



# A Tool for Computing Probabilistic Trace Alignments

Giacomo Bergami<sup>1</sup>, Fabrizio Maria Maggi<sup>1</sup>, Marco Montali<sup>1</sup>,  
and Rafael Peñaloza<sup>2</sup>

<sup>1</sup> Free University of Bozen-Bolzano, Bolzano, Italy  
gibergami@unibz.it, {maggi,montali}@inf.unibz.it

<sup>2</sup> University of Milano-Bicocca, Milan, Italy  
rafael.penaloz@unimib.it

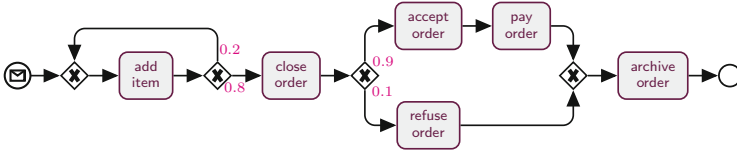
**Abstract.** Alignments pinpoint trace deviations in a process model and quantify their severity. However, approaches based on trace alignments use crisp process models and recent probabilistic conformance checking approaches check the degree of conformance of an event log with respect to a stochastic 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

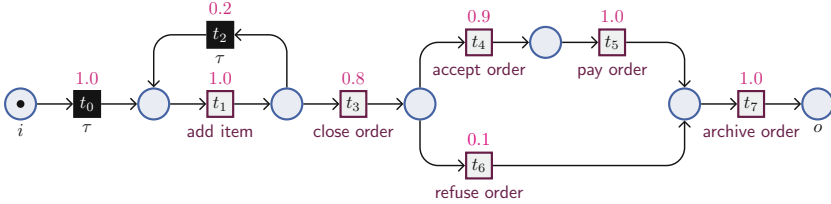
## 1 Introduction

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 [7], or by exploiting entropy-based measures [10, 11]. As these strategies are not based on trace alignments, they cannot be directly used to repair a given trace with one of the traces generated by a stochastic process model. In this paper, we present, for the first time, a tool for the probabilistic alignment of a trace and a stochastic reference model. The tool provides different alignment options since, conceptually, probabilistic trace alignment requires the analyst to balance between the alignment cost and the likelihood of model traces: an optimal but very unlikely alignment could, in fact, be much less interesting than a slightly worse but very likely alignment.

With reference to Fig. 1, a user might be interested in aligning the log trace  $\langle \text{add item, close order, archive order} \rangle$  with one of the two possible model traces  $\langle \text{add item, close order, accept order, pay order, archive order} \rangle$  or



**Fig. 1.** A simple order management process in BPMN, with choice probabilities.



**Fig. 2.** Encoding of the BPMN diagram in Fig. 1 using a stochastic Workflow net with bounded silence.

(add item, close order, refuse order, archive order). The latter model trace provides the least alignment cost, but comes with a very low probability ( $0.8 \cdot 0.1 = 0.08$ ); on the other hand, the former model trace gives a slightly worse alignment cost, but comes with a much higher probability ( $0.8 \cdot 0.9 = 0.72$ ). Depending on the context, analysts might prefer either the former or the latter alignment. In general, finding a portfolio of the “best”  $k$  alignments among all the distinct model traces empowers analysts to find their own trade-off between alignment cost and model trace probability.

To achieve 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 [8] requiring to re-compute the alignments for all possible traces in the log. For models generating a large number of model traces, this clearly becomes inefficient. Therefore, we propose a second strategy that produces an approximate ranking where  $x$  and  $\mathcal{X}$  are represented as numerical vectors via an embedding  $\phi$ . If the embedding  $\phi$  for  $\mathcal{X}$  is independent of the query of choice  $x$ , this does not require to constantly recompute the numeric vector representation for  $\mathcal{X}$ . Instead, it is possible to pre-order it to efficiently visit the search space. The developed tool is publicly available<sup>1</sup>.

<sup>1</sup> <https://github.com/jackbergus/approxProbTraceAlign>.

