$ to a different substring $\alpha\beta$ of size 2 (i.e., 2-grams[2]), the embedding represents how frequently and "compactly" this sub-trace is embedded in the trace σ' of interest. Therefore, we introduce a *decay factor* $\lambda \in [0,1] \subseteq \mathbb{R}$ that, for all m sub-strings where α and β appear in σ' at the same relative distance $z < |\sigma'|$, weights the resulting embedding as $\lambda^z m$.

Example 1. Consider tasks $\mathcal{A} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$. The 2-grams over $\mathcal{A}$ are $\mathcal{A}^2 = \{\mathsf{aa}, \mathsf{ab}, \mathsf{ac}, \mathsf{ba}, \mathsf{bb}, \mathsf{bc}, \mathsf{ca}, \mathsf{cb}, \mathsf{cc}\}$. TABLE I shows the embeddings of some traces. Being a 2-gram, trace cb has only one nonzero component, corresponding to itself, with ${\phi^{\mathrm{tr}}}_{\mathsf{cb}}(\mathsf{cb}) = \lambda$. Trace caa has the 2-gram ca occurring with length 1 ($\underline{\mathsf{ca}}\mathsf{a}$) and 2 ($\underline{\mathsf{c}}\mathsf{a}\underline{\mathsf{a}}$), and the 2-gram aa occurring with length 1 ($\mathsf{c}\underline{\mathsf{aa}}$). Hence: ${\phi^{\mathrm{tr}}}_{\mathsf{ca}}(\mathsf{caa}) = \lambda + \lambda^2$ and ${\phi^{\mathrm{tr}}}_{\mathsf{aa}}(\mathsf{caa}) = \lambda$. Similar considerations apply for the other traces. We now want to compute the similarity between trace caba and the other two traces. To do so, we sum, column-wise (that is, 2-gram by 2-gram) the product of the embeddings for each pair of traces. We then get $k_{\phi^{\mathrm{tr}}}(\mathsf{caba}, \mathsf{caa}) = \lambda^3 + (\lambda + \lambda^3)(\lambda + \lambda^2)$ and $k_{\phi^{\mathrm{tr}}}(\mathsf{caba}, \mathsf{cb}) = \lambda^3$, which induces the ranking $k_{\phi^{\mathrm{tr}}}(\mathsf{caba}, \mathsf{caa}) > k_{\phi^{\mathrm{tr}}}(\mathsf{caba}, \mathsf{cb})$. ◁

This trace kernel returns strong dissimilarity when the two traces have no shared 2-grams at any arbitrary length, but does not enjoy strong equality: the similarity of a trace with itself is at least λ^2 (returned when the trace is a 2-gram).

[2] For our experiments, we choose to consider only 2-grams but any p-grams of arbitrary length $p \geq 2$ might be adopted [13]. An increased size of p improves precision but also incurs in a worse computational complexity, as it requires to consider all the arbitrary sub-traces of length p whose constitutive elements occur at any distance from each other within the trace.

TABLE I: Embedding of traces caba, caa and cb.

	aa	ab	ac	ba	bb	bc	ca	cb	cc
caba	λ^2	λ	0	λ	0	0	$\lambda+\lambda^3$	λ^2	0
caa	λ	0	0	0	0	0	$\lambda+\lambda^2$	0	0
cb	0	0	0	0	0	0	0	λ	0

IV. Probabilistic Trace Alignment Pipeline

Our approach takes as input *(i)* a reference model represented as an SWN N, *(ii)* a minimum, positive probability threshold $\rho \in (0,1]$ *(iii)* a trace σ' of interest, and returns a ranking over all the N-traces having a probability $\geq \rho$, combining their probability values (probabilistic component) and their distance to σ' (alignment component).

A. Computation pipeline

The approach is realized through the pipeline shown in Fig. 6, consisting of the following five steps. In step 1, the reachability graph $RG(N)$ of N is constructed. In step 2, $RG(N)$ is converted into a corresponding transition graph $G_{RG(N)}$ that preserves model traces and their probabilities. We omit the details of this conversion, due to lack of space and the fact that it employs well-known techniques used to *shift labels* from transitions to states while preserving the behavior encoded by a transition system. Fig. 4 shows the result of this conversion when applied to the reachability graph in Fig. 3.

