

A Tool for Computing Probabilistic Trace Alignments[★]

Giacomo Bergami¹[0000–0002–1844–0851], Fabrizio Maria Maggi¹[0000–0002–9089–6896], Marco Montali¹[0000–0002–8021–3430], and Rafael Peñaloza²[0000–0002–2693–5790]

¹ Free University of Bozen-Bolzano, Italy
gibergami@unibz.it, {maggi, montali}@inf.unibz.it
² University of Milano-Bicocca
rafael.penaloza@unimib.it

Abstract. Alignments pinpoint trace deviations in a process model and quantify their severity. However, approaches based on trace alignments use crisp process models and recent probabilistic conformance checking approaches check the degree of conformance of an event log with respect to a stochastic process model instead of finding trace alignments. In this paper, for the first time, we provide a conformance checking approach based on trace alignments using stochastic Workflow nets. Conceptually, 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.

Keywords: Stochastic Petri Nets · Conformance Checking · Alignments.

1 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. Yet, 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 [9], or by exploiting entropy-based measures [13, 14]. As these strategies are not based on trace alignments and do not provide a ranked list of alignments, these cannot be directly used to repair a given trace with one of the traces generated by a stochastic process model. In this paper, we provide for the first time a tool for the probabilistic alignment of a trace and a stochastic reference model. Providing different alignment options is useful since, conceptually, probabilistic trace alignment requires the analyst to balance between the likelihood of the model trace with respect to which the alignment is computed and its cost.

[★] This research has been partially supported by the project IDEE (FESR1133) funded by the Eur. Reg. Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020.

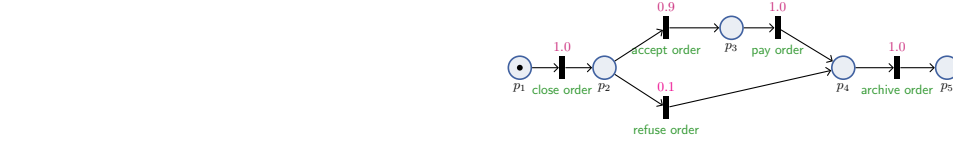


Fig. 1: Stochastic Workflow Net modeling our use-case scenario.

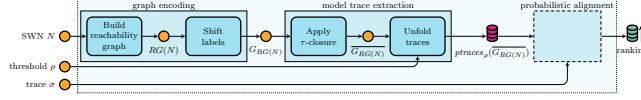


Fig. 2: Proposed pipeline to assess the probabilistic trace alignment.

With reference to Figure 1, a user might be interested to align the log trace $\langle \text{close order}, \text{archive order} \rangle$ with one of the two possible model traces $\langle \text{close order}, \text{accept order}, \text{pay order}, \text{archive order} \rangle$ or $\langle \text{close order}, \text{refuse order}, \text{archive order} \rangle$. 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. To do this, we frame the probabilistic trace alignment problem into the well-known k -Nearest Neighbors (k NN) problem [2] that refers to finding the k nearest data points to a *query* x from a set \mathcal{X} of *data points* via a distance function defined over $\mathcal{X} \cup \{x\}$. We introduce two ranking strategies. The first one is based on a brute force approach that reuses existing trace aligners [10] requiring to re-compute the alignments for all the possible traces. For models generating a large number of model traces, this becomes suboptimal: the second strategy produces an approximate ranking where x and \mathcal{X} are represented as numerical vectors via an embedding ϕ . If the embeddings ϕ for \mathcal{X} are independent of the query of choice x , this does not require to constantly recompute the numeric vector representation for \mathcal{X} and to pre-order it to efficiently visit the search space. We implemented and tested our solution ³.

2 Probabilistic Trace Alignment Pipeline

Our approach (Fig. 2) takes as input (i) a reference model represented as an Stochastic Workflow Nets N or an equivalent Transition Graph G , (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 greater than or equal to ρ , based on a combined consideration of their probability values and their distance to σ . First, we discuss input formats for (i) as in the GUI.

