Earth Movers' Stochastic Conformance Checking

Sander J.J. Leemans[1], Anja F. Syring[2], and Wil M. P. van der Aalst[2]

[1] Queensland University of Technology, Brisbane, Australia
[2] Process and Data Science (Informatik 9),
RWTH Aachen University, D-52056 Aachen, Germany

Abstract. Process Mining aims to support Business Process Management (BPM) by extracting information about processes from real-life process executions recorded in event logs. In particular, conformance checking aims to measure the quality of a process model by quantifying differences between the model and an event log or another model. Even though event logs provide insights into the likelihood of observed behaviour, most state-of-the-art conformance checking techniques ignore this point of view. In this paper, we propose a conformance measure that considers the stochastic characteristics of both the event log and the process model. It is based on the "earth movers' distance" and measures the effort to transform the distributions of traces of the event log into the distribution of traces of the model. We formalize this intuitive conformance metric and provide an approximation and a simplified variant. The latter two have been implemented in ProM and we evaluate them using several real-life examples.

Keywords: Stochastic process mining · stochastic conformance checking · stochastic languages · stochastic Petri nets.

1 Introduction

Today's information systems provide an abundance of information about activities performed by or for customers, employees, machines etc. Databases and transaction files can be converted to event logs ready for analysis. Process mining aims to provide analysts with procedures and tools to obtain insights from these recorded event logs. For instance, *process models*, which describe the process steps in the process (*activities*), which activities are to be executed and in what order activities can be executed, are used to document and prescribe processes. A process model can be obtained manually, by human analysts modelling, or automatically, by process discovery algorithms using recorded event data [1].

Both human analysts and process discovery algorithms might leave out certain behaviour of the process to make the model more readable or to capture only the "happy flow". In order to not limit the model to seen behaviour only, they also include other behaviour and generalise the model in this way. Therefore, before drawing conclusions from a model, it should be evaluated using a conformance checking technique. Such a technique compares a process model with an event log and highlights their differences. Using conformance checking,

deviations between log and model can be unearthed, as well as differences between different versions of a business process, for instance the same process in geographic regions or different periods [8]. Furthermore, stochastic conformance checking is used to evaluate discovered process models and process discovery algorithms.

In typical real-life processes, not all parts of the processes are executed equally often: rarely executed exception handling routines or infrequent paths might be included in the model. Knowledge of the likelihood of such paths is necessary to gain insights into performance aspects of a business process, for instance to predict the remaining duration of a running trace [21] or, given a particular deadline, to estimate the probability of missing the deadline [6]. We refer to a process model that defines likelihoods for its traces as a *stochastic process model*. Such models can be automatically discovered from event logs by stochastic process discovery techniques, such as [13]. Consequently, *stochastic conformance checking* compares an event log with a stochastic process model and highlights their differences.

For instance, consider the event log $[\langle a, b\rangle^1, \langle b, a\rangle^{99}]$ and the stochastic process model expressing the stochastic language $[\langle a, b\rangle^{0.99}, \langle b, a\rangle^{0.01}]$. Even though only 2% of the log and model's stochastic language overlaps, any conformance checking technique that does not take the stochastic perspective into account will consider the log and model to have a perfect fitness and precision.

In this paper, we first propose an intuitive theoretical stochastic conformance checking measure. This measure is based on the earth movers' distance, that is, given two distributions (piles of earth), the effort to transform one pile into the other in terms of dirt that needs to be moved times the distance over which dirt has to be moved. This measure is defined for stochastic languages with possibly infinitely many traces (process models with loops may have infinitely many possible behaviours), but challenging to compute automatically for the general case. Therefore, second, we introduce an approximation and a simplification. We provide algorithms and implementations for both these last two measures, and illustrate their differences with conformance checking techniques that do not take probabilities into account.

We apply the measures to several real-life logs and automatically discovered stochastic process models to show their applicability.

In the remainder of this paper, we first explore related work in Section 2 and introduce concepts in Section 3. In Section 4, we introduce the three measures, after which we evaluate and analyse them in Section 5. Section 6 concludes the paper.

2 Related Work

Stochastic Process Formalisms. Several formalisms to describe stochastic languages have been proposed. Next to Stochastic Petri Nets (SPNs) and Generalised GSPNs which we will introduce in Section 3, several extensions have been proposed. For instance, Markov regenerative SPNs [19] and generally distributed transition SPNs [14] allow for the modelling of generally distributed timed events.

Fluid SPNs extend SPNs with continuous fluid quantities to model physical systems [20], and controlled SPNs extend SPNs for decision support purposes [12]. For a more elaborate overview, please refer to [5], in which several types of SPN are discussed, as well as the feasibility to compute their case-duration distribution. Several of the SPN types support inhibitor arcs, but silent transitions are not supported. Typically, stochastic Petri nets are used to express and compute the temporal perspective of business processes, while we focus on the combination of the control flow and stochastic perspectives of the traces in the model. However, none of these works considers SPNs with silent transitions.

An exception is [3], in which stochastic Petri nets with silent transitions are considered. In [3], a method is proposed to, given an SPN, (a set of) initial marking(s) and a trace, compute the possible markings the SPN can be in after executing the trace, and how likely each marking is.

In Section 3, we introduce our formalisation of Generalised Stochastic Labelled Petri Nets (GSLPN), which differs from the variants in the mentioned papers by including silent transitions. Even though our implementation targets GSLPNs, our measures work for any stochastic process model, as long as it represents a stochastic language.