In step 3, the transition graph $G_{RG(N)}$ is processed applying a τ*-closure* that compiles away τ-transitions. This results into a new transition graph $\overline{G_{RG(N)}}$ that retains τ labels only in the initial and accepting states and for the rest exclusively employs visible labels in $\mathcal{A}$, while preserving model traces and their probabilities. Also in this case we omit the details: the transformation relies on well-known automata-based techniques for removing ϵ-moves. The only non-trivial observation is that, even in our case where probabilities are present, all τ transitions can still be removed thanks to the working hypothesis done for SWNs in Sect. III-A, namely absence of fully silent loops. As a result of this step, the transition graph in Fig. 4 results in that of Fig. 5.

In step 4, the τ-closed transition graph $\overline{G_{RG(N)}}$ is unfolded, so as to collect all the model traces that have a probability of at least ρ. To do so, we rely on the properties in Remarks 2 and 3 from Sect. III, which are inherited by $\overline{G_{RG(N)}}$. Specifically, they imply that no loop can be executed without strictly decreasing the resulting probability. This means that all valid sequences with a resulting probability of at least ρ can be enumerated and returned in a set. The so-obtained sequences are combined by merging those that produce the same trace, summing up their probabilities, thus obtaining the set $ptraces_\rho(\overline{G_{RG(N)}})$ of all the traces with probability $\geq \rho$. The closure operation also implies that the notion of model trace coincides with that of run, modulo removing the two τ labels attached to the initial and accepting nodes.

The last step takes the so-obtained model traces and ranks them by considering their probabilities and the similarity with the log trace σ' of interest. In Fig. 6, this is shown as a black-box. As we describe next, we have implemented this last step in two alternative ways: one computationally demanding but guaranteeing an optimal-ranking, the other more efficient but providing approximate ranking without optimality guarantees.

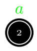
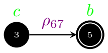
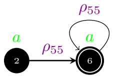
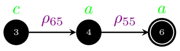
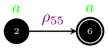
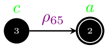
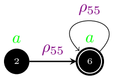
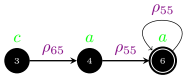
B. Alignment Strategies

Upon aligning an event log with a stochastic net, distinct model traces have different probabilities. Hence the retrieval of the best model traces maximizing the combined provision of minimum trace alignment cost and maximum model trace probability might not suffice. In some cases, the user could favor an alignment with a lower cost even if based on a less probable model trace, while, in other cases, they may prefer a model trace with a higher probability at the expense of a higher alignment cost. Hence, we align a log trace with a transition graph by retrieving the best k alignments among all model traces in $ptraces_\rho(N)$. This reduces to the kNN problem by finding the k nearest data points to a *query* x from a set $\mathcal{X}$ of *data points* w.r.t. a given distance function d_k.

Optimal-Ranking Trace Aligner. Here we reuse existing trace aligners such as [1], [6], where $d(\sigma',\sigma)$ is the Levenshtein distance. We express the ranking score as the product $\mathbb{P}_G(\sigma)d(\sigma',\sigma)$, considering the cost of the alignment (i.e., the distance between the model trace and the trace to be aligned) and the probability of the model trace. To represent the same intuition of such a weighted distance as a ranking function, we transform it into a similarity function returning 1 when $\sigma=\sigma'$ and $\mathbb{P}_G(\sigma')=1$ hold. We then express d as a normalized similarity score $s_d(\sigma',\sigma) := \frac{1}{d(\sigma',\sigma)+1}$. The maximum similarity is reached when the distance is 0 and the similarity decreases while the distance increases. The golden ranking function (i.e., the one producing the optimal-ranking) can therefore be represented as $\mathcal{R}(\sigma',\sigma) = \mathbb{P}_G(\sigma)\mathbb{P}_G(\sigma')s_d(\sigma',\sigma)$. The computation $\max\arg_{\sigma\in ptraces_\rho(G)}\mathcal{R}(\sigma',\sigma)$ returns the best optimal-ranking trace alignment for a log trace σ', where $\mathcal{R}$ must be computed a-new for all the possible σ'.

Approximate-Ranking Trace Embedder. Ranking optimality comes at the sub-optimal cost of a brute-force recomputation of $\mathcal{R}$ for each novel trace σ' to align. Since each embedding ϕ entails an associated similarity metric k_ϕ and hence an associated distance d_{k_ϕ} (Sect. III-C), we compute the embeddings for all the unfolded traces before performing the top-k search ensuring that they are independent of the trace to align, avoiding the brute-force cost. This computation gain comes with a loss in precision: the generation of precise embeddings for graph data with loops is NP-Complete [12] and, in its approximated

TABLE II: Projections over N-traces of length 4.