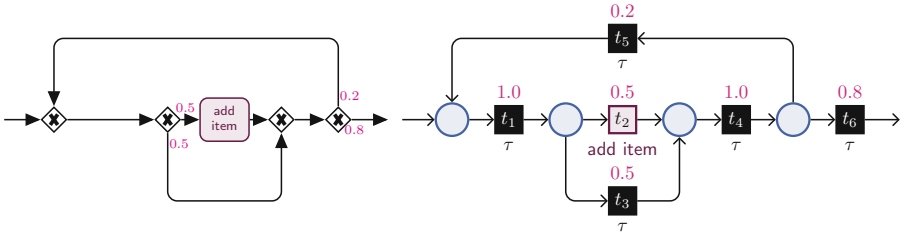


Fig. 3. A process model with a loop accepting fully silent iterations.

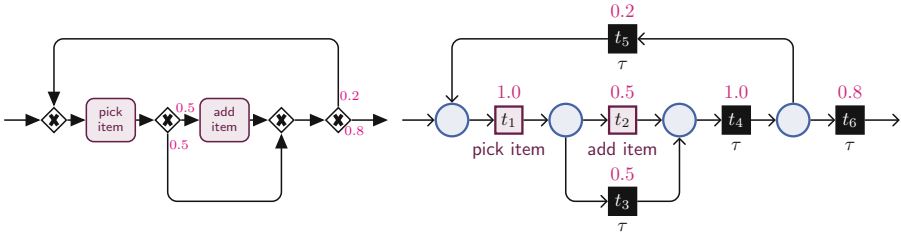


Fig. 4. A process model with a loop accepting iterations with skippable steps.

## 2 The Process Models

To formalize the kind of processes shown in Fig. 1, we resort to a special class of stochastic Petri nets, following what has been done in the literature [7, 11]. We describe next the features of the class we consider, and why they lead to an interesting trade-off between expressiveness and amenability to analysis. Figure 2 shows the encoding of the BPMN diagram of Fig. 1 into the Petri net class supported by our conformance checking tool.

**Untimed, Stochastic Nets.** We focus on stochastic Petri nets with immediate transitions, that is, we do not consider timed aspects such as delays and deadlines, but concentrate on the key feature of having a probability distribution over the enabled transitions. This is achieved by taking a standard Petri net and by assigning weights to its transitions. At each execution step, the probability of firing an enabled transition is then simply computed by dividing its weight by the total weight of all currently enabled transitions.

**Workflow Nets.** In the whole Petri net spectrum, we focus, as customary in process mining, on Workflow nets with a distinguished pair of input and output places, marking the start and completion of a case in the process. Specifically, a *model run* is a sequence of fireable transitions leading from the initial marking (which assigns one single token to the special input place, while leaving all the other places empty) to the final marking (which assigns one single token to the special output place, while leaving all the other places empty). As usual, the probability of a model run is then computed by multiplying the probabilities

of each transition. For example, by considering the net of Fig. 2, we have that  $\rho = \langle t_0, t_1, t_3, t_6, t_7 \rangle$  is a model run whose probability is  $0.8 \cdot 0.1 = 0.08$ .

In our specific setting, focusing on Workflow nets has the advantage that every model run is a maximal sequence of transition firings that cannot be extended into a different model run. This provides a direct way to characterize the (finite-length) runs accepted by the Workflow net and their probabilities, without the need of introducing additional constructs such as the probability of stopping in a marking.

**Silent Transitions.** To provide support for control-flow gateways, we include silent transitions in the net. More specifically, every transition comes with a label that corresponds either to the name of a (visible) task, or to the special symbol  $\tau$  (denoting a silent transition). Figure 2 shows how  $\tau$ -transitions are used to capture the BPMN process of Fig. 1. In particular, silent transition  $t_2$  is used to model that one can loop to add multiple items to the order. Notice that, for simplicity of modeling, we support the possibility of labeling multiple transitions with the same task.