Format: Petri_PNML. Petri Nets and Generalized Stochastic Petri Nets are well-established formalisms [14] for modelling processes [15] represented in the

³ <https://github.com/jackbergus/approxProbTraceAlign>

Petri Net Markup Language, supported by our tool. Due to the lack of space, we refer to [5] for the usual notation over Petri Nets. We restrict our interest to an interesting class of 1-*bounded* stochastic Petri nets with no timed transitions, namely UNTIMED STOCHASTIC WORKFLOW NETS denoted as N . We now sketch the properties of the SWN accepted in PNML format and loaded as **PetriNet** objects: we consider two customary markings: the *input* (resp. *output*) marking m_{in} (resp. m_{out}) assigning a single token to the input (resp. output) place, and no token elsewhere. We assume to have 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). Labels are associated to transitions via a labelling function λ . A *trace* is a finite sequence of labels from \mathcal{A} . The current state of execution is captured by a marking, i.e., a multiset of places P indicating how many tokens populate each place. The notions of transition enablement and firing are also the standard ones [12]: given a marking m and an enabled transition $t \in T_e$, the *firing probability* of t in m is $\mathbb{P}_{m,N}(t) = \frac{W(t)}{\sum_{t' \in T_e} W(t')}$. The probabilities associated to all enabled transitions in a marking always add up to 1. A *valid sequence* $\eta = t_1 \cdots t_n$ is a firing sequence starting from m_{in}^N and resulting in m_{out} . The probability $\mathbb{P}_N(\eta)$ of a valid sequence is the product of the probabilities associated to each transition: $\mathbb{P}_N(\eta) = \prod_{i \in \{1, \dots, n\}} \mathbb{P}_{m_{i-1}, N}(t_i)$. A trace $\sigma = a_1 \cdots a_m$ is a *model trace* (or N -trace for short) if there exists a valid sequence $\eta = t_1 \cdots t_n$ where the appended labels $\lambda(t_1) \cdots \lambda(t_n)$ is equivalent to σ once all the τ -s are stripped. There may be multiple valide sequences $\eta \in seqs_N(\sigma)$ underlying an N -trace σ . *traces*(N) is the (possibly infinite) set of N -traces. 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: $\mathbb{P}_N(\sigma) = \sum_{\eta \in seqs_N(\sigma)} \mathbb{P}_N(\eta)$. This corresponds to the intuition 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. These structural properties makes them suitable for loading BPMN nets as well: when the **Petri.BPMN** is choosen, either the firing weight estimator is constant by default (**W.CONSTANT**), or we can choose one from [5].

Example 1. Fig. 3 shows an example of an SWN with input place p_1 and output place p_7 . One run of the net is $\tau c \tau a a \tau$, which corresponds to trace **caa**. Overall, the net supports infinitely many finite traces of the form (represented using regular expressions): (i) aa^* , (ii) cb , (iii) caa^* .

Format: StochasticMatrix. The graph and trace embedding techniques cannot be directly defined over reachability graphs, as they rely on probabilistic TRANSITION GRAPHS [8] (class **ReadGraph**). Such graphs have transition probabilities associated to the edges, while nodes have labels in Σ , and represented via transition matrices. Each node is mapped by a matrix L to a single label, as the same label may be used for multiple nodes, while a matrix R represents a probability distribution over the next nodes to be picked upon executing a transition. Due to the lack of space, we refer to [8] for the standard notation for probabilistic Transition Graphs: matrices L and R can be exploited to determine

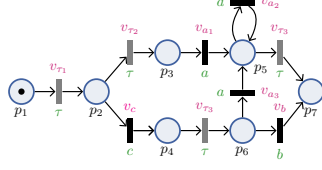


Fig. 3: A sample SWN. Labels are shown in green, τ transitions in grey, the SWN N . Probabilities are shown in magenta.