σ	G_σ	l	ω
a		1	$\rho_{23}\rho_{57}$
cb		2	ρ_{24}
aaa		3	$\rho_{23}\rho_{57}$
caa	 c a a ρ_{65} ρ_{55} 3 4 6	3	$\rho_{24}\rho_{57}$
aa		2	$\rho_{23}\rho_{57}$
ca		2	$\rho_{24}\rho_{57}$
aaaa		4	$\rho_{23}\rho_{57}$
caaa	 ρ_{55} c a a ρ_{65} ρ_{55} 3 4 6	4	$\rho_{24}\rho_{57}$

version, does not accurately represent the data using low-dimensional vectors [14]. So, our proposed embedding (that we indicate with ϕ^g) is weakly-ideal (Sect. III-C and [13]).

To obtain ϕ^g, we adapt the embedding strategy ϕ^{tr} from [15] by addressing some shortcomings of such a strategy: **(a)** it does not perform weakly-ideally, so we cannot numerically assess if two embeddings represent equivalent traces (Example 1); **(b)** it does not characterize τ-moves, so the probabilities of the initial and final τ-moves are not preserved; **(c)** it is affected by numerical errors from finite arithmetics: longer traces σ generated from skewed probability distributions $\overline{G_{RG(N)}}.\Lambda^i$ suffer from greater truncation errors, as smaller λ^i components for bigger $i < |\sigma|$ will be ignored, preventing a complete numerical vector characterization of σ in practice.

To overcome these shortcomings we **(a)** propose a weakly-ideal embedding, which also **(b)** exploits an ω factor for preserving probabilities from and to τ transitions. We also **(c)** mitigate the numerical truncation errors induced by trace length and probability distribution skewness through two sub-embedding strategies, ϵ^x and ν^x, where the former descends from ϕ^{tr} and the latter approximates the trace similarity via label frequencies similarity. Since a trace embedding for $\sigma \in ptraces_\rho(G)$ adequately representing the transitions in $\overline{G_{RG(N)}}.\Lambda$ requires an intermediate G representation, we map each σ to a transition graph G_σ as follows:

Definition 4 ($\overline{G}$ projection over traces). *Given a minimum probability threshold ρ and a τ-closed transition graph $\overline{G} = (V, s, t, L, R)$, for each trace $\sigma \in ptraces_\rho(\overline{G})$, the $\overline{G}$ projection over σ is a pair $(\overline{G}_\sigma, \omega)$, where $\overline{G}_\sigma$ is a transition graph such that (i) $\overline{G}_\sigma.V$ contains all distinct nodes generating σ from $\overline{G}$ (i.e., $\bigcup_{\xi \in runs_{\overline{G}}(\sigma)} seqs_{\overline{G}}(\xi)$); (ii) $\overline{G}_\sigma.s = s$; (iii) $\overline{G}_\sigma.t = t$; (iv) $\overline{G}_\sigma.L$ (and $\overline{G}_\sigma.R$) is the submatrix of L (and R) over the columns (and rows) in $\overline{G}_\sigma.V$, and $\omega \in [0, 1]$ is a* graph weight *preserving the transition probabilities from and to τ nodes and computed as $1 - \prod_{\xi \in runs_{\overline{G}}(\sigma), \eta \in seqs_{\overline{G}}(\xi)} \Big(1 - (ifte([L]_{\tau\eta_1}, [R]_{\eta_1\eta_2})$ $ifte([L]_{\tau t_n}, [R]_{t_{n-1}t_n})\Big)$, where $ifte(x, y) := x(y-1)+1$ returns y if $x = 1$ and 1 otherwise. We denote the set of all the $\overline{G}_\sigma$ as $\mathbf{G}_\rho(\overline{G})$.* ◁

TABLE III: Different sub-embedding definitions (ϵ^1, ϵ^2, ν^1, and ν^2) for ϕ^g.

