Author/s:
Leemans, SJJ;van der Aalst, WMP;Brockhoff, T;Polyvyanyy, A

Title:
Stochastic process mining: Earth movers' stochastic conformance

Date:
2021-02

Citation:
Leemans, S. J. J., van der Aalst, W. M. P., Brockhoff, T. & Polyvyanyy, A. (2021). Stochastic process mining: Earth movers' stochastic conformance. Information Systems, 102, https://doi.org/10.1016/j.is.2021.101724.

Persistent Link:
https://hdl.handle.net/11343/261060

Stochastic Process Mining: Earth Movers' Stochastic Conformance

Sander J.J. Leemans[a,*], Wil M.P. van der Aalst[b], Tobias Brockhoff[b], Artem Polyvyanyy[c]

[a] *Queensland University of Technology, Brisbane, Australia*
[b] *Rheinisch-Westfälische Technische Hochschule, Aachen, Germany*
[c] *The University of Melbourne, Australia*

Abstract

Initially, process mining focused on discovering process models from event data, but in recent years the use and importance of conformance checking has increased. Conformance checking aims to uncover differences between a process model and an event log. Many conformance checking techniques and measures have been proposed. Typically, these take into account the frequencies of traces in the event log, but do not consider the probabilities of these traces in the model. This asymmetry leads to various complications. Therefore, we define conformance for stochastic process models taking into account frequencies and routing probabilities. We use the earth movers' distance between stochastic languages representing models and logs as an intuitive conformance notion. In this paper, we show that this form of stochastic conformance checking enables detailed diagnostics projected on both model and log. To pinpoint differences and relate these to specific model elements, we extend the so-called 'reallocation matrix' to consider paths. The approach has been implemented in ProM and our evaluations show that stochastic conformance checking is possible in real-life settings.

Keywords: process mining, conformance checking, stochastic process mining

1. Introduction

Process mining aims to analyse event data in a process-centric manner and can be used to identify, predict and to address performance and compliance problems [2]. The uptake of process mining in industry has accelerated in recent years. Currently, there are more than 35 commercial offerings of process mining software (for instance Celonis, Disco, ProcessGold, myInvenio, PAFnow, Minit, QPR, Mehrwerk, Puzzledata, LanaLabs, StereoLogic, Everflow, TimelinePI,

*Corresponding author

Signavio and Logpickr). These products still focus on process discovery. However, the perceived importance of *conformance checking* is clearly growing (see for example the recent surveys by Gartner [28]).

Conformance checking techniques aim to compare *observed behaviour* in the form of an event log with *modelled behaviour*. Models may be expressed using BPMN, transition systems, Petri nets, process trees, statecharts, etc. Such models may have been made by hand or learned from event data using process discovery techniques. The first comprehensive conformance checking techniques used *token-based replay* in order to count produced, consumed, missing and remaining tokens [51]. Over the last decade, *alignment-based techniques* replaced token-based replay in process mining research. Alignments are used to directly relate observed traces to the corresponding closest paths through the model [4, 16]. Many conformance measures have been proposed throughout the years [2, 4, 8, 16, 19, 20, 23, 35, 38, 51, 58, 59]. These cover different quality dimensions. In [2], four major quality dimensions were identified: recall, precision, generalisation, and simplicity. Most of the conformance measures focus on the first two dimensions. *Recall measures* quantify the fraction of the log that "fits" the model. This intuitive notion can be operationalised in different ways, e.g., the percentage of observed traces that can be generated by the model or the number of missing and remaining tokens during replay [2, 4, 16]. *Precision measures* complement recall. Precision aims to quantify the fraction of modelled behaviour that was actually observed in the event log. Many precision notions have been proposed, but, unfortunately, most turned out to be problematic [3, 55]. There are several reasons for this. Here, we name the two most important problems encountered using most of the measures. First of all, a model with loops allows for infinitely many traces making it difficult to define the "fraction" of observed behaviour (i.e., the observed percentage of modelled traces is zero by definition). Second, precision depends on recall. When many of the observed traces are not fitting, we cannot talk about precision in a meaningful way (i.e., precision is not orthogonal to but is influenced by recall).

Intuitively, most measures reason in terms of the "fraction of observed behaviour" and the "fraction of modelled behaviour". The first fraction is easy to quantify, because the event log is finite and observed behaviours have a frequency. The second fraction is difficult to define, leading to the two problems related to precision mentioned before. How to talk about the "fraction of modelled behaviour" covered by the event data when the number of possible traces is infinite or many observed traces are non-fitting? Note that the event log contains only a sample of behaviour and it is odd to assume that for precision one would need to see all possible behaviour. The analysis in this paper shows that the *absence of probabilities in the process model are a direct cause for these problems*. Adding probabilities allows us to better reason about "fraction of modelled behaviour" covered by the event log. This paper shows that these problems indeed disappear when using probabilistic models. Therefore, we advocate the use of *stochastic conformance checking* and provide a new approach to compare event logs and process models.

This paper extends the work presented in [34] where we introduced the first

stochastic conformance checking technique. In [34], we proposed a measure for quantifying the difference between a process model having probabilities and a standard event log capturing the frequencies of traces. Both the process model and the event log are mapped onto *stochastic languages* that are then compared using the so-called *earth movers' distance* (EMD). Using EMD we can quantify the difference between a model and a log in a more objective manner. However, in [34] we only presented the EMD-based measure *without providing diagnostics*. In this paper, we extend our approach to also provide diagnostics projected onto the event log and the process model. We annotate logs and models with information explaining the differences. This will help to diagnose deviations. To this end, we added the new concept of *stochastic trace alignments* that relates observed traces to paths in the model in detail, and added projections of these trace alignments on the event log and on the model. Furthermore, the method now supports duplicate activity labels and unlabelled steps in process models. The novel EMD-based conformance checking technique based on the extended reallocation matrix has been implemented as a *ProM* plug-in and can be obtained by downloading the *Earth Movers' Stochastic Conformance Checking* package from `http://promtools.org`.

We believe that it is important to consider the stochastic perspective as a *first-class citizen* for the following reasons:

- Current *conformance checking* techniques are *asymmetric*, because the frequencies of traces in the event log are taken into account without having a comparable notion on the model side. This causes foundational problems when defining for example precision (handling loops and event logs that are relatively small or that contain deviating behaviour). As a result, conformance checking measures and diagnostics tend to be misleading. Consequently, process discovery techniques cannot be compared properly.

- Another reason to include the stochastic perspective is the obvious link to *simulation*, *prediction* and *recommendation* [2]. Simulation, prediction and recommendation models inherently require probabilities. For example, to predict the remaining process time of a running case, one needs to know the likelihood of the different paths the case can take. Also, in simulation models, we need to assign probabilities to choices [52]. Therefore, the quality of a process model is not only determined by its control-flow structure but also by its *stochastic perspective*.

Event logs typically have a clear Pareto distribution: it is quite common that less than 20% of the trace variants cover over 80% of the traces in the event log. In an event log with thousands of traces, a deviating trace variant that happened hundreds of times is clearly more severe than a deviating trace variant that happened only once. When models have no probabilities, the decision to include additional, less likely, paths in the model may have devastating effects on precision. If the model distinguishes between "highways" (paths in the model that have a high probability) and "dirt roads" (paths in the model that have a low probability), then it is less severe that a dirt road is not observed in the

event log. However, if highways are completely missing in the event log, then this is more severe. Conversely, the decision to include a dirt road in the model or not should have limited impact on conformance.

Existing conformance techniques are highly sensitive to what is included in the model and what not. Using existing measures a model may seem similar to the actual process, but is not. Conversely, the model may seem very different, but is actually very close to what is observed.

Stochastic conformance checking assumes the presence of probabilities in process models. Existing models typically do not have such probabilities. Fortunately, by using replay techniques, it is relatively easy to add probabilities to process models [48, 52]. These can be used as a first estimate and should be refined by domain experts after seeing conformance diagnostics. Given the problems mentioned, we feel that modellers should add probabilities to process models and that discovery techniques should directly return models with probabilities.

This paper extends [34] with support for silent and duplicate activities (e.g., skipping parts of the model or activities occurring in different parts of the process), detailed log- and model-based diagnostics, based on the new concepts of stochastic trace alignments. Furthermore, a new formally-proven more efficient implementation was added, and the evaluation was extended with several case studies.

The remainder of this paper is organised as follows. We start by providing a small motivating example in Section 2. Section 3 discusses related work. Notions such as event logs, process models, and stochastic languages are introduced in Section 4. Section 5 introduces the Earth Mover's Stochastic Conformance (EMSC) notion to compare stochastic languages and presents the reallocation matrix. Based on this, stochastic trace alignments are computed, which serve as input for diagnostics projected on the event log and process model. Section 6 evaluates the approach which has been implemented in ProM. Section 7 discusses applications and open challenges. Section 8 concludes the paper.

2. Motivating example

To motivate the need for *stochastic conformance checking*, we use a small toy example. Consider the process model in Figure 1 and the following five event logs:

$$L_1 = [\langle a,b,d,e\rangle^{490}, \langle a,d,b,e\rangle^{490}, \langle a,c,d,e\rangle^{10}, \langle a,d,c,e\rangle^{10}]$$
$$L_2 = [\langle a,b,d,e\rangle^{245}, \langle a,d,b,e\rangle^{245}, \langle a,c,d,e\rangle^{5}, \langle a,d,c,e\rangle^{5}, \langle a,b,e\rangle^{500}]$$
$$L_3 = [\langle a,b,d,e\rangle^{489}, \langle a,d,b,e\rangle^{489}, \langle a,c,d,e\rangle^{10}, \langle a,d,c,e\rangle^{10}, \langle a,b,e\rangle^{2}]$$
$$L_4 = [\langle a,b,d,e\rangle^{500}, \langle a,d,b,e\rangle^{500}]$$
$$L_5 = [\langle a,c,d,e\rangle^{500}, \langle a,d,c,e\rangle^{500}]$$

The process model is expressed in terms of an *accepting Petri net* with an initial marking $[p_1]$ and a final marking $[p_6]$. There are four possible traces starting in

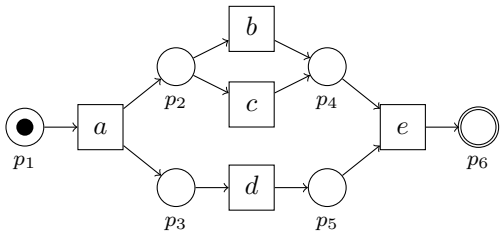


Figure 1: An accepting Petri net allowing for the traces $\langle a, b, d, e\rangle$, $\langle a, d, b, e\rangle$, $\langle a, c, d, e\rangle$, and $\langle a, d, c, e\rangle$.