Stochastic Conformance Checking. Conformance checking on non-stochastic process models has been addressed by many techniques (e.g. token-based replay and alignments, for an overview please refer to [4]). Such techniques typically consider two directions of inclusion: fitness (behaviour of the log is included in the model) and precision (behaviour of the model is included in the log). However, these notions of language inclusion do not apply to stochastic behaviour, as a stochastic language cannot include another stochastic language (for both languages, the probabilities of traces sum to 1). Therefore, a single similarity measure is more appropriate in a stochastic setting.

In [9], standard Petri nets are enriched with frequency information (that is, each transition gets a probability corresponding to the frequency of its label in the event log), and a most-probable alignment is computed in order to find the root cause of deviations. However, this approach is not intended to cope with arbitrary stochastic languages.

Hidden Markov Models (HMMs) have been used to model stochastic processes and to check conformance, for instance in [15]. HMMs express that transitions between states and the execution of activities in states can happen with certain probabilities. In Section 5.4.3 of [15], fitness and precision of non-stochastic Petri nets without concurrency are computed by translating the net to an HMM assuming that all transitions are equally likely. This approach might be applicable to verify the conformance of general stochastic languages as well, as long as the language can be expressed as an HMM.

3 Preliminaries

Stochastic Languages. Let Σ be a finite alphabet of *activities* (the different steps executed in a process) and Σ^* be the set of all possible sequences (*traces*) over the alphabet Σ. A *stochastic language* is a collection of traces with attached

probabilities. Formally, a stochastic language is a function $f\colon \Sigma^* \to [0,1]$ that maps each trace onto a probability such that $\sum_{t \in \Sigma^*} f(t) = 1$. For instance, $[\langle a,b,c\rangle^{\frac{2}{3}}, \langle a,c,b\rangle^{\frac{1}{3}}]$ is a stochastic language consisting of 2 traces, the first of which has a probability of $\frac{2}{3}$ and for which first an activity a was executed, followed by a b and a c. We denote the set of traces of a stochastic language M that have a nonzero probability with $\widetilde{M} = \{t \in \Sigma^* \mid M(t) > 0\}$.

Event Logs. An event log is a finite multiset of traces, which can be easily transformed into a stochastic language by normalising the trace quantities by dividing each trace's occurrences by the total number of traces. For instance, an event log consisting of 20 times $\langle a,b,c\rangle$ and 10 times $\langle b,a,c\rangle$ would have a corresponding stochastic language $[\langle a,b,c\rangle^{\frac{20}{30}}, \langle b,a,c\rangle^{\frac{10}{30}}]$. In this paper, we will use the term "event log" for the stochastic language belonging to an event log.

Earth Movers' Distance. The Earth Movers' Distance or Wasserstein distance describes the distance between two distributions [17]. In an analogy, given two piles of earth (the distributions), it expresses the effort required (in terms of quantity of earth and the horizontal distance it needs to be moved) to transform one pile into the other.

Stochastic Petri Nets. A *labelled Petri net* is a tuple (P, T, F, Σ, l), in which P is a finite set of places, T is a finite set of transitions, $F\colon (P \times T) \to (T \times P)$ is a flow relation, Σ is a finite alphabet of activities and $l\colon T \to \Sigma \cup \{\tau\}$ is a labelling function, such that $P \cap T = \emptyset$ and $\tau \notin \Sigma$. A *marking* is a multiset of places $\in P$, indicating the state of the net. A transition is *enabled* if each of its incoming places contains a token. When a transition *fires*, it changes the marking of the net by consuming and producing tokens to/from its connected places. If a transition $t \in T$ is labelled with an activity $l(t) = a \in \Sigma$, then a is executed whenever t fires. Note that multiple transitions might share a. A transition $t' \in T$ that is unlabelled ($l(t') = \tau$) is a *silent* transition: when t' fires, it may change the marking of the net but it does not correspond to the execution of an activity. A *path* through the model is a sequence of transition firings that starts in the *initial marking* and ends in a marking in which no transitions are enabled (a *deadlock*). That is, we consider each deadlock to be a final marking. The corresponding *trace* is the sequence of labelled transitions in a path.

A *Generalised Stochastic Labelled Petri Net* (GSLPN) is a tuple $(P, T, F, \Sigma, l, T_i, T_t)$ where (P, T, F, Σ, l) is a labelled Petri net, $T_i \subseteq T$ is a set of immediate transitions and $T_t \subseteq T$ is a set of timed transitions such that $T_i \cap T_t = \emptyset$. Immediate transitions $t \in T_i$ take precedence over $t' \in T_t$: timed transitions cannot fire if an immediate transition is enabled. A transition $t \in T_i$ is *immediate*, it has a *weight* $w(t)$ attached (this weight may depend on the marking) and if multiple transitions $T' \subseteq T_i$ are enabled, a transition t is chosen to fire with probability $w(t)/\sum_{t' \in T'} w(t')$. A timed transition t has an exponentially distributed waiting/enabling time, with firing rate parameter $\lambda(t)$. Due to the memory-less property of the exponential distribution, given a set of enabled

timed transitions $T' \subseteq T_t$, the probability that a particular transition t will fire first is $\lambda(t) / \sum_{t' \in T'} \lambda(t')$ [13].

The probability of a trace is defined as the sum of the probabilities over all paths through the model that produce the trace. Given a trace and a path through the model that produces the trace, then the probability of the trace is the product of the probabilities of the choices made in the model along the path. The stochastic language of an GSLPN is the weighted set of all traces through the model.