	$x = 1$	$x = 2$
$\epsilon^x_{\mathsf{ab}}(\overline{G}_\sigma) :=$	$\sum_{i=1}^{l} \lambda^i \frac{[LR^iL^t]_{\mathsf{ab}}}{\sum_{\mathsf{a'b'} \in \mathcal{A}^2} R^i_{\mathsf{a'b'}}}$	$\sum_{i=1}^{l} \lambda^i [\Lambda^i]_{\mathsf{ab}}$
$\nu^x_{\mathsf{a}}(\overline{G}_\sigma) :=$	$\frac{1}{c} \sum_{\sigma' \in ptraces_0(\overline{G}_\sigma)} \frac{\lvert\{\sigma'_i \in \sigma' \mid \mathsf{a} \in \mathcal{A} \wedge \sigma'_i = \mathsf{a}\}\rvert}{\lvert\sigma'\rvert}$	0

The graph weight ω derives from the outgoing edges of the initial node and the ingoing edges of the accepting node when such nodes are labeled as τ. Given that (i) the embedding strategy from [15] allows trace embedding only for visible (i.e., non-τ) transitions, and (ii) the trace extraction process discards the τ information, we use ω to preserve such information.

Example 2. Given the τ-closed transition graph $\overline{G_{RG(N)}}$ in Fig. 5, we assign the probability values $\rho_{23} = 0.8$, $\rho_{24} = 0.2$, $\rho_{55} = \rho_{57} = 0.5$, $\rho_{65} = 0.7$, and $\rho_{67} = 0.3$. The $ptraces_0(\overline{G_{RG(N)}})$ with maximum length 4 are: $\{\langle \mathsf{a}, 0.4\rangle, \langle \mathsf{aa}, 0.2\rangle, \langle \mathsf{aaa}, 0.1\rangle, \langle \mathsf{ca}, 0.07\rangle, \langle \mathsf{cb}, 0.06\rangle, \langle \mathsf{aaaa}, 0.05\rangle, \langle \mathsf{caa}, 0.035\rangle, \langle \mathsf{caaa}, 0.0175\rangle\}$. TABLE II shows the projected transition graphs associated to such traces, where only the relevant information for embedding them is displayed (e.g., all the τ-labeled nodes are removed). ◁

Our proposed embedding ϕ^g is computed for each transition graph generated in the former definition. The goal is to use k_{ϕ^g} for ranking all the traces generated by unfolding via such graphs. We extend the embedding ϕ^{tr} from [15] by including the traces associated probability, and making the ranking induced by k_{ϕ^g} the inverse of the ranking induced by the sum of the following distances: the transition correlations ϵ and the transition label frequency ν. We also require that the desired properties of ϕ^g are independent of the characterization of ϵ over the 2-grams in $\mathcal{A}^2$ and ν over the labels in $\mathcal{A}$, which provide different embedding strategies. Therefore, our proposed ϕ^g embedding is defined as follows:

Definition 5 (G-Embedding). *Given a $\overline{G}$ projection over σ $(\overline{G}_\sigma, \omega)$ and a tuning parameter $t_f \in [0, 1] \subseteq \mathbb{R}^+_0$, the* G-Embedding *$\phi^g$ over the visible 2-grams and transition labels, $\mathcal{A} \cup \mathcal{A}^2$, is defined by*

$$\phi^g{}_i(\overline{G}_\sigma) = \begin{cases} \omega \frac{\epsilon_{\mathsf{ab}}(\overline{G}_\sigma)}{\|\epsilon\|_2} t_f^{|R>0|} & i = \mathsf{ab} \\ \frac{\nu_{\mathsf{a}}(\overline{G}_\sigma)}{\|\nu\|_2} t_f^{|R>0|} & i = \mathsf{a} \end{cases}$$

where ν (and ϵ) represents the non-negatively defined embeddings associated to $\overline{G}_\sigma.L$ (both $\overline{G}_\sigma.R$ and $\overline{G}_\sigma.L$). ◁

Here, $\max\arg_{\sigma \in ptraces_\rho(G), G_\sigma \in \mathbf{G}_p(P)} k_{\phi^g}(G'_\sigma, G_\sigma)$ returns the best approximated trace alignment for a log trace represented as $G_{\sigma'}$. For sub-embeddings ϵ and ν, in our experiment section, we choose two possible interchangeable definitions

TABLE IV: Distinct SWNs and associated sets of unfolded traces discovered from the Sepsis Cases event log.