$[p_1]$ and ending in $[p_6]$: $\langle a, b, d, e\rangle$, $\langle a, d, b, e\rangle$, $\langle a, c, d, e\rangle$, and $\langle a, d, c, e\rangle$. These describe the model's behaviour. Note that we allow for Petri nets with duplicate and silent transitions. In fact, all notions defined in this paper are independent of the representation and can be used for any modelling language. We only use Petri nets to provide a graphical representation of model behaviour and to operationalise our approach (e.g., in conjunction with existing discovery techniques). The event logs are multisets of traces and each trace is a sequence of activities. Trace $\langle a, b, d, e\rangle$ models the execution of activities a, b, d, and e. This trace occurs 490 times in event log L_1 and 245 times in event log L_2. Each of the five event logs describes 1000 traces (to facilitate comparison).

Each trace in L_1 matches a trace of the model in Figure 1 and vice versa. Hence, all existing recall and precision measures tend to give a high score (i.e., good conformance). Half of the traces in L_2 do not fit the model ($\langle a, b, e\rangle$ is impossible according to the model in Figure 1, but occurs 500 times). Hence, all existing recall measures will report a low recall score for L_2. However, these measures will report a high score for recall when L_3 is considered. The reason is that in L_3, 99.8% of the traces are fitting ($\langle a, b, e\rangle$ occurs only twice). Existing recall measures tend to give high scores when L_4 and L_5 are considered since the model can reproduce all traces observed. However, both L_4 and L_5 are only covering two of the four traces allowed by the process model in Figure 1. Hence, existing precision measures tend to give a lower score for L_4 and L_5. Moreover, due to symmetry, there is no reason to consider L_4 and L_5 to be different from a precision point of view.

The above analysis of existing recall measures shows that *frequencies matter*. L_2 and L_3 have the same sets of traces, but 50% of the traces of L_2 are fitting and 99.8% of the traces of L_3 are fitting. Hence, most recall measures will consider L_3 to conform much better than L_2. The logical counterpart of frequencies in event logs are routing probabilities in process models. However, almost all existing measures ignore such routing probabilities. This leads to an asymmetry. In this paper, we argue that also *probabilities matter*. To illustrate this, we add probabilities to our accepting Petri net.

Consider the simplified *stochastic* process model in Figure 2. The numbers attached to transitions can be interpreted as *weights*. Assume a has occurred resulting in the marking $[p_2, p_3]$ enabling b, c, and d. The weights of the three

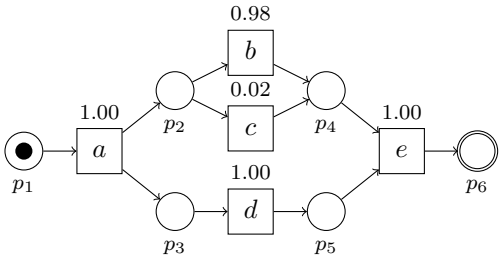


Figure 2: A stochastic Petri net defining the stochastic language $M = [\langle a,b,d,e\rangle^{0.49}, \langle a,d,b,e\rangle^{0.49}, \langle a,c,d,e\rangle^{0.01}, \langle a,d,c,e\rangle^{0.01}]$.

enabled transitions are respectively 0.98, 0.02, and 1.00. This means that b occurs first with probability $\frac{0.98}{0.98+0.02+1.00} = 0.49$, c occurs first with probability $\frac{0.02}{0.98+0.02+1.00} = 0.01$, and d occurs first with probability $\frac{1.00}{0.98+0.02+1.00} = 0.5$. Suppose that d occurs resulting in the marking $[p_2, p_5]$ enabling b and c. The weights of the enabled transitions are 0.98 and 0.02. This means that b occurs first with probability $\frac{0.98}{0.98+0.02} = 0.98$ and c occurs first with probability $\frac{0.02}{0.98+0.02} = 0.02$. Hence, the probability of trace $\langle a,d,b,e\rangle$ is $0.5 \times 0.98 = 0.49$, the probability of trace $\langle a,d,c,e\rangle$ is $0.5 \times 0.02 = 0.01$, etc. A stochastic language assigns probabilities to traces. For our example, $M = [\langle a,b,d,e\rangle^{0.49}, \langle a,d,b,e\rangle^{0.49}, \langle a,c,d,e\rangle^{0.01}, \langle a,d,c,e\rangle^{0.01}]$ is the stochastic language of the model in Figure 2. The stochastic process model in Figure 2 does not show timing information. However, the ordering of concurrent activities is resolved by time (d is "racing" against b and c). For example, in a Generalised Stochastic Petri Net (GSPN) both timed and immediate transitions can be used [10]. Immediate transitions have priority over timed transitions and do not take any time. A GSPN can be converted into an embedded Markov chain providing probabilities for each path in the model. Hence, we can abstract from time and focus on trace probabilities. Actually, we use so-called *generalised stochastic labelled Petri nets* which extend traditional GSPNs with a labelling function to allow for silent and duplicate activities (while abstracting from time durations and focusing on probabilities).

We can convert an event log into a stochastic language by dividing the frequency of each trace by the overall number of traces. For our five example logs, we obtain the following stochastic languages:

$L_1 = [\langle a,b,d,e\rangle^{0.49}, \langle a,d,b,e\rangle^{0.49}, \langle a,c,d,e\rangle^{0.01}, \langle a,d,c,e\rangle^{0.01}]$

$L_2 = [\langle a,b,d,e\rangle^{0.245}, \langle a,d,b,e\rangle^{0.245}, \langle a,c,d,e\rangle^{0.005}, \langle a,d,c,e\rangle^{0.005}, \langle a,b,e\rangle^{0.5}]$

$L_3 = [\langle a,b,d,e\rangle^{0.489}, \langle a,d,b,e\rangle^{0.489}, \langle a,c,d,e\rangle^{0.01}, \langle a,d,c,e\rangle^{0.01}, \langle a,b,e\rangle^{0.002}]$

$L_4 = [\langle a,b,d,e\rangle^{0.5}, \langle a,d,b,e\rangle^{0.5}]$

$L_5 = [\langle a,c,d,e\rangle^{0.5}, \langle a,d,c,e\rangle^{0.5}]$

By converting event logs and process models to stochastic languages, conformance is reduced to the problem of comparing stochastic languages. Consider

$M = [\langle a, b, d, e\rangle^{0.49}, \langle a, d, b, e\rangle^{0.49}, \langle a, c, d, e\rangle^{0.01}, \langle a, d, c, e\rangle^{0.01}]$ and the five event logs L_1, L_2, L_3, L_4, and L_5. Obviously, L_3 is closer to M than L_2. Similarly, L_4 is closer to M than L_5. In [34], we proposed to use the so-called *earth movers' distance* (EMD) to compare stochastic languages. If the probabilities of traces are considered as piles of sand, then EMD is the minimum cost of moving the sand from one distribution to another. EMD requires a distance notion. For our Earth Movers' Stochastic Conformance (EMSC) notion [34], we provided several distance notions, e.g., the edit distance between two traces.

If we assume the normalised edit distance between traces, then the EMD distance is a number between 0 (identical, i.e., fully conforming) and 1 (worst possible conformance). For our model M and logs $L_1, L_2, \ldots, L_5$ we find the following distances: 0 for L_1, 0.125 for L_2, 0.0005 for L_3, 0.005 for L_4, and 0.245 for L_5. Note that distance is the inverse of similarity, i.e., for model M and logs $L_1, L_2, \ldots, L_5$ we find the following EMSC similarity measures: 1 for L_1, 0.875 for L_2, 0.9995 for L_3, 0.995 for L_4, and 0.755 for L_5. Hence, given M, L_1 has the best conformance, L_3 is much better than L_2, and L_4 is much better than L_5. This matches our intuition, e.g., L_5 does not have any executions of b although, according to the model in Figure 2, b should be executed for 98% of cases. Note that there is just one conformance measure and not two separate measures for recall and precision. This makes sense considering that increasing the probability of one trace should coincide with lowering the probabilities of other traces.

The example presented in this section shows that probabilities matter and cannot be excluded.

3. Related Work

This section discusses related work. Section 3.1 discusses techniques for conformance checking in process mining. Section 3.2 presents two case studies in process mining, and broader in Business Process Management, that can directly benefit from stochastic conformance checking. Finally, Section 3.3 summarises modelling formalisms for describing stochastic languages.

3.1. Conformance Checking Techniques

Existing conformance checking techniques can be broadly classified based on the analysis they apply, namely *quantification* and *characterisation* techniques. Quantification techniques assign a number, often between zero and one, which measures deviations between a given event log and process model. Then, intuitively, a conformance measurement of zero signifies that the log does not fit the model, while a measurement of one suggests that the log fits the model perfectly. In general, a measurement has a higher value if the model can replay more (parts of) traces from the log. Instead of mapping discrepancies and commonalities between the log and model onto a numeric domain, the characterisation techniques pinpoint them by means of descriptive artefacts. Often, such artefacts capture a specific class of deviations and commonalities, for example the minimal necessary deviations between the log and model. Next, in

Sections 3.1.1 and 3.1.2, we review prominent works on, respectively, quantification and characterisation conformance checking techniques.