For instance, Figure 1 contains a GSLPN in which all transitions are immediate. The stochastic language of this model consists of two traces, $\langle a, b \rangle$ and $\langle a, c \rangle$. In the model, there are infinitely many paths resulting in these traces, and their probabilities are geometric series. For $\langle a, b \rangle$:

$$\frac{1}{2} + \frac{1}{2}\frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2} + \ldots = \sum_{n=0}^{\infty} \frac{1}{2}(\frac{1}{4})^n = \frac{\frac{1}{2}}{1 - \frac{1}{4}} = \frac{2}{3}$$

For $\langle a, c \rangle$:

$$\frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2} + \ldots = \sum_{n=0}^{\infty} \frac{1}{4}(\frac{1}{4})^n = \frac{\frac{1}{4}}{1 - \frac{1}{4}} = \frac{1}{3}$$

This example illustrates that for GSLPNs, it might be challenging to establish the stochastic language.[3] To the best of our knowledge, this challenge has not been solved yet for our GSLPNs (in particular, for silent transitions) [3,16], however an in-depth discussion is outside the scope of this paper.

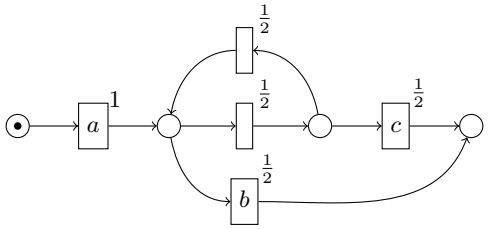


Fig. 1: Example of a generalised stochastic Petri net. All transitions are immediate.

Please note that livelocks, that is the inability to reach a final marking with nonzero probability, invalidate the stochastic language, as paths that enter the livelock with a certain probability p cannot terminate, thus the total probability of the corresponding stochastic language will be at most $1 - p$ rather than 1. Therefore, in this paper, we assume that GSLPNs do not have livelocks.

The firing of transitions in GSLPNs only depends on the current state of the model: immediate transitions are randomly chosen by weight and timed transition are exponentially distributed and hence memoryless. Therefore, GSLPNs satisfy the Markov property and can be translated to Markov chains.

[3] Notice that in case of duplicated labels, there might be exponentially, but finitely, many paths through the model for a particular trace.

4 Method

In this section, we introduce our measure for stochastic conformance checking. We first introduce the theoretical measure and illustrate it with a running example. Second, we provide a method to compute the measure.

4.1 Earth Movers' Stochastic Conformance

In this section, we transform the analogy of the Earth Movers' Distance to stochastic languages: our measure expresses the cost of transforming the distribution of traces of one language into the distribution of the other language.

First, we introduce the concept of reallocation functions, which indicate how a stochastic language is transformed into another stochastic language. Second, we introduce a distance function, which expresses the cost of transforming one trace into another trace. Third, we introduce a cost function that expresses the cost of a particular reallocation function. Finally, we define the measure and we give a variant that considers unit trace distances.

Reallocation. We first introduce a function that indicates the movement of probability mass between two stochastic languages. Let L and M be stochastic languages, then a *reallocation* function $r\colon \widetilde{L} \times \widetilde{M} \to [0,1]$ describes how L can be transformed into M. That is, $r(t,t')$ describes the probability mass of $t \in \widetilde{L}$ that should be moved to $t' \in \widetilde{M}$. The function $r(t,t)$ indicates the probability mass of $t \in \widetilde{L}$ that remains at $t \in \widetilde{M}$.

To ensure that a reallocation function properly transforms L into M the probability mass of each $t \in \widetilde{L}$ should be accounted for. Hence, the row for t should sum up to $L(t)$.

$$\forall_{t \in \widetilde{L}}\ L(t) = \sum_{t' \in \widetilde{M}} r(t,t') \tag{1}$$

Similarly, the mass of traces $t' \in M$ should be preserved:

$$\forall_{t' \in \widetilde{M}}\ M(t') = \sum_{t \in \widetilde{L}} r(t,t') \tag{2}$$

We refer to the set of all reallocation functions r that adhere to equations (1) and (2) as $\mathcal{R}$ (note that $\mathcal{R}$ depends on L and M).

For instance, consider the stochastic languages $L_e = [\langle a \rangle^{\frac{1}{4}}, \langle a,a \rangle^{\frac{3}{4}}]$ and $M_e = [\langle a \rangle^{\frac{1}{2}}, \langle a,a \rangle^{\frac{1}{4}}, \langle a,a,a \rangle^{\frac{1}{8}}, \langle a,a,a,a \rangle^{\frac{1}{16}} \ldots]$. An example reallocation function r_e is:

r_e	$\langle a \rangle$	$\langle a,a \rangle$	$\langle a,a,a \rangle$	$\langle a,a,a,a \rangle$	$\langle a,a,a,a,a \rangle$	$\ldots$
$\langle a \rangle$	$\frac{1}{4}$	0	0	0	0	$\ldots$
$\langle a,a \rangle$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\ldots$

In this tabular visualisation, Equation (1) states that each row should sum up to the corresponding value in language L_e (e.g. the first row sums up to $\frac{1}{4}$ as $L_e(\langle a \rangle) = \frac{1}{4}$). Similarly, Equation (2) expresses that each column should sum up to the corresponding mass in M_e.