Having silent transitions and repeated labels deeply impacts the obtained framework. In fact, a model run does not directly correspond to a model trace, intended as a “legal” sequence of (visible) tasks according to the process. On the one hand, a model run yields a corresponding trace by extracting, in order, the labels attached to the transitions contained therein, projecting away the invisible ones. For example, model run  $\rho$  above yields the model trace  $\langle \text{add item}, \text{close order}, \text{archive order} \rangle$ . On the other hand, the same model trace may be obtained through distinct model runs, differing from each other in terms of the silent tasks they contain. Hence, in general, to obtain the probability of a model trace, one must sum up the probabilities of *all* (possibly, infinitely many) model runs yielding that trace. This number is guaranteed, by construction, to be between 0 and 1 (since the collective sum of the probabilities of all model runs is indeed 1). In our example, the probability of  $\langle \text{add item}, \text{close order}, \text{archive order} \rangle$  is that of the model run  $\rho$  (i.e., 0.08), since  $\rho$  is the only model run yielding that trace.

**Nets with “Bounded Silence”.** The last requirement we impose over our nets is that of *bounded silence*. This requirement states that the net cannot accept runs containing unboundedly many consecutive silent transitions. Mathematically, this means that there exists an a-priori bound on the maximum number of silent transitions that can be fired between two visible transitions.

In conceptual modeling terms, this requirement imposes that it is not possible to have loops in the process where an entire iteration consists only of visible transitions, that is, where an iteration can be executed without any visible task to witness its existence. We argue that, in this case, executing an entire iteration where all visible transitions are skipped is not different from the situation where the iteration is not executed at all. Consider, for example, the process fragment shown in Fig. 3. There, the trace consisting of three item additions could be produced by infinitely many distinct runs, each containing a different number of silent iterations in the loop.

On the other hand, it is perfectly possible to have loops with skippable tasks, provided that there exists at least one visible transition witnessing that an iteration in the loop has been executed. Figure 4 shows a variant of the process fragment in Fig. 3 where each iteration must be witnessed by (visibly) picking an item, then deciding whether to add it or not (the latter choice resulting in a silent step). This variant has bounded silence, as two consecutive iterations need to contain two distinct executions of the pick item task, with at most one silent transition in between.

In mathematical terms, our untimed, stochastic workflow nets with bounded silence enjoy the following property. Consider a net with a bound  $b$  on the maximum number of consecutive silent transitions. Given a model trace of length  $n$ , a model run yielding that trace must have a length bounded by  $n + (n + 1) \cdot b$ . This gives us a direct way to compute the probability of such a model trace:

1. we fetch all model runs of length at most  $n + (n + 1) \cdot b$  (which can be easily done with a bounded-depth search strategy over reachable markings);
2. among all such runs, we keep all and only those that yield the model trace of interest;
3. we sum up the probabilities of the so-filtered model runs.

Notice that if the net is bounded in the usual Petri net sense, then we can directly compute this probability by inspecting the reachability graph of the net.

### 3 The Probabilistic Trace Alignment Tool

Our tool takes as input (i) a reference model represented as a BPMN model or an equivalent stochastic Workflow net, (ii) a minimum positive probability threshold  $\rho \in (0, 1]$  (iii) a trace  $\sigma$  of interest, and returns a ranking over all the model traces having a probability greater than or equal to  $\rho$ , based on a combined consideration of their probability values and their distance from  $\sigma$ .

**Transition Graphs.** The model trace probabilities can be directly computed by inspecting the reachability graph of the reference model. Still, graph embedding techniques required to represent traces as data points (e.g., vectors) cannot be directly defined over reachability graphs since they rely on probabilistic *Transition Graphs* [6]. Such Transition Graphs can be computed by shifting the transition labels over graph nodes, and performing  $\tau$ -closures, while preserving  $\tau$ -transitions for both start and completion nodes, if required, to preserve trace probabilities. An example of a Transition Graph is shown in Fig. 5.