In [49], the authors discuss a generalisation of the conformance checking problem. Instead of the classical setting, in which either the compared process model or event log is fully trusted, the generalised approach strives for a controlled balance of the trust in the quality of the compared log and process model. Given three distance functions, one between pairs of logs, another between pairs of models, and a third one between pairs composed of one model and one log, the generalised conformance checking is realised as an optimisation problem that searches for a log and model in the trusted proximity to the original compared log and model that demonstrate "good" conformance.

3.1.1. Quantification Techniques

One recall and several precision techniques presented in [51] are established on statistics of replaying event log traces in the model. The recall measure penalises tokens that are required but are missing to support the execution of traces and tokens that remain after traces get executed. The precision measures penalise large numbers of enabled transitions and mismatches in the pairs of activities that characterise the behaviours captured in the log and model.

In [4], the authors present a technique for measuring recall based on the notion of an optimal alignment, which describes minimal deviations between traces of the log and model. The approach establishes recall by normalising the total cost of optimal alignments of all traces in the log against the worst-case scenario alignments that maximise deviations between trace activities by deliberately avoiding capturing commonalities of traces. In [39], a precision measure is proposed that constructs a prefix automaton of states of the model that are visited by the traces of the log. Then, for each automaton state, the approach evaluates if there are enabled activities by the model not present in the log. In [4, 9], this approach is extended by aligning the log with the model before constructing the prefix automaton, which enables support for log traces that do not fit the model perfectly. Finally, in [36], this approach is further extended to quantify the commonalities and discrepancies in the data-, resource- and time-related aspects of the compared traces.

In [24], the precision of discovered models is measured using negative events, based on the notion of a confusion matrix. In [59], the authors devise an approach that establishes precision as the ration of the behaviour allowed by the model that does not contradict the generated negative events. Finally, in [57], the measure from [24] was generalised by assigning weights to negative events that reflect the confidence of the event being negative. Consequently, the precision measure from [24] was generalised as well.

An approach for measuring precision based on anti-alignments is presented in [19]. Given a log trace, an anti-alignment is a trace described by the model that is as much different from the log trace as possible. In the approach, some traces are removed from the log, and then the anti-alignments are used to establish how precise the model describes the log traces.

In [33], the authors present recall and precision measures that abstract the behaviour captured in the model and log to subsets of activities of the size of some input parameter k, encode the abstractions as deterministic finite automata and, finally, perform structural comparisons on the automata. A technique that measures precision and recall based on common abstract representations of process models and logs, called k-order Markovian abstractions, is presented in [12]. The higher the value of k used to construct the abstractions, the more refined the computations, though at the cost of exponential run time. The idea of using abstract representations of models and logs for computing the precision and recall are also explored in [51, 60].

In [46], the authors present precision and recall measures grounded in the notion of topological entropy of an irreducible language. The measures allow for a monotone assessment of the compared collections of traces specified by the model and log. In [43, 27], the measures are extended to account for traces that differ in some activities.

Recently, the process mining community started discussing formal properties that good conformance measures should satisfy [55, 3, 46, 43]. By demonstrating that a measure fulfills certain properties, one, indirectly, establishes its usefulness, as the properties allow explaining and comparing the measured values. As of today, the entropy-based conformance measures [46] are the only known measures that satisfy all the properties put forward by the community [46, 54].

All the above-discussed techniques ignore the stochastic perspective of traces described in the compared model and log. Next, we discuss techniques that take the stochastic perspective into account.

In [53], the question of how to compare actual process execution with the scheduled execution in terms of performance was addressed, using queueing networks. Queueing networks describe inter-case timing dependencies, while traces in stochastic languages (used in this paper) are in princple independent. Furthermore, the queueing networks of [53] do not support choice, which is a key concept of stochastic behaviour. Furthermore, where [53] uses Markov chains to compare models of behaviour, we directly compare the behaviour of logs/models with one another without using any abstraction, and we allow approximate matching using Levenshtein. It would be interesting to combine the concepts of [53] and this paper.

In [29], a range of desired properties for stochastic precision and recall measures is presented, and techniques grounded in the entropy of the compared stochastic languages that indeed satisfy these properties are introduced. These techniques do not take the stochastic perspective of both log and model into account *at the same time*. That is, recall ignores the stochastic perspective of the model, and precision ignores the stochastic perspective of the log.

Finally, in [44], the authors propose an entropic relevance measure for stochastic conformance checking, which measures the average number of bits required to compress a log trace based on the given stochastic process model, and is computable in time linear in the size of the input event log.

3.1.2. Characterisation Techniques

A (control-flow) alignment between a trace from an event log and a trace described by a model is a sequence of moves, where a move is a pair in which the first element refers to an activity in the log trace and the second element refers to a matching activity in the model trace. As the matching may not be feasible due to the discrepancies between the traces, one element in a move can be a special skip symbol to denote that the only activity in the move pair, either from the log or model, could not be matched. Note that moves in an alignment must be composed in such a way that the projection to first (second) elements of the moves that omits the skip symbols is the log (model) trace. An optimal alignment between a log trace and a process model is an alignment between the log trace and some trace described by the model with the lowest possible cost per some non-negative costing model on moves. Consequently, an alignment describes the "cheapest" possible deviations between the log trace and traces supported by the model. The notion of control-flow alignment and the first technique for constructing optimal alignments were proposed in [8, 4].

The work in [23] presents a technique for conformance checking between an event log and process model based on their common representations as event structures, artefacts that encode the partial order semantics of processes. The technique proceeds by folding the event log and unfolding the process model into two structures and then constructing their partially synchronised product that explains the commonalities and discrepancies of the two compared sets of behaviour. As an event log does not contain information about concurrent execution of activities, the technique is parametrised by a function, a concurrency oracle, that mines the partial order dependencies between the activities.

In [18], the authors generalised the notion of an optimal control-flow alignment to account for data, time and resources perspectives encoded both in process models and event logs. The technique starts by constructing an optimal control-flow alignment and then extending it to further perspectives. The technique discovers a valid process model execution with a minimal - subject to the limitation imposed by the employed two-phase search approach - cost of deviations with the given observed trace. The approach to multi-perspective alignment presented in [35] overcomes the limitation of the approach in [18] by suggesting a technique for constructing globally optimal multi-perspective alignments across all the process perspectives. Finally, in [40], the authors presented an approach that, given two event logs, constructs an abstract visualisation, in the form of a graph or table, of their statistically significant differences.

As of today, to the best of our knowledge, there are no characterisation conformance techniques that take the stochastic perspective of the compared model and log into account. Hence, this paper aims to address this.

3.2. Case Studies

In [45], the authors report on the outcomes of a project with a major German health insurance company that aimed to analyse and simplify approximately 4,000 models captured using the Event-driven Process Chain (EPC) notation annotated with probabilities of taking decisions at the designed branching

points. The models and annotations were developed manually, by the business analysts of the company. The probabilities were developed to reflect the likelihood of various business scenarios when handling clients or performing internal operations. Consequently, the insurer relied on the models to estimate the number of employees, i.e., staff-hours, to hire to support all the business processes in the upcoming financial year. The models and the probability annotations were reviewed and updated every year, which constituted a significant effort on the company side. Using logs of executed business processes at the end of a financial year, stochastic conformance checking can be used to analyse the correctness of the original probability estimates and guide an automatic configuration of decision probabilities in models used to plan resources for the next year.

The work in [14] presents an analytical approach for comparing whether organisations execute, what essentially is, the same business process in the same way. The approach allows comparing observed process executions with the intended design deployed by the organisation, as well as with variants of the design deployed by other organisations. Such comparisons can help organisations understand what they do differently, and what potential for standardisation of business processes exists. The comparison is based on classical, non-stochastic, alignments. By extending the analytics offered in [14] to stochastic conformance checking, one can obtain a more accurate diagnostics which pinpoints frequent and rear discrepancies, rather than witnessing their presence or absence. We will elaborate on this in Section 6.5.

3.3. Formalisms for Describing Stochastic Languages

There exist several modelling languages for capturing stochastic languages. Here, we discuss several of such languages.

A *Markov chain* is a sequence of random variables [22]. The possible values of the variables form the state space of the chain. Common types of Markov chains differ in how they represent time, either discrete- or continuous-time, and in the state space they encode, either countable or continuous [11]. Usually, the probability of a random variable in the Markov chain does not depend on the history of the chain, i.e., the values of the preceding variables. In Markov chains with memory, however, the future values of variables depend on the past values.

A *probabilistic automaton* [47] generalises the concept of a deterministic finite automaton by specifying probabilities for state transitions and a distribution for choosing the initial state. Consequently, the probabilistic automaton is also a generalisation of the concept of a discrete-time countable Markov chain. Given a current state, the probabilities indicate the likelihoods of the next input, however given an input, the next state is deterministic.

In a classical Petri net, transitions fire non-deterministically. In a stochastic Petri net, transitions race to fire, using a delay drawn from a distribution. The reachability graph of a stochastic Petri net can be mapped to a Markov chain, where each state in the reachability graph is mapped to the corresponding state in the Markov chain, and each transition firing is mapped to a transition in a Markov chain of the corresponding probability. *Generalised stochastic Petri net*s extend this notion with a second category of transitions: immediate transitions,

which fire before the other transitions, with a likelihood based on their weight relative to the total weight of all enabled immediate transitions [37]. In the next section, we extend generalised stochastic Petri nets with transition labels and introduce them formally.

4. Preliminaries

In this section, we introduce existing concepts to be used in the remainder of the paper.