Trace Distance. Second, a trace distance function d expresses the "distance" between traces: $d\colon \Sigma^* \times \Sigma^* \to [0,1]$. This function is 0 if and only if two traces are equal: $d(t,t') = 0 \Leftrightarrow t = t'$. Furthermore, this function is required to be symmetrical, that is $d(t,t') = d(t',t)$.

For example, we can use the normalised string edit (Levenshtein) distance. The Levenshtein distance expresses the minimum number of edit operations required to transform a trace into another trace using the event insertion, deletion and substitution operations [11]. As this distance has an upper bound in the number of events in the longest of the two traces, it can be normalised: we choose $d_l(t,t')$ to be the Levenshtein distance divided by the maximum length of t and t'.

For instance, consider our two stochastic languages L_e and M_e again. Then, the normalised Levenshtein distance d_l is:

	$\langle a \rangle$	$\langle a,a \rangle$	$\langle a,a,a \rangle$	$\langle a,a,a,a \rangle$	$\langle a,a,a,a,a \rangle$	...
$\langle a \rangle$	0	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{3}{4}$	$\frac{4}{5}$	...
$\langle a,a \rangle$	$\frac{1}{2}$	0	$\frac{1}{3}$	$\frac{2}{4}$	$\frac{3}{5}$	...

Cost. Given two stochastic languages, several reallocation functions might exist. However, the Earth Movers' Distance problem aims to express the *shortest* distance between the two languages, that is, the least probability mass movement over the least distance between traces.

Therefore, the cost to transform a stochastic language L into a stochastic language M using a reallocation function r is the inner product of reallocation and distance:

$$\text{cost}(r, L, M) = r \cdot d = \sum_{t \in \widetilde{L}} \sum_{t' \in \widetilde{M}} r(t,t') d(t,t') \tag{3}$$

By construction, d returns values between 0 and 1, hence $0 \leq \text{cost}(r, L, M) \leq 1$ for any r, L and M.

For instance, considering our example with L_e, M_e and r_e again, the cost of r_e given L_e and M_e is computed as follows:

$$\begin{aligned}
\text{cost}(r_e, L_e, M_e) &= \frac{1}{4} \cdot 0 + 0 \cdot \frac{1}{2} + 0 \cdot \frac{2}{3} + 0 \cdot \frac{3}{4} + 0 \cdot \frac{4}{5} + \dots \\
&\quad \frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot \frac{1}{3} + \frac{1}{16} \cdot \frac{2}{4} + \frac{1}{32} \cdot \frac{3}{5} + \dots \\
&= \frac{1}{8} + \sum_{n=3}^{\infty} \frac{n-2}{2^n \cdot n} = \frac{13}{8} - \log 4 \\
&\approx 0.238706
\end{aligned}$$

Earth Movers' Stochastic Conformance. Finally, the Earth Movers' Stochastic Conformance (EMSC) measure is defined as the lowest cost for any reallocation function r and given L and M. To align EMSC with existing conformance

checking measures, it is mirrored such that 1 indicates perfect conformance and 0 indicates the worst conformance.

$$\text{EMSC}(L, M) = 1 - \min_{r \in \mathcal{R}} \text{cost}(r, L, M) \tag{4}$$

In our running example, r_e is an optimal reallocation function, thus $\text{EMSC}(L_e, M_e) \approx 0.761294$.

Unit Distances. If we choose the trace distance function d differently, another version of EMSC appears: this function can also be chosen such that each pair of traces is either classified as equal with value 0 or as unequal with value 1. Intuitively, the Earth Movers' Distance expresses the amount of earth that has to be moved times the (possibly normalised) distance it has to be moved. The new choice for unit distances takes the distance out of the equation: the earth (traces) is either moved or not, but the distance over which it is moved is not taken into account.

Hence, in our conformance measure with unit distances, we only need to take into account how much probability mass of L is to be moved, and not where this probability mass will be put in M. This simplifies the reallocation function r considerably and removes the need for the minimisation step of Equation (4), yielding a much simpler measure for unit trace distances uEMSC:

$$\text{uEMSC}(L, M) = 1 - \sum_{t \in \widetilde{L}} \max(L(t) - M(t), 0) \tag{5}$$

Intuitively, the unit distance measure expresses the mass probability ("amount") of behaviour in L that is not supported or in surplus of the behaviour in M, without considering distances between traces.

If $\widetilde{L}$ is finite, for instance if L is an event log, then this sum is finite and can be computed without constructing the full stochastic language of M explicitly; M only needs to be queried for probabilities of traces that appear in L (which might still be challenging for GSLPNs as shown in Section 3).

4.2 Truncated Earth Movers' Stochastic Conformance

Computing EMSC measure in the previous section poses several challenges: the stochastic language might have infinitely many traces and many reallocation functions r might apply and need to be evaluated.

To address these challenges, we introduce a derivative measure *truncated* EMSC (tEMSC) that truncates infinite languages and searches for an optimal reallocation function. In the remainder of this section, we discuss how tEMSC addresses the two challenges, we analyse the new measure and we discuss its implementation.

Handling Models with Infinite Languages. If the stochastic language M has infinitely many traces, then Equation (3) has infinitely many terms, making EMSC challenging to compute in practice. To handle such a process model, we truncate

its language to only contain a certain user-chosen mass of probability (m). We chose to do this in a breadth-first prioritised fashion, consequently shorter likely traces tend to be included before longer unlikely traces.