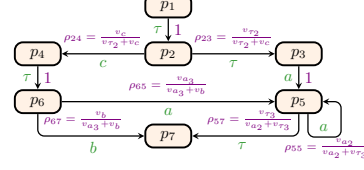


Fig. 4: Reachability graph $RG(N)$ of the SWN N . Probabilities are shown in violet.

the probability of reaching a node labeled by $\beta \in \Sigma$ from any node labeled $\alpha \in \Sigma$ in n steps with $[LR^n L^\top]_{\alpha\beta} / [LL^\top]_{\alpha\alpha}$, that we can shorthand as $[G.A^n]_{\alpha\beta}$ [8]. A transition graph G can be visualized as shown in Fig. 5. Still, an edge from node i to node j is only shown if the transition probability $[G.R]_{ij}$ is positive. Valid sequences and model traces as well as their probability are defined similarly from SWNs, and we employ the same notation. Matrices as well as Linear Algebra operations are implemented using sparse matrices and vectors from Eigen3.

Transforming SWN \rightarrow TG. If input is provided as a SWN N , we internally represent all transition firings of an SWN, together with their probabilities, in a reachability graph $RG(N)$ (Fig. 4 and **PetriMatrix** file format) by interpreting concurrency by interleaving. The transition probability function P assigns to each transition an edge probability in $RG(N)$ as customary. Due to the lack of space, we skip the customary definition of $RG(N)$. For a given state, distinct net transitions with the same label might produce the same consequent state, thus becoming indistinguishable from the trace perspective as they collapse into a single edge of the reachability graph. So, we accumulate all their firing probabilities into a single edge value. Given an SWN N , we always assume that, in addition to its boundedness $RG(N)$ does not contain loops where all edges are labeled with τ . Multiple τ -transitions may be used, but never creating completely invisible loops. The absence of loops that solely go through τ -transitions is reasonable modeling wise (as skipping multiple times an entire loop iteration is conceptually not distinguishable from not skipping it at all - so we would require there the presence of at least a visible activity witnessing that the iteration was executed, possibly skipping parts of it). Under this assumption, there are only boundedly many valid sequences that can produce a given trace σ . 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. We directly encode **PetriNets** to **ReadGraph**.

Example 2. Consider the SWN N of Example 1. Considering trace caa , it is easy to see that it has only one underlying run, namely $\tau c \tau a a \tau$, in turn produced by a single underlying valid sequence, and that $\mathbb{P}_N(caa) = 1 \cdot \rho_{24} \cdot 1 \cdot \rho_{65} \cdot \rho_{55} \cdot \rho_{57}$.

We can show that there exists a conversion from $RG(N)$ (Fig. 4) into a transition graph $G_{RG(N)}$ (Fig. 5) preserving model traces as well as their probabilities via well-known techniques used to *shift labels* from automata theory.

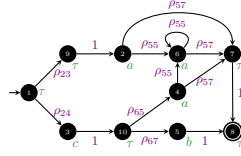


Fig. 5: Transition graph $G_{RG(N)}$ encoding the reachability graph $RG(N)$.

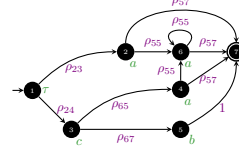


Fig. 6: Transition graph $\overline{G_{RG(N)}}$ resulting from the transition graph in $G_{RG(N)}$ after τ -closure.





τ -closure The resulting transition graph $G_{RG(N)}$ is processed applying a τ -closure that compiles away τ -transitions. This results into a new transition graph $\overline{G_{RG(N)}}$ (Fig. 6) retaining τ labels only in the initial and accepting states while, for the rest, it exclusively operates over visible labels in \mathcal{A} . The transformation relies on well-known automata-based techniques for removing ϵ -moves while preserving model traces, as well as their associated probabilities. All τ transitions can still be removed thanks to the working hypothesis done for SWNs, as no loops can contain only τ transitions.