Stochastic Languages. Given an alphabet Σ of activities, a *trace* is a sequence of occurrences of activities. Let $\mathcal{T} = \Sigma^*$ denote the universal set of traces. A *stochastic language* L assigns a probability to each trace:

$$L{:}\mathcal{T} \to [0,1] \text{ such that } \sum_{t \in \mathcal{T}} L(t) = 1$$

A *finite stochastic language* assigns a positive probability to a finite number of traces. An *event log* is a multiset of traces. Inherently, an event log denotes a finite stochastic language. Finally, a stochastic language L can be projected to its corresponding set of traces $\widetilde{L}$:

$$\widetilde{L} = \{t \mid L(t) > 0\}$$

Stochastic Process Models. A *labelled Petri net* $(P, T, F, \Sigma, \lambda, M_0)$ consists of a set of places P, a set of transitions T such that $P \cap T = \varnothing$, a flow relation $F{:}(P \times T) \to (T \times P)$, a finite alphabet of activities Σ such that $\tau \notin \Sigma$, a transition labelling function $\lambda{:}T \to \Sigma \cup \{\tau\}$, and an initial marking M_0. We use the standard semantics of Petri nets [5]: a *marking* containing tokens on places of P indicates the state of the net, and a transition $t \in T$ is *enabled* if all places connected to it by F contain tokens. When t *fires*, it consumes and produces tokens on the places as indicated by the flow relation F, and if $\lambda(t) \neq \tau$, indicates the execution of activity $\lambda(t)$.

A *generalised stochastic labelled Petri net* (GSLPN) is an octuple $(P, T, F, \Sigma, \lambda, M_0, T_i, w)$ in which $(P, T, F, \Sigma, \lambda, M_0)$ is a labelled Petri net, $T_i \subseteq T$ is a set of immediate transitions, and $w{:}T \to \mathbb{R}$ is a weight function. Semantically, if an immediate transition $(t \in T_i)$ is enabled, then no *timed transition* $(t' \in T \backslash T_i)$ is enabled. Let T' be the set of enabled transitions. Then, the probability that a transition $t' \in T'$ fires, is $w(t') / \sum_{t'' \in T'} w(t'')$.

Please note that immediate transitions do not increase the expressivity of GSLPN, that is, every GSLPN can be expressed without immediate transitions [17], however these transitions have been used in literature to increase the legibility of stochastic models, and some stochastic process discovery techniques return both types of transitions [48]. Similarly, while silent transitions are typically not part of stochastic Petri net definitions, they are omnipresent in process mining and thus part of stochastic process discovery [48], and thus we defined GLSPN with a labelling function λ.

In literature, timed transitions are typically modelled as firing after a particular time, modelled by an exponential distribution [50]. However, by the memory-less property of the exponential distribution, for our purposes it suffices to model *which* transition fires with what probability, rather than *how long* it takes before transitions fire.

Stochastic Path Languages. A *path* is a sequence of transitions that brings a stochastic process model from its initial state M_0 to a final state[1]. Let $\mathcal{P} = T^*$ denote the universal set of all paths. Then, a *stochastic path language* M denotes a probability for each path:

$$M: \mathcal{P} \to [0,1] \text{ such that } \sum_{p \in \mathcal{P}} M(p) \leq 1$$

A *finite stochastic path language* assigns a positive probability to a finite number of paths. Notice the inequality; we explicitly allow *partial stochastic path language*s in which $\sum_{p \in \mathcal{P}} M(p) < 1$. As we will show in Section 5.7, we can omit the "tail" of exceptional paths, for instance going through a loop many times. Using the labelling function λ of the stochastic Petri net, a path can be projected onto a trace (where $\cdot$ denotes trace concatenation):

$$\lambda(\langle\rangle) = \langle\rangle$$
$$\lambda(\langle p_1 \rangle \cdot p) = \begin{cases} \lambda(p) & \text{if } \lambda(p1) = \tau \\ \langle \lambda(p_1) \rangle \cdot \lambda(p) & \text{otherwise} \end{cases}$$

Finally, a stochastic path language M can be projected to its corresponding set of paths $\widetilde{M}$ and its corresponding language:

$$\widetilde{M} = \{p \mid M(p) > 0\}$$
$$\lambda(M) = \{\lambda(p) \mid p \in \widetilde{M}\}$$

Note that the techniques presented in this paper are independent of the stochastic process formalism: the techniques require a stochastic (path) language, but no particular formalism.

5. Earth Movers' Stochastic Conformance Checking

In this section, we introduce our approach for stochastic conformance checking of event logs and stochastic process models: the Earth Movers' Stochastic Conformance (EMSC). Intuitively, EMSC mimics the earth movers' distance: consider both the log and the model as piles of sand, each having a particular shape. Then, the earth movers' distance is the minimal effort to transform

[1] In this paper, without loss of generality, we assume that every deadlock state is a final state and that every final state is a deadlock state. We also assume that from each reachable state, it is possible to reach a deadlock state (that is, no livelocks).

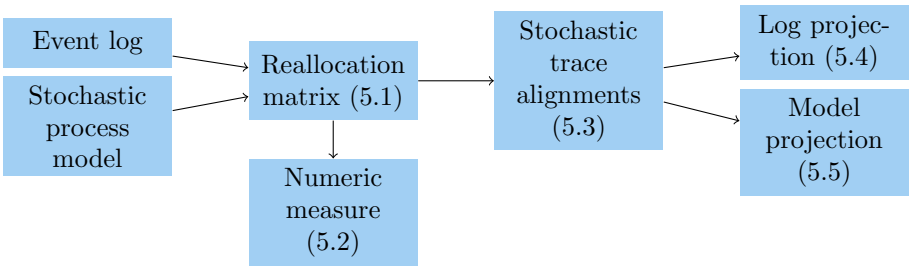
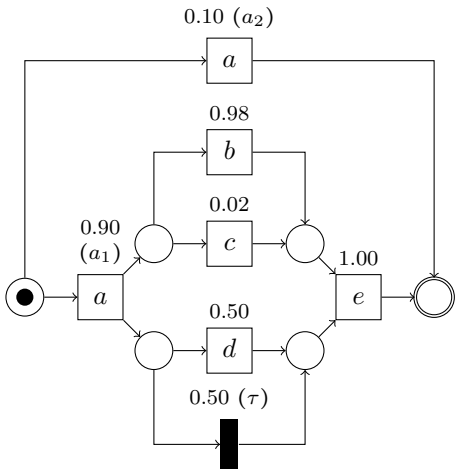


Figure 3: Overview of our approach. The numbers refer to the sections in which we discuss the concepts and steps.

$$[\langle a,b,e\rangle^{30},\ \langle a\rangle^{10},\ \langle a,d,e\rangle^{4},\ \langle a,b,d,e\rangle^{20},\ \langle a,d,b,e\rangle^{30},\ \langle a,c,d,e\rangle^{3},\ \langle a,c,e\rangle^{3}]$$

(a) Event log L_E.



(b) Stochastic model M_E.

Figure 4: Running example of an event log and a stochastic process model.

one pile into the other, that is, the amount of sand that needs to be moved multiplied by the distance over which it needs to be moved.

Figure 3 shows an overview of the approach: first, a *reallocation matrix* is constructed, which shows how probability mass is to be moved between the log and the model (Section 5.1). From a reallocation matrix, a difference measure is derived (Section 5.2). In order to offer more detailed insights into the differences between log and model, first a set of *stochastic trace alignments* is computed (Section 5.3), which are consequently projected on the event log (Section 5.4) and the model (Section 5.5).

In the remainder of this section, we introduce each of these steps in detail, using the running example shown in Figure 4. We finish the section with two extensions to log-log and model-model comparisons in Section 5.6 and discuss practical considerations and efficient implementation in Section 5.7.

Log \ Model	p	$\langle a_2 \rangle$	$\langle a_1, b, d, e \rangle$	$\langle a_1, b, \tau, e \rangle$	$\langle a_1, c, d, e \rangle$	$\langle a_1, c, \tau, e \rangle$	$\langle a_1, d, b, e \rangle$	$\langle a_1, d, c, e \rangle$	$\langle a_1, \tau, b, e \rangle$	$\langle a_1, \tau, c, e \rangle$
	0.1	0.2205	0.2205	0.0045	0.0045	0.2205	0.0045	0.2205	0.0045	
$\langle a, b, e \rangle$	0.3	0	0	0.0795	0	0	0	0	0.2205	0
$\langle a \rangle$	0.1	0.1000	0	0	0	0	0	0	0	0
$\langle a, d, e \rangle$	0.04	0	0	0.0355	0	0	0	0.0045	0	0
$\langle a, b, d, e \rangle$	0.2	0	0.1950	0.0050	0	0	0	0	0	0
$\langle a, d, b, e \rangle$	0.3	0	0	0.0795	0	0	0.2205	0	0	0
$\langle a, c, d, e \rangle$	0.03	0	0.0255	0	0.0045	0	0	0	0	0
$\langle a, c, e \rangle$	0.03	0	0	0.0210	0	0.0045	0	0	0	0.0045

Table 1: A reallocation matrix for example log L_E and model M_E of Figure 4. To distinguish transitions of M_E with the same label, these have been numbered, and the silent transition is denoted with τ.

5.1. Reallocation Matrix

A *reallocation matrix* indicates how the probability mass of the traces of a log can be transformed into the probability mass of the paths of a model and vice versa: the rows indicate how the mass of a trace is distributed over the paths, while the columns indicate how the probability mass of a path is distributed over the traces.

Definition 1 (Reallocation matrix)**.** *Let L be a finite stochastic language and let M be a finite stochastic path language. Then $R\colon L \times M \to [0,1]$ is a* reallocation matrix *if and only if:*

1. *Each row, representing a trace $t \in \widetilde{L}$, sums to the probability $L(t)$:*

$$\forall_{t \in \widetilde{L}}\, L(t) = \sum_{p \in \widetilde{M}} R(t,p)$$

2. *Each column, representing a path $p \in \widetilde{M}$, sums to at least the probability $M(p)$:*

$$\forall_{p \in \widetilde{M}}\, M(p) \leq \sum_{t \in \widetilde{L}} R(t,p)$$

Notice the asymmetry in requirements 1 and 2: in order to support models with infinitely many traces in a finite reallocation matrix, Requirement 2 is an inequality (we elaborate on this in Section 5.7).