Experiment Conf. ($\mathcal{U}$)	*Model*	$+$*W. Estimator*	ρ	$\lvert ptraces_\rho(G)(P_\mathcal{U})\rvert$
SM_CONS_20	SplitMiner 2.0 [17]	+`Constant`	0	157
SM_FORK_20	SplitMiner 2.0 [17]	+Fork [18]	0	32
SM_PAIR_20	SplitMiner 2.0 [17]	+PairScale [18]	0	157
STPETRI_20	Rogge-Solti [10]		10^{-5}	1612

($x = 1$ and $x = 2$) shown in TABLE III: here, l is the path length (reported in TABLE II), and c for ν^1 is a normalization factor such that $\sum_{\mathsf{a}\in\mathcal{A}} \nu^1_{\mathsf{a}}(P) = 1$. While ν^2 implies to completely ignore the label frequency contribution, ϵ^2 is the direct implementation of ϕ^{tr} from Sect. III-C, and ϵ^1 and ϵ^2 only differ from the normalization perspective.

$t_f \in [0,1] \subseteq \mathbb{R}^+_0$ and $\lambda \in [0,1] \subset \mathbb{R}^+_0$ are tuning parameters that can be inferred from the available data [16]. The latter describes the previously mentioned decay factor, while t_f represents the relevance of our embedding representation as the number of edges within $\overline{G}_\sigma$ increases. In our experiments and examples, we choose $t_f = 0.0001$ and $\lambda = 0.07$. This representation is independent of the representation associated with a trace to be aligned. Therefore it does not have to be recomputed at each alignment with a different σ'.

The kernel k_{ϕ^g} associated to ϕ^g is a function of the distance $\|\hat{\epsilon} - \hat{\epsilon}'\|_2^2$ and $\|\hat{\nu} - \hat{\nu}'\|_2^2$ for traces σ' and σ:

Proposition 1. *Given* $(\overline{G}_\sigma, \omega)$ *and* $(\overline{G}_\sigma, \omega')$ *with* $\overline{G}_\sigma = (s, t, L, R)$ *and* $\overline{G}_\sigma = (s', t', L', R')$*, the definition for* k_{ϕ^g} *is expanded as follows:*

$$k_{\phi^g}(\overline{G}_\sigma, \overline{G}_\sigma) = \omega\omega' t_f^{|R>0|+|R'>0|}\left(1 - \frac{\|\hat{\epsilon} - \hat{\epsilon}'\|_2^2}{2}\right) + \\ + t_f^{|R>0|+|R'>0|}\left(1 - \frac{\|\hat{\nu} - \hat{\nu}'\|_2^2}{2}\right)$$

Proof. By definition of k_ϕ as a vector dot product for any embedding ϕ and by $\|\mathbf{x} - \hat{\mathbf{x}}'\|_2^2 = (2 - 1\langle \hat{\mathbf{x}}, \hat{\mathbf{x}}'\rangle)$ (Sect. III-C). ⊣

When $\hat{\epsilon}$ and $\hat{\epsilon}'$ are affected by numerical cancelation due to truncation error (i.e., $\|\hat{\epsilon} - \hat{\epsilon}'\|_2^2 \to 0$), the ν strategy intervenes as a backup ranking strategy. The first term of the sum does not affect the ranking, as it reduces to a constant factor.

Properties. We can prove that when two traces σ' and σ are equivalent (i.e., have the same sequence of labels and the same probability), the kernel computation reduces to $\omega\omega'$. When both weights are 1, the kernel returns 1. We call this condition *weak equality* because we cannot possibly prove that when the kernel is equal to $\omega\omega'$ then the two traces we are comparing are equivalent (there could be equal embeddings coming from non-equivalent traces). As shown in Sect. 1, traces having neither 2-grams nor transition labels in common have kernel 0 and vice versa (*strong dissimilarity*). Since weak equality and strong similarity hold, the embedding performs *weakly-ideally*.

V. Experimental Results

For experimenting our proposed approach to probabilistic trace alignment, we used the Sepsis Cases event log.[3]

In particular, we split the dataset into a training set, containing the "*happy traces*" lasting at most the average trace duration in the log ($\leq 2.3 \cdot 10^7$ ms), and a test set, containing the traces with the highest execution times. We used the training set to generate either an SWN, using the approach presented in [10], or a BPMN with only exclusive gates using Split Miner 2.0 [17] that was then converted into a Petri net [19]. This Petri net was later on converted into an SWN by using a firing weight estimator: we chose the `Fork` and the `PairScale` estimators from [18] and we denote as `Constant` a naive estimator assuming that all the transition enabled in a given marking are equiprobable. From such SWNs, we generated distinct sets of unfolded traces (of different sizes). The experimental settings are summarized in TABLE IV. The experiments described in the following sections have the aim of evaluating the benefits of performing the approximate-ranking strategy over the optimal-ranking one.