As a side effect, Equation (2) does not hold for truncated stochastic languages, as the probabilities of traces in such a language do not sum up to 1. Therefore, this equation is weakened to:

$$\forall_{t' \in \widetilde{M}} \; M(t') \leq \sum_{t \in \widetilde{L}} r(t, t') \tag{6}$$

Let $\mathcal{R}'$ denote the set of all reallocation functions that adhere to equations (1) and (6). Furthermore, let M_m denote a truncated stochastic language of M such that at least m of M's probability mass is included: $\sum_{t \in M_m} M_m(t) \geq m$ and $\forall_{t \in M_m} M_m(t) \leq M(t)$. Then:

$$\text{tEMSC}(L, M, m) = 1 - \min_{r \in \mathcal{R}'} \text{cost}(r, L, M_m) \tag{7}$$

Then, by construction:

Corollary 1. *Let L and M be stochastic languages. Then, for m approaching 1,* tEMSC *and* EMSC *coincide:* $\text{EMSC}(L, M) = \lim_{m \to 1} \text{tEMSC}(L, M, m)$.

In a model-model comparison context, L might have infinitely many traces as well. Then, a symmetric argument applies, but an extra requirement on the reallocation function is necessary, that is, the sum of this function should be 1.

An alternative to weakening the equation is to normalise the probability mass of the model after truncation. However, as this alters the stochastic language which is an input for the conformance calculation, we discard this option.

Another option to determine the point of truncation could be to unfold the model until we included all traces that have the same length as the longest trace of the event log. For the included traces of the model, we calculate the reallocation cost as defined before. All traces part of the excluded probability mass are longer than all traces of the event log. Based on this, we assume all excluded traces of the model to be unfitting with distance 1. Therefore, this measure gives a lower bound to EMSC. However, we leave this option for future work.

Efficient Minimisation. Given a trace distance function different from the described unit costs, the goal is to obtain an optimal reallocation function that yields minimal costs. This can be achieved by solving a linear programming problem.

Linear programming is a technique that optimises a given linear function, the objective function, with respect to given linear constraints. Based on this objective function the linear programming algorithm finds the minimal/maximal value in the region of feasible solutions defined by the constraints [18].

To find the optimal reallocation function with minimal cost, we chose Equation (3) as our objective function. The constraints of the optimisation problem

are defined by equations (1) and (6). Hence, only reallocation functions that preserve the probability mass of each $t \in \widetilde{L}$ as well as the mass of $t' \in \widetilde{M}$ are valid. To complete the construction of the solution space, we define the reallocation $r(t, t')$ to be non-negative.

Considering our two stochastic languages L_e and M_e, and the distances given by the normalised Levenshtein distance function, the linear programming problem is constructed as follows:

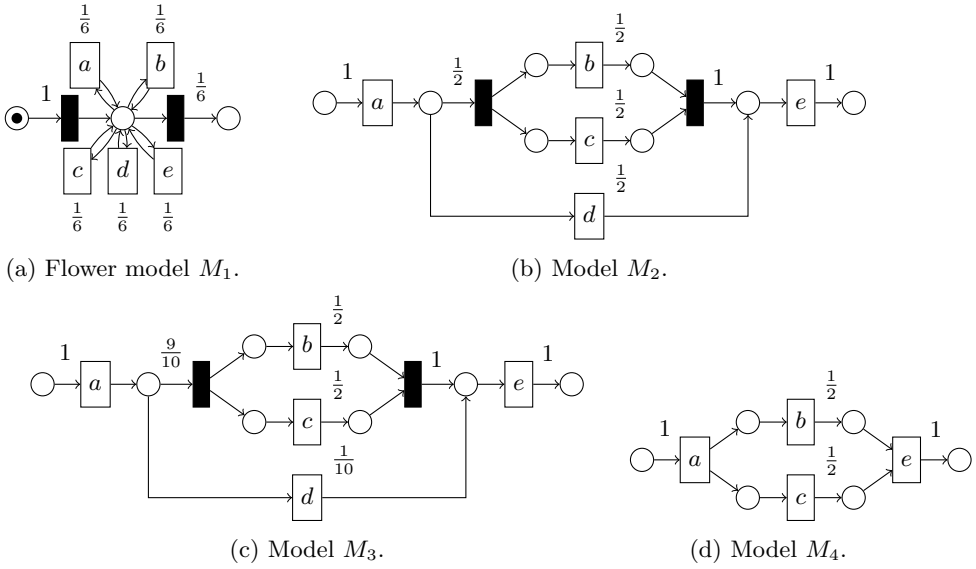
$$\begin{aligned}
\text{Minimise } & r(t_1, t'_1) \cdot 0 + r(t_1, t'_2) \cdot \frac{1}{2} + r(t_1, t'_3) \cdot \frac{2}{3} + \ldots, \\
\text{Subject to } & \frac{1}{4} \leq r(t_1, t'_1) + r(t_1, t'_2) + r(t_1, t'_3) + \ldots, \\
& \ldots \\
& \frac{1}{2} \leq r(t_1, t'_1) + r(t_2, t'_1), \\
& \frac{1}{4} \leq r(t_1, t'_2) + r(t_2, t'_2), \\
& \ldots \\
& 0 \leq r(t_1, t'_1), r(t_1, t'_2), r(t_1, t'_3) \ldots
\end{aligned}$$