Table 1 shows a reallocation matrix for our example log L_E and M_E of Figure 4. In the paths of M_E, for readability, transitions with the same label have been numbered, and the silent transition is denoted with τ. In this matrix, for instance, all the probability of log trace $\langle a \rangle$ is mapped to the model path $\langle a_2 \rangle$.

Intuitively, following the analogy of the earth movers' distance, the cost of a reallocation matrix is the total probability mass that is moved multiplied by the distance over which it is moved.

Log \ Model	$\langle a_2 \rangle$	$\langle a_1, b, d, e \rangle$	$\langle a_1, c, \tau, e \rangle$	$\langle a_1, d, b, e \rangle$	$\langle a_1, b, \tau, e \rangle$	$\langle a_1, d, c, e \rangle$	$\langle a_1, \tau, c, e \rangle$	$\langle a_1, \tau, b, e \rangle$	$\langle a_1, c, d, e \rangle$
$\langle a, b, e \rangle$	0.67	0.25	0.33	0.25	0	0.50	0.33	0	0.50
$\langle a \rangle$	0	0.75	0.67	0.75	0.67	0.75	0.67	0.67	0.75
$\langle a, d, e \rangle$	0.67	0.25	0.33	0.25	0.33	0.25	0.33	0.33	0.25
$\langle a, b, d, e \rangle$	0.75	0	0.50	0.50	0.25	0.50	0.50	0.25	0.25
$\langle a, d, b, e \rangle$	0.75	0.50	0.50	0	0.25	0.25	0.50	0.25	0.50
$\langle a, c, d, e \rangle$	0.75	0.25	0.25	0.50	0.50	0.50	0.25	0.50	0
$\langle a, c, e \rangle$	0.67	0.50	0	0.50	0.33	0.25	0	0.33	0.25

Table 2: The distance matrix for example log L_E and model M_E.

The distance measure is a parameter of the approach, for which in this paper we propose the normalised string edit (Levenshtein) distance. The Levenshtein distance describes the minimum number of insertions, deletions and substitutions to transform one string into another. For a trace t and a path p, we define their distance $\delta(t, p)$ to be the Levenshtein distance between t and the down-projected trace of p ($\lambda(p)$) divided by the maximum length of t and $\lambda(p)$. The distance matrix of our example log L_E and model M_E is shown in Table 2.

Definition 2 (Cost of reallocation matrix)**.** *Let L be a finite stochastic language, let M be a finite stochastic path language and let R be a reallocation matrix over L and M. Then, the cost of R is the inner product over reallocation and distance:*

$$cost(R, L, M) = \sum_{t \in \widetilde{L}, p \in \widetilde{M}} R(t, p)\delta(t, p)$$

For instance, the cost for our reallocation matrix in Table 1, example log L_E and model M_E is 0.0475.

Finally, given an event log and a model, we refer to a reallocation matrix for these with minimum cost as an *optimal reallocation matrix*. The reallocation matrix in Table 1 is an optimal reallocation matrix for our example log L_E and model M_E. Please note that the optimal reallocation matrix is not unique: the columns for $\langle a_1, b, \tau, e \rangle$ and $\langle a_1, \tau, b, e \rangle$ are interchangeable.

5.2. Numeric Measure

The cost of a reallocation matrix provides a measure for the differences between a given log and model. Analogous to other conformance checking techniques, we transform the cost into a conformance measure, such that 1 indicates perfect conformance and 0 indicates the largest difference:

Definition 3 (Earth Movers' Stochastic Conformance)**.** *Let L be a finite stochastic language, let M be a finite stochastic path language and let R be an optimal reallocation matrix for L and M. Then, the* Earth Movers' Stochastic Conformance *(*EMSC*) measure* $\text{EMSC}(L, M)$ *is* $1 - cost(R, L, M)$.

Considering Definition 2, as $0 \leq \delta \leq 1$ and R sums to 1 over all t and p, the cost of a reallocation matrix is a number between 0 and 1. To illustrate this, we prove that the extreme values of EMSC, being 0 and 1, coincide with complete disjointness and equivalence, respectively. First, we prove that EMSC is 1 if and only if the log and model express the same stochastic language.

Lemma 1. *Let L be a finite stochastic language and let M be a finite stochastic path language. Then,* $\mathrm{EMSC}(L, M) = 1 \Leftrightarrow L = \lambda(M)$.

Proof. $\Rightarrow$ Assume $\mathrm{EMSC}(L, M) = 1$. Then, an optimal reallocation matrix R for L and M has zero cost. Consider $t \in \widetilde{L}$ and $p \in \widetilde{M}$ such that $R(t, p) \neq 0$. By Definition 2, $\delta(t, p) = 0$, hence $t = \lambda(p)$. By Definition 1, $\sum_{p' \in \widetilde{M}} R(t, p') = L(t)$, thus $L = \lambda(M)$.

$\Leftarrow$ Assume $L = \lambda(M)$. Consider $p \in \widetilde{M}$ and $t \in \widetilde{L}$ such that $\lambda(p) = t$. As $M(p) \leq L(t)$ and $\delta(l, p) = 0$, if we choose $R(t, p) = M(p)$ then the column of p does not contribute to the cost. This holds for all paths p, therefore there is a reallocation matrix with zero cost, hence $\mathrm{EMSC}(L, M) = 1$.

□

Second, we prove that EMSC is 0 if and only if the alphabets of the model and the log are disjoint:

Lemma 2. *Let Σ_L and Σ_M be alphabets, L be a finite stochastic language such that $\Sigma_L = \{a \in t \mid t \in L\}$ and let M be a finite stochastic path language such that $\Sigma_M = \{a \in \lambda(p) \mid p \in \widetilde{M}\}$. Then,* $\mathrm{EMSC}(L, M) = 0 \Leftrightarrow \Sigma_L \cap \Sigma_M = \varnothing$.

Proof. $\Rightarrow$ Assume $\mathrm{EMSC}(L, M) = 0$. Then, an optimal reallocation matrix R for L and M has unit cost, thus for all $t \in L$ and $p \in M$, the normalised Levenshtein distance is 1. Towards contradiction, assume $a \in \Sigma_L \cap \Sigma_M$, then there are t, p such that $a \in t$, $a \in p$. Then, the normalised Levenshtein distance of t and p is < 1. Hence, $\Sigma_L \cap \Sigma_M = \varnothing$.

$\Leftarrow$ Assume $\Sigma_L \cap \Sigma_M = \varnothing$. Then, for any $t \in L$ and $p \in M$ the normalised Levenshtein distance is 1. Hence, $\mathrm{EMSC}(L, M) = 0$.

□

5.3. Stochastic Trace Alignments

A reallocation matrix R defines, for a finite stochastic language L and a finite stochastic path language M, how the probability mass of L is reallocated to the probability mass of M. That is, for each combination of $t \in \widetilde{L}$ and $p \in \widetilde{M}$, $R(t, p)$ indicates the probability mass that is moved from t in L to p in M. To compare t and p in detail, we consider a *stochastic trace alignment*, which shows how t can be transformed into p using only insertions and deletions.

Intuitively, analoguous to (non-stochastic) alignments [8], a stochastic trace alignment is a sequence of *moves*, each indicating either a step in both t and p (a *synchronous move*), or a step only in t (a *log move*), or a step only in p (a *model move*):

Definition 4 (Move)**.** *Let Σ be an alphabet of activities such that $\text{-}\!\!\rightarrow \notin \Sigma$ and $\tau \notin \Sigma$, and let T be a set of transitions, labelled by the function $\lambda{:}T \to \Sigma \cup \{\tau\}$. Then, a* move *is a pair $(m_a, m_b) \in \Sigma \cup \{\text{-}\!\!\rightarrow\} \times T \cup \{\text{-}\!\!\rightarrow\}$ such that $\neg(m_a = \text{-}\!\!\rightarrow \wedge m_b = \text{-}\!\!\rightarrow)$ and $(m_a \neq \text{-}\!\!\rightarrow \wedge m_b \neq \text{-}\!\!\rightarrow) \Rightarrow m_a = \lambda(m_b)$.*

A stochastic trace alignment is a sequence of moves, such that the synchronous and log moves form the trace t, while the synchronous and model moves form the path p. To formally define stochastic trace alignments, we use two helper functions to project a sequence of moves to either the log ($\uparrow$) or model ($\downarrow$) perspective:

$$\uparrow(\langle\rangle) = \langle\rangle$$

$$\uparrow(\langle (m_a, m_b)\rangle \cdot X) = \begin{cases} \uparrow(X) & \text{if } m_a = \text{-}\!\!\rightarrow \\ \langle m_a \rangle \cdot \uparrow(X) & \text{otherwise} \end{cases}$$

$$\downarrow(\langle\rangle) = \langle\rangle$$

$$\downarrow(\langle (m_a, m_b)\rangle \cdot X) = \begin{cases} \downarrow(X) & \text{if } m_b = \text{-}\!\!\rightarrow \\ \langle m_b \rangle \cdot \downarrow(X) & \text{otherwise} \end{cases}$$

Definition 5 (Stochastic trace alignment)**.** *Given a trace t and a path p, a* stochastic trace alignment *is a sequence $\langle m_1, \dots m_n\rangle$ of moves, such that $t = \uparrow(\langle m_1, \dots m_n\rangle)$ and $p = \downarrow(\langle m_1, \dots m_n\rangle)$.*

An *optimal* stochastic trace alignment has the maximum number of synchronous moves for a given trace t and path p. We refer to such an alignment with $A(t,p)$. Intuitively, given a corresponding reallocation matrix R, $R(t,p)$ then refers to the probability mass of the trace alignment.