Unfolding. Next, we efficiently unfold the previously closed graph to collect TG-traces having probability greater or equal than ρ (Minimum trace probability) via the Eigen3 linear algebra library. Unfolding is efficiently performed via sparse matrices for TGs over the Eigen library. Since an SWN has a Workflow net as underlying control-flow structure and given that TG inherits the results from SWNs, no loop can be executed without strictly decreasing the resulting probability. So, all valid sequences with a resulting probability of at least ρ can be enumerated and returned in a set. The so-obtained sequences are then combined by merging those that produce the same trace, summing up their probabilities, thus obtaining the set of all the traces having a probability greater than or equal to ρ , $ptraces_{\rho}(\overline{G_{RG(N)}})$. The closure operation also implies that the notion of model trace collapses with the one of run modulo removing the initial and the final τ labels attached to the initial and accepting nodes.

Alignment Strategy. The last step takes the so-obtained TG-traces and ranks the best k by considering their probabilities and the alignment cost with the trace σ of interest. In the database community, this problem is usually tackled via k -Nearest Neighbors (kNN) that refers to finding the k nearest data points to a *query* x from a set \mathcal{X} of *data points* via a distance function d_k defined over $\mathcal{X} \cup \{x\}$. In particular, by exploiting ad-hoc data structures, such as VP-Trees and KD-Trees, we can retrieve the neighborhood of x in \mathcal{X} of size k by pre-ordering (*indexing*) \mathcal{X} via d_k and starting the search from the top-1 alignment. Our tool benchmarks both strategies and gives results in CSV.

1) Optimal-Ranking Trace Aligner. By reusing existing trace aligners [1, 10] providing an alignment cost $d(\sigma, \sigma')$ between model and log traces as the Levenshtein distance and using the decision theory, we can express the ranking as a preliminary ranked distance $\mathbb{P}_N(\sigma')d(\sigma, \sigma')$, considering both the alignment

Table 1: Projections of $\overline{G_{RG(N)}}$ over N -traces of length 4.

σ'	$G_{\sigma'}$	l	ω
a		1	$\rho_{23}\rho_{57}$
cb		2	ρ_{24}
aaa		3	$\rho_{23}\rho_{57}$
caa		3	$\rho_{24}\rho_{57}$

σ'	$G_{\sigma'}$	l	ω
aa		2	$\rho_{23}\rho_{57}$
ca		2	$\rho_{24}\rho_{57}$
aaaa		4	$\rho_{23}\rho_{57}$
caaa		4	$\rho_{24}\rho_{57}$

cost and the model trace probability. We can represent it as a ranking function returning 1 when $\sigma' = \sigma$ and $\mathbb{P}_N(\sigma) = 1$ hold. We then express d as a normalized similarity score $s_d(\sigma, \sigma') := \frac{1}{d(\sigma, \sigma') + 1}$. The golden ranking function (i.e., the one producing the optimal ranking) can therefore be represented as $\mathcal{R}(\sigma, \sigma') = \mathbb{P}_N(\sigma')\mathbb{P}_N(\sigma)s_d(\sigma, \sigma')$. The computation $\max \arg_{\sigma' \in \text{ptraces}_\rho(N)} \mathcal{R}(\sigma, \sigma')$ 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 .

2) Approximate-Ranking Trace Embedder. Since each embedding ϕ entails an associated similarity metric k_ϕ , we can compute the embeddings for all the unfolded traces before performing the top- k search ensuring that they are independent of the trace to align, thus avoiding the brute-force cost. This computational gain comes with a loss in precision [8] and, in its approximated version, is not able to accurately represent the data using low-dimensional vectors [16]. To obtain our proposed embedding ϕ^g , we adapt the embedding strategy ϕ^{tr} from [11] by addressing some shortcomings: we **(a)** propose a weakly-ideal embedding [7] which, differently from current literature, **(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 $\sigma' \in \text{ptraces}_\rho(N)$ adequately representing the transitions in $\overline{G_{RG(N)}}$ requires an intermediate G representation, we map each σ' to a pair $(G_{\sigma'}, \omega)$, where *(i)* $G_{\sigma'}$ is a transition graph containing all nodes and edges leading to the valid sequences in $\overline{G_{RG(N)}}$ forming a trace σ' , where all τ -labelled nodes are removed, while *(ii)* the graph weight ω preserves the weight of the possible initial and final edges that were removed due to the former requirement.