4.3 Example & Implementation

Example. Consider the following event logs: $L_1 = [\langle a, b, c, e\rangle^{0.25}, \langle a, c, b, e\rangle^{0.25}, \langle a, d, e\rangle^{0.5}]$ and $L_2 = [\langle a, b, c, e\rangle^{0.45}, \langle a, c, b, e\rangle^{0.45}, \langle a, d, e\rangle^{0.1}]$. Furthermore, consider the GSLPNs shown in Figure 2, and the corresponding tEMSC and uEMSC measures in Table 1. Model M_1 is a model that supports any behaviour (a *flower* model), with uniform probabilities of the individual activities and ending the trace. Intuitively, this model differs considerably from both logs L_1 and L_2, which is reflected in the low measures. Models M_2 and M_3 have the same language, but their stochastic perspective differs markedly. A conformance checking technique that is not stochastic aware would consider these models to be equivalent, resulting in equivalent measures, even though their behaviour is very different from a stochastic perspective: d is much less likely in M_3. Intuitively, M_2 is closer to L_1 than to M_3, (which is closer to L_2) and this is reflected in all our measures. Finally, from M_4 the trace $\langle a, d, e\rangle$ is missing, and, accordingly, the measures being lower for L_1 than for L_2 indicates that this trace had a higher probability in L_1.

With respect to tEMSC and uEMSC, observe that both measures are consistent in their ranking of the logs and models.

Implementation. Both tEMSC and uEMSC have been implemented as plug-ins of the ProM framework [7]. The plug-in of tEMSC takes an event log and an GSLPN (as returned by a stochastic process discovery technique [13]), and constructs the full stochastic language of the log, as well as the truncated stochastic language of the GSLPN, with a user-specified m. Second, a linear programming



(a) Flower model M_1.

(b) Model M_2.

(c) Model M_3.

(d) Model M_4.

Fig. 2: Four example GSLPNs, in which all transitions are immediate.

Table 1: Our measures on the example logs and models.

model	tEMSC ($m = 0.8$)		uEMSC	
	L_1	L_2	L_1	L_2
M_1	0.46	0.45	0.0010288065843622	0.0010288065843621
M_2	1	0.8	1	0.6
M_3	0.8	1.0	0.6	1.0
M_4	0.75	0.95	0.5	0.9

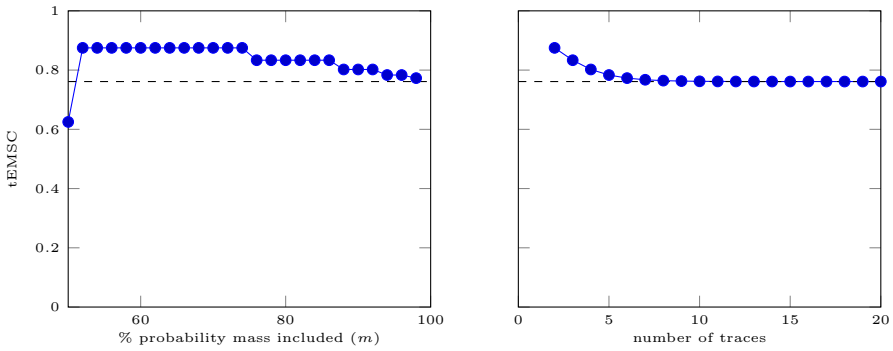
problem is constructed and solved using the LpSolve library [2]. For uEMSC, the GSLPN is not allowed to have executable loops of silent transitions (see Section 3). The source code of both plug-ins is available at https://svn.win.tue.nl/repos/prom/Packages/EarthMoversStochasticConformanceChecking/Trunk/.

5 Evaluation

In this section, we evaluate the newly introduced measures: the theoretical EMSC, the truncated tEMSC and the unit-distance uEMSC. First, we illustrate these on our running example. Second, we apply the measures to real-life logs to show their applicability.

Example. For illustrative purposes, we apply tEMSC and uEMSC to our running example consisting of the stochastic languages $L_e = [\langle a \rangle^{\frac{1}{4}}, \langle a, a \rangle^{\frac{3}{4}}]$ and

$M_e = [\langle a \rangle^{\frac{1}{2}}, \langle a, a \rangle^{\frac{1}{4}}, \langle a, a, a \rangle^{\frac{1}{8}}, \langle a, a, a, a \rangle^{\frac{1}{16}} \ldots]$ to show the influence of truncating on a simple loop. We apply tEMSC to L_e and M_e for increasing m.



(a) varying parameter (m).

(b) varying number of traces with increasing loop unfoldings.

Fig. 3: tEMSC measured over our example L_e and M_e. The dashed lines indicate the EMSC values.

Figure 3a shows the results, as well as the theoretical EMSC value for L_e and M_e. After an initial climb when the two traces of the log are not represented by the truncated model, tEMSC temporarily stabilises at 0.875. This stable range of m indicates that the truncation includes more probability mass than m, that is, if we choose $m = 52\%$ then the truncation nevertheless includes 74% of the probability mass. Only at $m = 76\%$, another trace is included in the truncated language. After this point, the truncation includes more and more traces that are not in the event log, thus tEMSC drops and seems to approach the theoretical EMSC value shown in Section 4.

To illustrate the convergence to EMSC, we repeat the experiment where we manually create stochastic languages for L_e and M_e, where we unfold the loop of M_e an increasing number of times. Figure 3b shows these results. From this graph, it is clear that tEMSC quickly converges to the theoretical EMSC value with every trace and loop unfolding added. At 6 traces, which corresponds to $m = 98\%$, the difference is a negligible 0.01.

The run time for these examples was too small to warrant any conclusion.