**Alignment Strategies.** We frame the probabilistic trace alignment problem into the well-known  $k$ -Nearest Neighbors ( $k$ NN) problem 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.

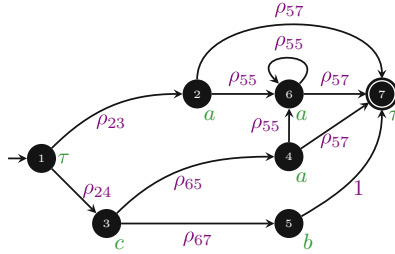



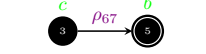
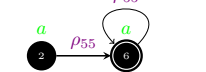
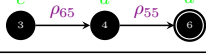
Fig. 5. An example of a Transition Graph.



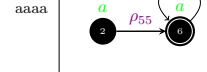
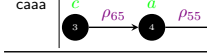
**1) Optimal-Ranking Trace Aligner.** One way to probabilistically align traces is to reuse existing trace aligners [1, 8], where the distance  $d(\sigma, \sigma')$  between model and log traces is the Levenshtein distance. We can then express the ranking score as the product  $\mathbb{P}_N(\sigma')d(\sigma, \sigma')$ , considering both the alignment cost (given by the distance between the model trace and the trace to be aligned) and the model trace probability. We can represent this weighted distance as a ranking function returning 1 when  $\sigma' = \sigma$  and  $\mathbb{P}_N(\sigma) = 1$  hold. To this aim, we need to 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  $\mathcal{R}(\sigma, \sigma') = \mathbb{P}_N(\sigma')\mathbb{P}_N(\sigma)s_d(\sigma, \sigma')$ . Such a function can be exploited to generate the best optimal-ranking trace alignment for a log trace  $\sigma$ , where  $\mathcal{R}$  must be computed a-new for all possible  $\sigma$ .

**2) Approximate-Ranking Trace Embedder.** Ranking optimality comes at the sub-optimal cost of a brute-force recomputation of  $\mathcal{R}$  for each novel trace  $\sigma$  to align. Since each embedding  $\phi$  entails an associated similarity metric  $k_\phi$ , and hence an associated distance  $d_{k_\phi}$ , we can compute the embeddings for all the model 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 [3, 6] and, in its approximated version, is not able to accurately represent the data using low-dimensional vectors [12]. Our aim is to represent model traces, which might be composed by different valid sequences (paths), as vectors. We are also interested in the intersection of embedding strategies for whole node-labeled graphs with string embedding traces, as we use vectors to compare model traces with log traces.

To obtain our proposed embedding  $\phi^g$ , we hence adapt the embedding strategy  $\phi^{\text{tr}}$  from [9] by addressing some shortcomings: we **(a)** propose a weakly-ideal embedding [5], which, differently from current literature, **(b)** exploits an  $\omega$  factor for preserving probabilities from and to  $\tau$  transitions. We also **(c)** mitigate the numerical truncation errors induced by trace length and probability distribution skewness through two sub-embedding strategies,  $\epsilon$  and  $\nu$ , where the former descends from  $\phi^{\text{tr}}$  and the latter approximates the trace similarity via label frequency similarity. Since a model trace embedding, in our tool, requires an intermediate representation  $G$  of the model trace, we map each  $\sigma'$  to a pair

**Table 1.** Projected Transition Graphs associated to model traces with maximum length 4.

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

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

$(G_{\sigma'}, \omega)$ , where (i)  $G_{\sigma'}$  is a transition graph containing all the paths describing  $\sigma'$ , where all  $\tau$ -labeled nodes are removed, while (ii) the graph weight  $\omega$  preserves the weight of the possible initial and final edges that were removed due to the former requirement.

Given the  $\tau$ -closed Transition Graph 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 model traces 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  $\tau$ -labeled nodes are removed).