Example 3. Given the τ -closed transition graph $\overline{G_{RG(N)}}$ in Fig. 6, 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 $\text{ptraces}_0(\overline{G_{RG(N)}})$ with maximum length 4 are: $\{\langle a, 0.4 \rangle, \langle aa, 0.2 \rangle, \langle aaa, 0.1 \rangle, \langle ca, 0.07 \rangle, \langle cb, 0.06 \rangle, \langle aaaa, 0.05 \rangle, \langle caa, 0.035 \rangle, \langle caaa, 0.0175 \rangle\}$. Table 1 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).

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

	$x = 1$	$x = 2$
$\epsilon_{ab}^x(\overline{G}_{\sigma'}) :=$	$\sum_{i=1}^l \lambda^i \frac{[LR^i L^i]_{ab}}{\sum_{\sigma' b' \in \mathcal{A}^2} R_{b'}^i}$	$\sum_{i=1}^l \lambda^i [A^i]_{ab}$
$\nu_a^x(\overline{G}_{\sigma'}) :=$	$\frac{1}{c} \sum_{\sigma'' \in p\text{traces}_0(\overline{G}_{\sigma'})} \frac{ \{\sigma'_i \in \sigma'' \mid a \in \mathcal{A} \wedge \sigma''_i = a\} }{ \sigma'' }$	0

Our proposed embedding ϕ^g is computed for each pair $(G_{\sigma'}, \omega)$ generated from each unfolded trace. We can now extend the embedding ϕ^{tr} [11] 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 1 (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 ϕ^g over the visible 2-grams and transition labels, $\mathcal{A} \cup \mathcal{A}^2$, is defined by*

$$\phi_i^g(\overline{G}_{\sigma'}) = \begin{cases} \omega \frac{\epsilon_{ab}(\overline{G}_{\sigma'})}{\|\epsilon\|_2} t_f^{|R|>0|} & i = ab \\ \nu_a(\overline{G}_{\sigma'}) t_f^{|R|>0|} & i = a \end{cases}$$

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, $\max \arg_{\sigma' \in p\text{traces}_p(N), G_{\sigma'} \in \mathbf{G}_p(P)} k_{\phi^g}(G_{\sigma'}, G_{\sigma'})$ 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 ($x = 1$ and $x = 2$) shown in Table 2: here, l is the path length (reported in Table 1), and c for ν^1 is a normalization factor such that $\sum_{a \in \mathcal{A}} \nu_a^1(P) = 1$. While ν^2 implies to completely ignore the label frequency contribution, ϵ^2 is the direct implementation of ϕ^{tr} from [11], and ϵ^1 and ϵ^2 only differ from the normalization perspective. $t_f \in [0, 1] \subseteq \mathbb{R}_0^+$ (**Tuning**) and $\lambda \in [0, 1] \subseteq \mathbb{R}_0^+$ (**Lambda**) are tuning parameters that can be inferred from the available data [6]. 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 $\lambda = 0.07$. Omitted proofs show that our embedding performs weakly-ideally.

Example 4. The dot products $k_{\phi^g}(\sigma, \sigma') = \langle \phi^g(\overline{G}_{\sigma}), \phi^g(\overline{G}_{\sigma'}) \rangle$ with sub-embedding ν^1 and ϵ^1 , for each trace $\sigma' \in p\text{traces}_0(N)_{|\sigma'| \leq 4}$, are represented in Table 3. k_{ϕ^g} approximates the optimal ranking as it tends to rank the transition graphs $\overline{G}_{\sigma'}$ (generated from \overline{G} via projection) similarly to the N -traces over \mathcal{R} . In the same table, the ranking similarities shared between the two different ranking strategies are highlighted in blue, while the most evident ranking discrepancies are marked in red. Such similarities and differences can be assessed via Spearman

Table 3: Comparison between the ranking induced by the optimal ranking \mathcal{R} and the proposed kernel $k_{\phi g}$ with embedding strategies ϵ^1 and ν^1 : arrows \downarrow remark the column of choice under which we sort the rows.