Real-Life Event Logs. In this experiment, we evaluate the applicability of tEMSC and uEMSC to 16 publicly available real-life logs and stochastic process models. First, we apply the Stochastic Miner (SM) [13] to these logs to obtain stochastic Petri nets. As these nets contain silent transitions, they can be seen as GSLPNs (Section 3). Table 2 shows the logs and their complexity. Furthermore, it shows that a GSLPN was discovered for only 6 logs in the 24BGB of RAM we had available, which illustrates the need for more research into stochastic process discovery techniques and their implementations.

Table 2: Real-life event logs used in the evaluation.

	activities	traces	events	discovery [13]	uEMSC	rank
BPIC12	36	13087	262200	out of memory		
BPIC15-1	398	1199	52217	out of memory		
BPIC15-2	410	832	44354	out of memory		
BPIC15-3	383	1409	59681	out of memory		
BPIC15-4	356	1053	47293	out of memory		
BPIC15-5	389	1156	59083	out of memory		
BPIC18 Control summary	7	43808	161296	✓	0.599	1
BPIC18 Department control	6	29297	46669	✓	0.120	2
BPIC18 Entitlement application	20	15620	293245	out of memory		
BPIC18 Geo parcel documents	16	29059	569209	out of memory		
BPIC18 Inspection	15	5485	197717	out of memory		
BPIC18 Parcel document	10	14750	132963	✓	$1.15 \cdot 10^{-4}$	5
BPIC18 Payment application	24	43809	984613	out of memory		
BPIC18 Reference alignment	6	43802	128554	✓	$0.22 \cdot 10^{-2}$	3
Road Traffic Fines	11	150370	561470	✓	$2.88 \cdot 10^{-4}$	4
Sepsis	16	1050	15214	✓	$1.52 \cdot 10^{-14}$	6

Second, we apply our new measures to the logs and the six discovered GSLPNs. For tEMSC, we use various parameters m (see Equation (7)) to study how the inclusion of mass influences the returned values. In the remainder of this section, we first discuss run times, then the results of tEMSC followed by the results of uEMSC.

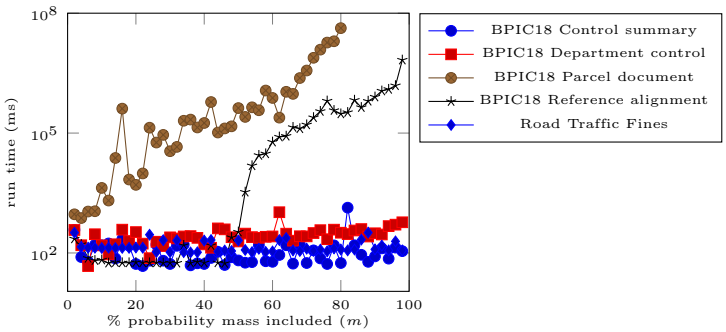


Fig. 4: Run time of tEMSC for several real-life logs.

Run Time. The run times of uEMSC were negligible: all measures finished within a second. For tEMSC, run times are shown in Figure 4. Please note that y-axis is logarithmic, and that due to the inherent nondeterministic nature of tEMSC and the multithreadedness of the implementation, these measures are indicative only, especially for lower m.

Some of the values could not be obtained: for `BPIC18 Parcel Document` with $m = 80$, the linear programming optimisation ran out of memory, while for `Sepsis` with any m, the explicit creation and truncation of the stochastic language ran out of memory. For 4 out of 6 logs, computation took less than a few seconds, which was considerably less than the discovery technique, which could take hours on these logs. However, a general trend towards longer run times is visible as m approaches 100%. For `BPIC18 Parcel document` and `BPIC18 Reference alignment`, computation could take up to little over an hour. A manual inspection revealed that this is caused by the size of the language described in the stochastic models: especially in models that combine concurrency with looping behaviour. In such models, the probability mass per trace

decreases and more traces are necessary to cover a certain probability mass, which makes it very challenging to obtain a high probability mass m. This is a clear limitation of the current technique, which might be addressed in future work.

*Truncated EMSC (*tEMSC*).* Second, we discuss the returned values of tEMSC, which are shown in Figure 5 for m (that is, the minimum probability mass covered in the truncation step) varying from 2% to 98%.

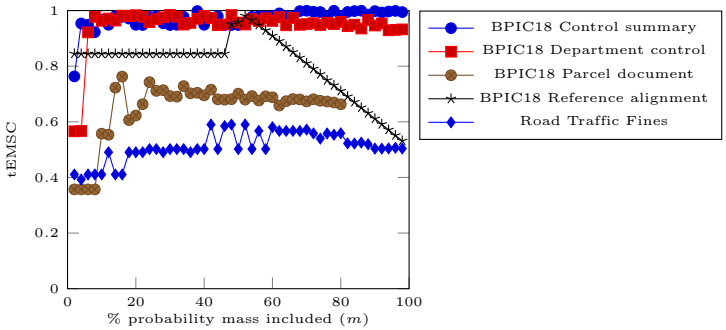


Fig. 5: tEMSC with varying mass truncation parameter (m).

Most measures show expected behaviour with increasing m: due to nondeterminism of the truncation, for lower m they vary considerably, but stabilise with m approaching 100%. An exception is `BPIC18 Reference alignment`, which increases to 0.98 for $m = 52$, after which it decreases almost linearly. A manual inspection revealed that the GSLPN contains many loops, while most of the event log's traces do not exhibit repeating activities. As m increases, more traces are added by unfolding loops and, as these new traces are not in the log, the measured tEMSC drops.