Our proposed embedding  $\phi^g$  is computed for each pair  $(G_{\sigma'}, \omega)$ . The goal is to use  $k_{\phi^g}$  for ranking all model traces. To this aim, we extend the embedding  $\phi^{\text{tr}}$  [9] by including the trace probabilities, and making the ranking induced by  $k_{\phi^g}$  the inverse of the ranking induced by the sum of the following distances: the transition correlations  $\epsilon$  and the transition label frequency  $\nu$ . Therefore, our proposed  $\phi^g$  embedding is defined as follows:

**Definition 1 (G-Embedding).** Given a  $\bar{G}$  projection over  $\sigma'$   $(\bar{G}_{\sigma'}, \omega)$  and a tuning parameter  $t_f \in [0, 1] \subseteq \mathbb{R}_0^+$  (that can be inferred from the available data [4]), the G-Embedding  $\phi^g$  exploiting the matrix representation of [6] in  $\nu$  and  $\epsilon$  for Transition Graphs is defined as

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

Here, the kernel function associated to such embedding can be exploited to return the best approximated trace alignment for a log trace represented as  $G_{\sigma'}$ . It can be proven that our embedding performs weakly-ideally. In addition, the proposed embedding is independent of the trace to be aligned. Therefore, it does not have to be recomputed at each alignment for a different  $\sigma$ .

**Table 2.** Comparison between the ranking induced by the optimal ranking  $\mathcal{R}$  and the proposed kernel  $k_{\phi g}$ : arrows  $\downarrow$  remark the column of choice under which we sort the rows.

$\sigma'$	$(\mathbb{P}_N(\sigma'), \downarrow s_d(\sigma, \sigma'))$	$= \mathcal{R}(\sigma, \sigma')$	$k_{\phi g}(\overline{G}_\sigma, \overline{G}_{\sigma'})$	$\sigma'$	$\downarrow \mathcal{R}(\sigma, \sigma')$	$\sigma'$	$\downarrow k_{\phi g}(\overline{G}_\sigma, \overline{G}_{\sigma'})$	
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	caaaa	$9.84 \cdot 10^{-41}$
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	$8.72 \cdot 10^{-41}$

Table 2 shows the output of our tool. In particular, in the example, the kernel  $k_{\phi g}$  of model traces of maximum length 4 is provided. From the results, it is possible to see that  $k_{\phi g}$  approximates the optimal ranking as it tends to rank the transition graphs  $\overline{G}_{\sigma'}$  (generated from  $\overline{G}$  via projection) similarly to the model traces over  $\mathcal{R}$ . In the 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.

**Acknowledgments.** 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.

## 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. *Am. Stat.* **46**(3), 175–185 (1992)
- Bergami, G., Bertini, F., Montesi, D.: Hierarchical embedding for DAG reachability queries. In: IDEAS, pp. 24:1–24:10. ACM (2020)
- Diressens, K., Ramon, J., Gärtner, T.: Graph kernels and gaussian processes for relational reinforcement learning. *Mach. Learn.* **64**(1–3), 91–119 (2006). <https://doi.org/10.1007/s10994-006-8258-y>
- Gärtner, T.: A survey of kernels for structured data. *SIGKDD* **5**(1), 49–58 (2003)
- Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: hardness results and efficient alternatives. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT-Kernel 2003. LNCS (LNAI), vol. 2777, pp. 129–143. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45167-9\\_11](https://doi.org/10.1007/978-3-540-45167-9_11)
- Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers’ stochastic conformance checking. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNBIP, vol. 360, pp. 127–143. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26643-1\\_8](https://doi.org/10.1007/978-3-030-26643-1_8)
- de Leoni, M., Marrella, A.: Aligning real process executions and prescriptive process models through automated planning. *Expert Syst. Appl.* **82**, 162–183 (2017)



9. 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)
10. Polyvyanyy, A., Kalenkova, A.A.: Monotone conformance checking for partially matching designed and observed processes. In: ICPM, pp. 81–88 (2019)
11. 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)
12. Seshadhri, C., Sharma, A., Stolman, A., Goel, A.: The impossibility of low-rank representations for triangle-rich complex networks. *Proc. Natl. Acad. Sci.* **117**(11), 5631–5637 (2020)