Approximation. To assess how well the proposed approximate-ranking strategy approximates the optimal-ranking one, we use the Spearman correlation index [20] to express the correlation between the ranking provided by each sub-embedding strategy for ϕ^g and the optimal-ranking. Fig. 7 shows the average Spearman index for traces of different lengths in the test set. We can see from the plots that the sub-embeddings considering only information about the edges (i.e., the ones where the features corresponding to the ν dimension are set to zero) have in general a higher correlation with the optimal-ranking, but their correlation values are less stable w.r.t. the length of the trace to be aligned. In the case of **STPETRI_20**, the correlation is lower than for the other configurations (lower than 0.7 for all sub-embeddings). For **SM_PAIR_20** and **SM_CONS_20**, the correlation index is around 0.8 for $\epsilon^1\&\nu^2$ and $\epsilon^2\&\nu^2$, and almost 1 for $\epsilon^1\&\nu^1$ and $\epsilon^2\&\nu^1$, but less stable for these sub-embeddings especially for longer traces. In the case of **SM_FORK_20**, the correlation is maximum for all sub-embedding strategies.

Efficiency. With reference to the plots in Fig. 8, we evaluated the efficiency of computing the trace alignment over both optimal-ranking and approximate-ranking strategies over two different data structures enabling kNN queries, i.e., VP-Trees and KD-Trees. We conducted our experiments for $k = 20$, and we used the Levenshtein distance as distance function for the optimal-ranking strategy. While the average query time (over traces of the same length) for the optimal-ranking strategy includes the *indexing time* for generating all the vectors of the search space (that has to be constructed from scratch for each query) and the time for the neighborhood search, the approximate-ranking one includes the neighborhood search time and the time needed for the embedding transformation of the trace to be aligned σ' (in this case, the indexing is performed only once before the query time); in particular, in the latter case, in addition to averaging the query time over traces of the same length, we also consider the average embedding time for all the possible embedding strategies introduced

[3] https://data.4tu.nl/articles/Sepsis_Cases_-_Event_Log/12707639

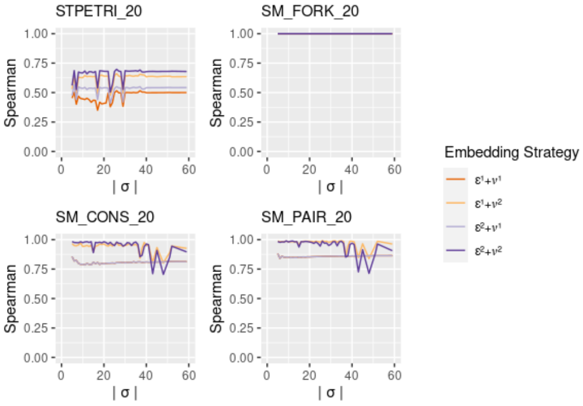


Fig. 7: Approximation comparison.

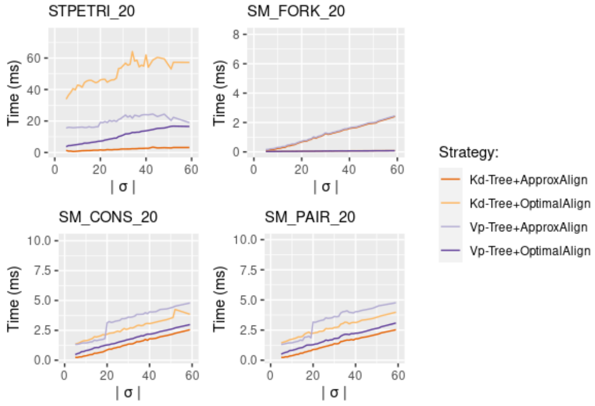


Fig. 8: kNN alignment benchmark.

in this paper (and also used in the previous section). Fig. 8 plots the result of such experiments: when using KD-Trees to represent the search space, the time required to generate all the alignments needed to compute $\mathcal{R}$ truly dominates the cost of generating the embedding $\phi^g(\overline{G}'_\sigma)$ for datasets with a higher number of model traces such as **STPETRI_20**, while the cost for $\phi^g(\overline{G}'_\sigma)$ becomes non-negligible when the stochastic net generates a more restricted set of traces, requiring to compute a lower number of alignments to generate the optimal-ranking (like, for example, in the case of **SM_FORK_20**). Finally, we can see that, in general, the computation time increases with the length of the traces to be aligned.