For instance, considering trace $t = \langle a, c, d, e\rangle$ and path $p = \langle a_1, b, d, e\rangle$ from our example log L_E and model M_E, the following is an optimal stochastic trace alignment with three synchronous moves:

t	a	c	$\text{-}\!\!\rightarrow$	d	e
p	a_1	$\text{-}\!\!\rightarrow$	b	d	e

Please note that, like for non-stochastic alignments [8], the optimal stochastic alignment is not necessarily unique: for instance, in our example the log move on c and the model move on b could be swapped, yielding another optimal stochastic trace alignment. As another example, consider the trace $\langle a\rangle$ and the path $\langle a_1, a_2, a_3\rangle$. While it is clear that the event a needs to be mapped to either transition a_1, a_2 or a_3, there is no information available as to which one it should be, and consequently which transitions should be considered model moves. This is a common challenge in conformance checking techniques, thus no conformance checking technique can be made deterministic without making arbitrary choices.

In our implementation, we compute an optimal stochastic trace alignment $A(t,p)$ for each pair of $t \in \widetilde{L}$ and $p \in \widetilde{M}$ for which the reallocation matrix

$R(t,p) > 0$. We re-use the intermediate steps of the Levenshtein computation to construct a path through the state space of $t \times p$ deterministically.

5.4. Log Projection

A set of stochastic trace alignments provides detailed information about the relation between an event log and a model. In this section, we project this information on the traces of the event log: for each event, we visualise the probability that the event is synchronous with the model.[2]

We first introduce a helper function ϕ that takes a sequence of moves and an index k and returns whether the k^{th} log or synchronous move in the trace alignment is a synchronous move:

$$\phi(\langle (m_a, m_b) \rangle \cdot X, k) = \begin{cases} \phi(X,k) & \text{if } m_a = \text{-}\!\!\rightarrow \\ \phi(X,k-1) & \text{if } m_a \neq \text{-}\!\!\rightarrow \wedge k > 1 \\ m_b \neq \text{-}\!\!\rightarrow & \text{if } m_a \neq \text{-}\!\!\rightarrow \wedge k = 1 \end{cases}$$

Let L be a finite stochastic language, let M be a finite stochastic path language, and let $t = \langle e_1, \dots e_n \rangle$ be a trace such that $t \in \widetilde{L}$. Then, given that we observe trace t in the log, e_i is a synchronous move in the model with likelihood proportional to the probability mass of traces in which e_i is a synchronous move.

Definition 6 (Event synchronous-likelihood)**.** *Let L be a finite stochastic language and let M be a finite stochastic path language. Furthermore, let $t = \langle e_1 \dots e_n \rangle$ be a trace in $\widetilde{L}$. Then, for each event e_i at index i in t:*

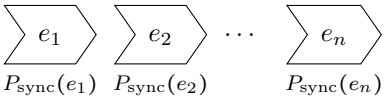
$$P_{sync}(e_i) = \frac{\sum_{p \in \widetilde{M}} R(t,p) \times \begin{cases} 1 & \textit{if } \phi(A(t,p), i) \\ 0 & \textit{otherwise} \end{cases}}{\sum_{p \in \widetilde{M}} R(t,p)}$$

Intuitively, given that a trace t happens in the log, the likelihood that event $e_i \in t$ is executed in the model (and e_i thus is a synchronous move) is $P_{\text{sync}}(e_i)$. Symmetrically, the likelihood that e_i is not executed in the model (and e_i thus is a log move) is $1 - P_{\text{sync}}(e_i)$.

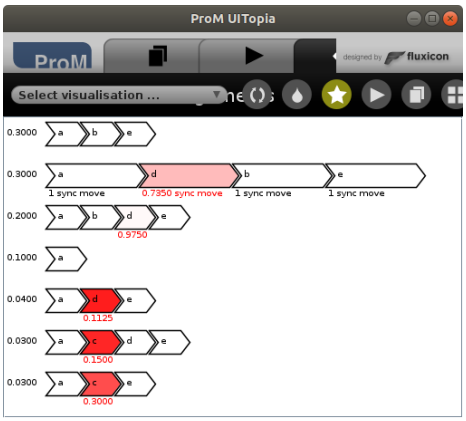
For instance, for our event log L_E and model M_E, consider the trace $t = \langle a, c, d, e \rangle$. According to an optimal reallocation matrix (Table 1), the probability of t is reallocated to two paths: $p_1 = \langle a_1, b, d, e \rangle$ and $p_2 = \langle a_1, c, d, e \rangle$, with probability $R(t, p_1) = 0.0255$ and $R(t, p_2) = 0.0045$. Consider the stochastic trace alignments $\begin{array}{c|cccccc} t & a & c & \text{-}\!\!\rightarrow & d & e \\ \hline p_1 & a_1 & \text{-}\!\!\rightarrow & b & d & e \end{array}$ and $\begin{array}{c|cccc} t & a & c & d & e \\ \hline p_2 & a_1 & c & d & e \end{array}$. Then, $P_{sync}(c \text{ at position } 2 \text{ in } p_2) = \frac{0.0045}{0.0255+0.0045} = 0.15$. Thus, event c in t is a synchronous move with 15% probability if trace t occurs.

To visualise this log projection, we use the concepts as shown in Figure 5: the traces of events are shown as rows of chevrons. Each chevron is annotated with its P_{sync} probability and coloured with a shade of red accordingly, which highlightes the deviating events.

[2] Formally, these are fractions that can be interpreted as probabilities as they are [0,1].



(a) The visualisation concepts.



(b) The model projection of our example log L_E, model M_E and reallocation matrix.

Figure 5: Visualisation of log moves. Given a trace $t = \langle e_1, \ldots e_n \rangle$, the visualisation shows each event e_i as a chevron, annotated with the likelihood that this event is synchronous ($P_{\text{sync}}(e_i)$). To highlight deviations, a chevron is coded with a shade of red: the lower the synchronous-move value, the darker the red.

5.5. Model Projection

In order to visualise where a given model differs from an event log, we also project the stochastic alignments onto a process model.

We first introduce two helper functions to count the occurrences of a particular transition in a stochastic trace alignment. The first (φ) simply counts the occurrences of a particular transition x:

$$\varphi(m, x) = |[(m_a, m_b) \in m \mid m_b = x]|$$

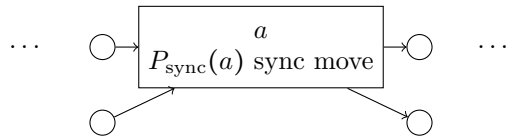
The second helper function φ' counts the number of synchronous moves on a particular transition x in a stochastic trace alignment:

$$\varphi'(m, x) = |[(m_a, m_b) \in m \mid m_b = x \wedge (\lambda(m_b) = \tau \vee m_a \not\in \rightarrow)]|$$

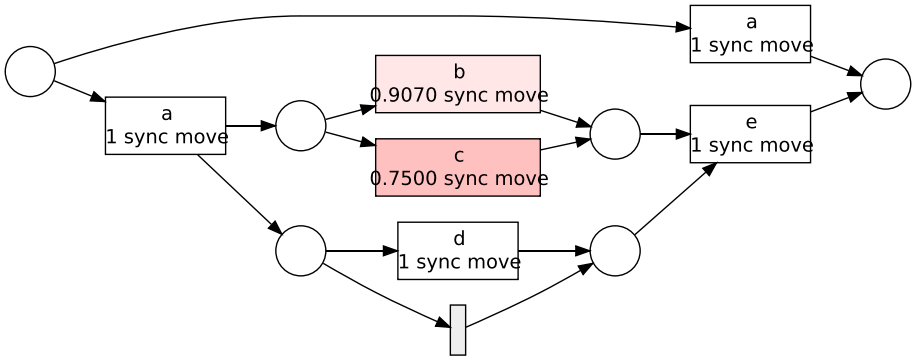
Definition 7. *Let L be a finite stochastic language, let M be a finite stochastic path language, and let x be a transition in the stochastic model from which M is derived. Then, given that the model indicates that x should be executed in the model, it is actually executed (and not a model move) with likelihood:*

$$P_{sync}(x) = \frac{\sum_{t \in \widetilde{L} \wedge p \in \widetilde{M} \wedge \varphi(A(t,p),x) \geq 1} R(t,p) \frac{\varphi'(A(t,p),x)}{\varphi(A(t,p),x)}}{\sum_{t \in \widetilde{L} \wedge p \in \widetilde{M} \wedge \varphi(A(t,p),x) \geq 1} R(t,p)}$$

Intuitively, for a transition x, $P_{\text{sync}}(x)$ indicates the probability that if x should fire according to the model, it actually fired in the event log. Symmetrically, $1 - P_{\text{sync}}(x)$ is the likelihood of a model move on x if x should happen according to the model.



(a) Concepts of the visualisation.



(b) The model projection of our example log L_E, model M_E and reallocation matrix.

Figure 6: Differences of stochastic language projected on a model.

For instance, consider the transition d in Figure 4. The paths in which d is executed are $\langle a_1, b, d, e\rangle$, $\langle a_1, d, b, e\rangle$, $\langle a_1, d, c, e\rangle$ and $\langle a_1, c, d, e\rangle$, and in total 5 trace alignments are relevant for these paths. In all of these trace alignments, d is synchronous only, so $P_{sync}(d) = 1$.

To visualise these model projections, we use the concepts shown in Figure 6a: the model is visualised as usual, however each transition is also annotated with its probability P_{sync}, and coloured accordingly with a shade of red. The visualisation of the model projection of our example log L_E, model M_E (Figure 4) and reallocation matrix (Table 1) is shown in Figure 6b.