*Unit-distance EMSC (*uEMSC*).* The results for uEMSC are shown in Table 2. As identified in Section 3, loops of silent transitions challenge the computation of the probability of a trace in an GSLPN ($M(t)$ in Equation (5)). A manual inspection revealed that none of the discovered models contained such loops. However, most models contained lots of concurrency, which makes the state space of the model *huge* and thus the probability of individual traces in the model is very low, leading to low uEMSC measures.

If the use case at hand involves choosing a stochastic process model that best represents a given event log, then uEMSC and tEMSC mostly agree: of the 10 possible pairs of models (out of which the model closest to the log is to be chosen), 8 times uEMSC and tEMSC agree on which model is the closest.

Reproducibility. The experiments were performed on a single machine with 3.5GHz quadcore CPU and 24GB RAM available for each experiment process, running fully patched Windows 7 in January 2019. The source code is available.

6 Conclusion

Recently the interest in stochastic-aware conformance checking increased within the process mining community. Despite a larger awareness about the importance of a stochastic view on the process model, to this day there are only a few conformance checking techniques that consider the stochastic characteristics of both event log and model. This paper, however, presents conformance checking measures that compare the stochastic languages of event logs and process models. In essence, this is achieved by measuring "how much probability mass that has to be moved how far" to transform one language into the other. We introduced three variations of the measure: a theoretical and two adapted versions, which are also feasible on process models with an infinite set of traces.

These adapted variants were implemented and their practical relevance was illustrated on real-life event logs. The experiments showed the influence of the probability mass parameter on the run time and the results of the measure. The evaluation showed the trade-offs between run time and memory usage, and accuracy, and it would be interesting for future work to inspect different strategies of choosing this parameter. Additionally, it would be interesting to investigate the influence of different distance functions, since the current work only compares the Levenshtein distance to a simple unit distance. However, the measure could easily be extended to use other distance functions as well.

Although EMSC simply requires two stochastic languages, its implementation starts from an event log and a stochastic Petri net. Currently, it is challenging to establish the language of a GSLPN as well as calculating the probability of a single trace in the net. Especially loops of silent transitions are shown to be problematic. Extending the technique with a new method which solves this problem will improve the reliability of the measure. Furthermore, for models with a large state space where each trace only has a low probability, the technique would benefit from a more efficient truncation implementation. Searching the model until the required probability mass has been collected is a time critical part of the measure.

In [9], a technique to calculate the most probable alignment between a model and a log is proposed, based on the probabilities of behaviour observed in the event log. For future work, it would be interesting to incorporate their technique in our reallocation function. Instead of achieving the result with the lowest cost, the algorithm would aim for the most probable reallocation.

In general, with this paper we want to stress the importance of stochastic-aware process mining and hope to inspire more discussion and contributions on this topic, for instance on the need for dual (recall/precision) measures vs. single measures or on dependencies of choices in models.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer (2016)
2. Berkelaar, M., Eikland, K., Notebaert, P.: lp_solve 5.5. Software (May 1 2004)
3. Cabasino, M.P., Hadjicostis, C.N., Seatzu, C.: Probabilistic marking estimation in labeled Petri nets. IEEE Trans. Automat. Contr. **60**(2), 528–533 (2015)

4. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
5. Ciardo, G., German, R., Lindemann, C.: A characterization of the stochastic process underlying a stochastic Petri net. IEEE TSE **20**(7), 506–515 (1994)
6. Conforti, R., Fink, S., Manderscheid, J., Röglinger, M.: PRISM - A predictive risk monitoring approach for business processes. In: BPM. pp. 383–400 (2016)
7. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: Petri Nets. pp. 444–454 (2005)
8. van Eck, M.L., Lu, X., Leemans, S.J.J., van der Aalst, W.M.P.: PM ˆ2 : A process mining project methodology. In: CAiSE. pp. 297–313 (2015)
9. Koorneef, M., Solti, A., Leopold, H., Reijers, H.A.: Automatic root cause identification using most probable alignments. In: BPM Workshops. pp. 204–215 (2017)
10. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware conformance checking: Properties and measures. Information Systems **submitted**, http://leemans.ch/publications/papers/is2019leemans.pdf
11. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
12. de Meer, H., Düsterhöft, O.: Controlled stochastic Petri nets. In: SRDS. pp. 18–25 (1997)
13. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In: BPM Workshops. pp. 15–27 (2013)
14. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic Petri nets. Inf. Syst. **54**, 1–14 (2015)
15. Rozinat, A.: Process mining : conformance and extension. Ph.D. thesis, Ph. D. thesis, Eindhoven University of Technology (2010)
16. Ru, Y., Hadjicostis, C.N.: Bounds on the number of markings consistent with label observations in petri nets. IEEE Trans. Automation Science and Engineering **6**(2), 334–344 (2009)
17. Rüschendorf, L.: The Wasserstein distance and approximation theorems. Probability Theory and Related Fields **70**(1), 117–129 (1985)
18. Sierksma, G.: Linear and integer optimization : theory and practice. Chapman & Hall/CRC, Boca Raton (2015)
19. Trivedi, K.S., Puliafito, A., Logothetis, D.: From stochastic Petri nets to Markov regenerative stochastic Petri nets. In: MASCOTS. pp. 194–198 (1995)
20. Tuffin, B., Chen, D.S., Trivedi, K.S.: Comparison of hybrid systems and fluid stochastic Petri nets. Discrete Event Dynamic Systems **11**(1-2), 77–95 (2001)
21. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. CoRR **1805.02896** (2018)