VI. Conclusions and Future Works

We framed probabilistic trace alignment as a kNN problem. Our approach balances the likelihood of the aligned trace and the cost of the alignment by providing the top-k alignments instead of a single alignment as output. The experimentation shows that the approximated top-k ranking provides a good trade-off between accuracy and efficiency especially when the reference stochastic net generates several model traces. Future works will investigate probabilistic alignments over fuzzy-labeled nodes and declarative process models. We also aim at improving the efficiency and accuracy of the proposed approach by intervening both on the embedding and the algorithmic strategies.

References

[1] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC 2011*. IEEE, 2011, pp. 55–64.

[2] S. J. J. Leemans, A. F. Syring, and W. M. P. van der Aalst, "Earth movers' stochastic conformance checking," in *BPM*, vol. 360. Springer, 2019, pp. 127–143.

[3] A. Polyvyanyy and A. A. Kalenkova, "Monotone conformance checking for partially matching designed and observed processes," in *ICPM*, 2019, pp. 81–88.

[4] A. Polyvyanyy, A. Solti, M. Weidlich, C. Di Ciccio, and J. Mendling, "Monotone precision and recall measures for comparing executions and specifications of dynamic systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 3, pp. 17:1–17:41, 2020.

[5] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[6] M. de Leoni and A. Marrella, "Aligning real process executions and prescriptive process models through automated planning," *Expert Syst. Appl.*, vol. 82, pp. 162–183, 2017.

[7] S. J. J. Leemans, A. F. Syring, and W. M. P. van der Aalst, "Earth movers' stochastic conformance checking," in *Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings*, ser. Lecture Notes in Business Information Processing, T. T. Hildebrandt, B. F. van Dongen, M. Röglinger, and J. Mendling, Eds., vol. 360. Springer, 2019, pp. 127–143.

[8] M. Alizadeh, M. de Leoni, and N. Zannone, "History-based construction of alignments for conformance checking: Formalization and implementation," in *SIMPDA*, vol. 237. Springer, 2014, pp. 58–78.

[9] M. A. Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems," *ACM Trans. Comput. Syst.*, vol. 2, no. 2, pp. 93–122, 1984.

[10] A. Rogge-Solti, W. M. P. van der Aalst, and M. Weske, "Discovering stochastic petri nets with arbitrary delay distributions from event logs," in *BPMW13*, 2013, pp. 15–27.

[11] G. Bergami, F. M. Maggi, M. Montali, and R. Peñaloza, "A tool for probabilistic trace alignments," in *CAiSE Forum*. Springer, 2021.

[12] T. Gärtner, P. A. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *COLT/Kernel 2003*, vol. 2777. Springer, 2003, pp. 129–143.

[13] T. Gärtner, "A survey of kernels for structured data," *SIGKDD*, vol. 5, no. 1, 2003.

[14] C. Seshadhri, A. Sharma, A. Stolman, and A. Goel, "The impossibility of low-rank representations for triangle-rich complex networks," *Proceedings of the National Academy of Sciences*, vol. 117, no. 11, pp. 5631–5637, 2020.

[15] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. J. C. H. Watkins, "Text classification using string kernels," *J. Mach. Learn. Res.*, vol. 2, pp. 419–444, 2002.

[16] K. Driessens, J. Ramon, and T. Gärtner, "Graph kernels and gaussian processes for relational reinforcement learning," *Mach. Learn.*, vol. 64, no. 1-3, pp. 91–119, 2006.

[17] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, no. 2, pp. 251–284, 2019.

[18] A. Burke, S. Leemans, and M. Wynn, "Stochastic process discovery by weight estimation," in *PQMI*, 10 2020.

[19] W. M. P. van der Aalst and B. F. van Dongen, "Discovering Petri nets from event logs," in *Trans. on Petri Nets and Other Models of Concurrency VII*, 2013, pp. 372–422.

[20] G. Bergami, F. Bertini, and D. Montesi, "Hierarchical embedding for DAG reachability queries," in *IDEAS*. ACM, 2020, pp. 24:1–24:10.