5.6. Log-Log and Model-Model

Thus far, we used our Earth Movers' Stochastic Conformance checking technique only for log-model comparisons. However, using a few simple modifications, the concepts apply to log-log and model-model comparisons as well. That is, the reallocation matrix (Definition 1) is updated accordingly:

Definition 8 (Reallocation matrix - log-log)**.** *Let L_1 and L_2 be finite stochastic languages. Then $R: L_1 \times L_2 \to [0, 1]$ is a* reallocation matrix *if and only if:*

1. *Each row, representing a trace $t \in \widetilde{L_1}$, sums to the probability $L(t)$:*

$$\forall_{t \in \widetilde{L_1}} L_1(t) = \sum_{t' \in \widetilde{L_2}} R(t, t')$$

2. *Each row, representing a trace $t' \in \widetilde{L_2}$, sums to the probability $L(t')$:*

$$\forall_{t' \in \widetilde{L_2}} L_2(t') = \sum_{t \in \widetilde{L_1}} R(t, t')$$

For model-model comparisons, both finite stochastic path languages might not have their probabilities of traces sum to 1. Therefore, a third requirement is necessary to guarantee that the reallocation matrix itself sums to 1.

Definition 9 (Reallocation matrix - model-model)**.** *Let M_1, M_2 be finite stochastic path languages. Then $R: M_1 \times M_2 \to [0,1]$ is a* reallocation matrix *if and only if:*

1. *Each row, representing a path $p \in \widetilde{M_1}$, sums to at least the probability $M_1(p)$:*
$$\forall_{p \in \widetilde{M_1}} M_1(p) \leq \sum_{p' \in \widetilde{M_2}} R(p, p')$$
2. *Each column, representing a path $p' \in \widetilde{M_2}$, sums to at least the probability $M_2(p')$:*
$$\forall_{p' \in \widetilde{M_2}} M_2(p') \leq \sum_{p \in \widetilde{M_1}} R(p, p')$$
3. *The sum of the matrix is 1:*
$$\sum_{p \in \widetilde{M_1}, p' \in \widetilde{M_2}} R(p, p') = 1$$

In a similar manner, the log and model projections can be applied to log-log and model-model comparisons accordingly. We will show an example of this in Section 6.5.

5.7. Computing EMSC *in Practice*

Handling Infinite Behaviour. The distance and reallocation matrices are defined for a finite number of model paths. However, process models that contain loops might have an infinite number of paths, thus enumerating all paths is impossible. To support such models, paths are extracted until a user-chosen total probability mass has been covered, or a user-chosen time-out has been reached (whichever comes first).

Then, a partial stochastic path language might remain, that is, a language with the probability mass sum of less than 1. Definition 1 is lenient towards such partial languages: intuitively, the probability mass that is 'missing' from the partial language is distributed over the paths in the partial language by the reallocation matrix.

Finding an Optimal Reallocation Matrix. A straightforward approach to compute the optimal reallocation matrix is by means of linear programming. To this end, we introduce a variable for each entry of the reallocation matrix and formulate a Linear Program (LP) with the requirements of Definition 1 as its constraints and with Definition 2 as its objective function. Given a finite

stochastic language L and a finite stochastic path language M, we obtain the following LP:

$$\textit{minimise} \quad \sum_{t \in \widetilde{L}} \sum_{p \in \widetilde{M}} x_{tp} \cdot \delta(t,p) \tag{1}$$

$$\textit{s.t.} \qquad \forall t \in \widetilde{L} : \sum_{p \in \widetilde{M}} r_{tp} = L(t) \qquad \text{(Supply)} \tag{2}$$

$$\forall p \in \widetilde{M} : \sum_{t \in \widetilde{L}} r_{tp} \overset{(=)}{\leq} M(p) \quad \text{(Demand)} \tag{3}$$

$$\forall t \in \widetilde{L}\, \forall p \in \widetilde{M} : \qquad r_{tp} \geq 0 \qquad \text{(Non-negativity)} \tag{4}$$

where Equations 2 and 3 directly correspond to the Requirements 1 and 2 of Definition 1. Note that for path languages that do not require unfolding (that is, for models without loops) or for log-log comparison, Equation 3 becomes an equality. In this case, the linear program is equivalent to the general linear programming formulation of EMD.

However, in previous tests, solving the LP with general-purpose linear programming was the most time-consuming step of the approach [34], since it requires to operate on an explicitly formulated constraint matrix of size $(|\widetilde{L}|+|\widetilde{M}|)\cdot(|\widetilde{L}||\widetilde{M}|)$. In order to more efficiently solve this problem, we, resort to work from the field of operations research, or, more specifically, on methods for solving the transportation problem which already has been studied since the late 1930s [56]. In the Hitchcock transportation problem [25], we consider m supply nodes with certain quantities of a commodity serving a set of n demand nodes with predefined demands. While the supply nodes correspond to the traces in the log (the rows in Definition 1), the demand nodes correspond to the paths in the model (the columns). The objective is to find a minimum cost transshipment from the supply to the demand nodes where the cost of a shipment between two nodes is linear in the shipped quantity. The equality-constrained version of the above linear program (thus, not-truncated languages from models without loops) is an instance of the Hitchcock transportation problem.

Considering the specific structure of the transportation problem LP, dedicated methods, which do not require an explicitly formulated constraint matrix, have been developed. In our implementation, we use the method proposed in [41] that combines an Exterior Point Simplex Algorithm (EPSA) for the transportation problem [42] with the initialisation method in [6]. As is characteristic for simplex-type algorithms, the algorithm operates on a series of basic solutions until an optimal solution is found. In contrast to the normal Simplex algorithm, the intermediate basic solutions do not need to be feasible with respect to the constraints. Regarding the underlying network of the transportation problem, each of the encountered (linearly independent) basic solutions naturally corresponds to a forest in which each tree describes a closed subflow of commodities. We store and update these trees using the performance-oriented array-based implementation proposed in [1]. Besides, this characterisation of the optimal solution in terms of a basic solution also shows that the number of nonzero

entries in the minimal-cost reallocation matrix is bounded by $|\widetilde{L}| + |\widetilde{M}| - 1$. For integer demands, the algorithm has a complexity of $\mathcal{O}(\min(m,n) \times (m+n) * D)$, where D denotes the total demand [41]. For continuous demands (as in EMSC's case), proving a theoretical bound remains future work.

We initialise the algorithm according to the method in [6], which has been shown to perform best among the initialisation methods considered in [41]. Basically, we search the least-cost demand node for each supply node, and fill the corresponding edge by the supply node's capacity. Note that this might oversupply the demand node; thus, it might not be a solution according to Definition 1. Nevertheless, assuming that the log and the model describe similar behaviour, this greedy initial "solution" might already be close to the final optimal solution and, thus, might reduce the number of iterations required by the algorithm. As future work, it would be interesting to investigate the influence of initialisation strategies on run time.

The combined method has been shown to be approximately 3 times faster than the constraint-matrix-free adaption of the general Simplex algorithm to the transportation problem [41]. Even though the basic method assumes equality in Equation 3, thus excluding unfolded path languages, it can still be applied to the relaxed problem with inequalities in Equation 3. In Appendix A, we show that it suffices to stop and retrieve the optimal solution as soon as all column inequalities are satisfied.

Tool Support. The techniques described in this paper have been implemented as plug-ins of the ProM framework [21]. There are plug-ins available for log-log, log-model and model-model comparisons. As input, the plug-ins support stochastic Petri nets [48], and/or standard event logs. The implementation is open source and publicly available from `http://svn.win.tue.nl/repos/prom/Packages/EarthMoversStochasticConformanceChecking`.

6. Evaluation

In this section, we evaluate the Earth Movers' Stochastic Conformance (EMSC) checking technique as presented in this paper using four experiments: we first illustrate the necessity of conformance checking techniques to consider stochastic information. Second, we illustrate the influence of unfolding infinite behaviour on the proposed measure. Third, we show the feasibility of the approach on real-life event logs and models. Fourth, we illustrate the applicability of the log projections on a real-life log. Finally, we describe a use case that goes beyond the typical log-model conformance checking setting. We conclude the section by reviewing the reproducibility of the experiments.

6.1. Stochastic Information

As a first step, we illustrate the limitations of conformance checking techniques that ignore the stochastic perspective. We included four techniques: the established technique to compute fitness (alignments [4]), and three precision

log	EMSC (this paper)		Alignments [4]		ETC [9]		MFP [12]		MCC [43]	
	similarity	rank	fitness	rank	precision	rank	precision	rank	precision	rank
L_1	1.0000	1	1.0000	1	1.0000	1	1.0000	1	1.0000	1
L_2	0.8725	4	0.9286	5	1.0000	1	1.0000	1	1.0000	1
L_3	0.9995	2	0.9997	4	1.0000	1	1.0000	1	1.0000	1
L_4	0.9950	3	1.0000	1	0.9091	4	0.5833	4	0.982	4
L_5	0.7550	5	1.0000	1	0.9091	4	0.5833	4	0.982	4

Table 3: Existing conformance checking techniques applied to our example event logs and the model in Figure 2.

techniques: ETC precision [9], Markovian precision (PFP) [12], and Monotonic precision (MCC) [43]. We apply these techniques to one process model (Figure 2) and several event logs introduced in Section 2:

$$L_1 = [\langle a,b,d,e\rangle^{490}, \langle a,d,b,e\rangle^{490}, \langle a,c,d,e\rangle^{10}, \langle a,d,c,e\rangle^{10}]$$
$$L_2 = [\langle a,b,d,e\rangle^{245}, \langle a,d,b,e\rangle^{245}, \langle a,c,d,e\rangle^{5}, \langle a,d,c,e\rangle^{5}, \langle a,b,e\rangle^{500}]$$
$$L_3 = [\langle a,b,d,e\rangle^{489}, \langle a,d,b,e\rangle^{489}, \langle a,c,d,e\rangle^{10}, \langle a,d,c,e\rangle^{10}, \langle a,b,e\rangle^{2}]$$
$$L_4 = [\langle a,b,d,e\rangle^{500}, \langle a,d,b,e\rangle^{500}]$$
$$L_5 = [\langle a,c,d,e\rangle^{500}, \langle a,d,c,e\rangle^{500}]$$

The results are shown in Table 3. Conformance checking techniques typically address two dimensions: fitness (what part of the event log is represented by the model) and precision (what part of the model is also present in the event log). Typical fitness measures take the frequencies of traces in the log into account, as each trace contributes to the final measure, but the likelihoods in the model are not considered at all.