σ'	$ \langle \mathbb{P}_N(\sigma'), \downarrow s_d(\sigma, \sigma') \rangle $	$\mathcal{R}(\sigma, \sigma')$	$k_{\phi g}(\overline{G}_\sigma, \overline{G}_{\sigma'})$	σ'	$\downarrow \mathcal{R}(\sigma, \sigma')$	σ'	$\downarrow k_{\phi g}(\overline{G}_\sigma, \overline{G}_{\sigma'})$	
caa	0.035	0.8333	0.0292	$1.14 \cdot 10^{-40}$	a	0.2500	a	$8.16 \cdot 10^{-21}$
caaa	0.0175	0.8333	0.0145	$9.84 \cdot 10^{-41}$	aa	0.1428	ca	$1.89 \cdot 10^{-24}$
a	0.4	0.6250	0.2500	$8.16 \cdot 10^{-21}$	aaa	0.0714	cb	$7.64 \cdot 10^{-25}$
aaaa	0.05	0.6250	0.0357	$8.44 \cdot 10^{-41}$	ca	0.0500	caa	$1.14 \cdot 10^{-40}$
aa	0.2	0.7142	0.1428	$9.28 \cdot 10^{-41}$	cb	0.0428	caaa	$9.84 \cdot 10^{-41}$
aaa	0.1	0.7142	0.0714	$8.72 \cdot 10^{-41}$	aaaa	0.0357	aa	$9.28 \cdot 10^{-41}$
ca	0.07	0.7142	0.0500	$1.89 \cdot 10^{-24}$	caa	0.0292	aaaa	$8.44 \cdot 10^{-41}$
cb	0.06	0.7142	0.0428	$7.64 \cdot 10^{-25}$	caaa	0.0145	aaa	$8.72 \cdot 10^{-41}$

Correlation Index [4]. Evaluations exploiting such index were removed due to lack of space; our future work will include those as well as temporal benchmarks.

References

- Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: EDOC 2011. pp. 55–64. IEEE (2011)
- Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* **46**(3), 175–185 (1992)
- Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. *Knowl. Inf. Syst.* **59**(2), 251–284 (2019)
- Bergami, G., Bertini, F., Montesi, D.: Hierarchical embedding for DAG reachability queries. In: IDEAS. pp. 24:1–24:10. ACM (2020)
- Burke, A., Leemans, S., Wynn, M.: Stochastic process discovery by weight estimation. In: PQMI (10 2020)
- Driessens, K., Ramon, J., Gärtner, T.: Graph kernels and gaussian processes for relational reinforcement learning. *Mach. Learn.* **64**(1-3), 91–119 (2006)
- Gärtner, T.: A survey of kernels for structured data. *SIGKDD* **5**(1) (2003)
- Gärtner, T., Flach, P.A., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: COLT/Kernel 2003. vol. 2777, pp. 129–143. Springer (2003)
- Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers’ stochastic conformance checking. In: BPM. vol. 360, pp. 127–143. Springer (2019)
- de Leonì, M., Marrella, A.: Aligning real process executions and prescriptive process models through automated planning. *Expert Syst. Appl.* **82**, 162–183 (2017)
- Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.J.C.H.: Text classification using string kernels. *J. Mach. Learn. Res.* **2**, 419–444 (2002)
- Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.* **2**(2), 93–122 (1984)
- Polyvyanyy, A., Kalenkova, A.A.: Monotone conformance checking for partially matching designed and observed processes. In: ICPM. pp. 81–88 (2019)
- Polyvyanyy, A., Solti, A., Weidlich, M., Di Ciccio, C., Mendling, J.: Monotone precision and recall measures for comparing executions and specifications of dynamic systems. *ACM Trans. Softw. Eng. Methodol.* **29**(3), 17:1–17:41 (2020)
- Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: BPMW13. pp. 15–27
- Seshadhri, C., Sharma, A., Stolman, A., Goel, A.: The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences* **117**(11), 5631–5637 (2020)