As argued in Section 1, intuitively, L_1 is most similar to the stochastic behaviour of the model (M), as its stochastic behaviour is equivalent. In our experiment, this was unearthed by all measures. Second, half of L_2's traces do not fit the model. In L_3, only 2 traces do not fit the model, but otherwise this log is similar to L_1. Thus, L_3 is closer to M than L_2. In the experiment, the precision measures do not spot any difference (as un-fitting traces are not a precision issue). The fitness measure and EMSC both rank L_3 higher than L_2. Finally, log L_4 consists of the most-occurring traces of M, while L_5 only contains the least-occurring traces. Thus, L_4 is more similar to M than L_5 is. In the experiment, only the EMSC is able to make this difference; all other measures consider L_4 and L_5 equivalent.

We conclude that this small experiment illustrates the limitations of existing techniques, despite some of them being stochastic techniques, as they do not distinguish between the different stochastic behaviour used in this example, and can thus not be used to provide insights into the differences between these types of behaviour.

6.2. The Influence of Unfolding

To show the influence of unfolding on a simple loop, we apply our technique to an example consisting of the event log $L = [\langle a\rangle^1, \langle a,a\rangle^3]$ and a stochastic

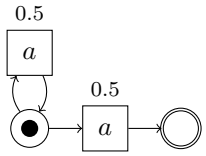


Figure 7: A process model with infinite behaviour.

process model M, shown in Figure 7, with stochastic language $[\langle a\rangle^{\frac{1}{2}}, \langle a,a\rangle^{\frac{1}{4}}, \langle a,a,a\rangle^{\frac{1}{8}}, \langle a,a,a,a\rangle^{\frac{1}{16}} \ldots]$.

Analytical solution. The EMSC of L and M can be computed analytically, using the following distance matrix:

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

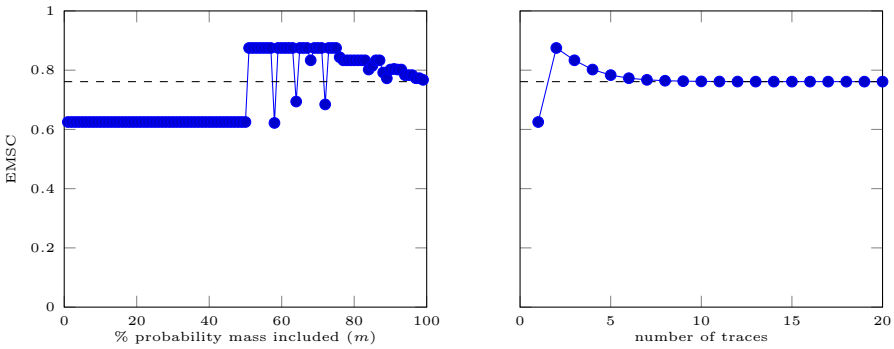
Then, an optimal reallocation matrix R is:

R	$\langle a\rangle$	$\langle a,a\rangle$	$\langle a,a,a\rangle$	$\langle a,a,a,a\rangle$	$\langle a,a,a,a,a\rangle$	...
$\langle a\rangle$	$\frac{1}{4}$	0	0	0	0	...
$\langle a,a\rangle$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	...

Finally, EMSC can be computed as follows:

$$\begin{aligned} cost(R,L,M) &= \frac{1}{4}\cdot 0 + 0\cdot\frac{1}{2} + 0\cdot\frac{2}{3} + 0\cdot\frac{3}{4} + 0\cdot\frac{4}{5} + \ldots + \\ &\quad \frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot 0 + \frac{1}{8}\cdot\frac{1}{3} + \frac{1}{16}\cdot\frac{2}{4} + \frac{1}{32}\cdot\frac{3}{5} + \ldots \\ &= \frac{1}{8} + \sum_{n=3}^{\infty}\frac{n-2}{2^n\cdot n} = \frac{13}{8} - \log 4 \\ &\approx 0.238706 \\ \mathrm{EMSC}(L,M) &= 1 - cost(R,L,M) \\ &\approx 0.761294 \end{aligned}$$

The experiment. In this experiment, we stepwise increase the amount of unfolded behaviour (m) from 0.01 to 0.99. Figure 8a shows the results, as well as the analytical solution for L and M (indicated by the dashed line). Up to $m = 50\%$, the unfolding only adds one trace to the model (consisting of a single a, which has a likelihood of 50% in the model), which yields a constant EMSC value of 0.625. This stable range of m indicates that the unfolding includes more probability mass than m, that is, if we choose $m = 2\%$ then the unfolding nevertheless includes 50% of the probability mass. At $m = 51\%$, a second trace is added, and



(a) Varying unfolding (parameter m).

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

Figure 8: EMSC measured over our example L and M. The dashed lines indicate the analytical solution.

the EMSC value jumps to 0.875, which remains constant until $m = 75\%$, with a few exceptions in which the EMSC value shortly drops. A manual inspection revealed that as the unfolding is multithreaded, there are race conditions on which paths are included, thus yielding these outliers in the EMSC value. We argue that this is an artifact of the limited language size and low m value in this experiment. Nevertheless, unfolding is inherently non-deterministic, thus EMSC is also non-deterministic. From $m = 76\%$, more traces are included in the unfolded language, such that after this point, the unfolding includes more and more traces that are not in the event log, multithreading issues become less relevant, and consequently EMSC drops and seems to approach the analytical solution.

Second, to illustrate the convergence to the analytical solution, we repeat the experiment where we manually create stochastic languages for L and M, where we unfold the loop of M an increasing number of times. Figure 8b shows these results. From this graph, it is clear that EMSC quickly converges to the analytical solution with every trace and loop unfolding added. At 6 traces, which corresponds to $m = 98\%$, the difference is a negligible 0.01.

Thus, we conclude that the unfolding parameter m influences the resulting EMSC value, that in some cases an analytical value can be computed, and that the unfolding eventually converges to this analytical value. In the next section, we show the influence of m on real life event logs. In future work, it would be interesting to investigate which classes of behaviour can be solved analytically.

6.3. Feasibility on Real-Life Event Logs

In this third experiment, we evaluate the applicability of EMSC to 11 publicly available real-life logs and stochastic process models discovered from them. First, we apply the Stochastic Miner (SM) [50] to these logs to obtain stochastic Petri nets. As these nets contain silent transitions, they can be seen as GSLPNs

Table 4: Real-life event logs used in the evaluation.

log	activities	traces	events	discovery [50]
BPIC12	36	13087	262200	✗ out of memory → baseline
BPIC18 Control summary	7	43808	161296	✓
BPIC18 Department control	6	29297	46669	✓
BPIC18 Entitlement application	20	15620	293245	✗ out of memory → baseline
BPIC18 Geo parcel documents	16	29059	569209	✗ out of memory → baseline
BPIC18 Inspection	15	5485	197717	✗ out of memory → baseline
BPIC18 Parcel document	10	14750	132963	✓
BPIC18 Payment application	24	43809	984613	✗ out of memory → baseline
BPIC18 Reference alignment	6	43802	128554	✓
Road Traffic Fines	11	150370	561470	✓
Sepsis	16	1050	15214	✓

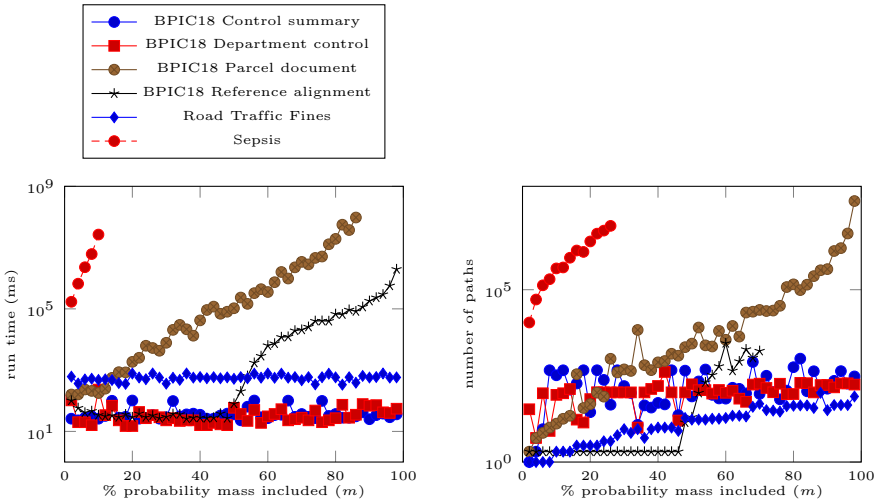
(see Section 4). Table 4 summarises the logs and their complexity. The discovery technique succeeded in discovering a process model for only 6 logs, running out of the 55GB of RAM we had available, which indicates the need for more research into stochastic process discovery techniques and their implementations. For each event log for which discovery was not successful, we used a baseline stochastic model. These baseline models were obtained by first applying Inductive Miner - infrequent [32] to obtain a Petri net, after which the transitions in this net were given weights according to the relative occurrence of their labels (activities) in the event log.

Second, we apply our new measures to the logs and the discovered GLSPNs. In order to apply EMSC, the behaviour is unfolded as described in Section 5.7, using various parameters m to study how the inclusion of mass influences the returned values. We vary m from 2% to 98%. In the remainder of this section, we discuss the results in two steps: we first discuss the run time and the number of paths considered, followed by a discussion on the returned EMSC values.

Number of paths & run times. First, we discuss the number of traces that resulted after unfolding. The results are shown in Figure 9. Please note that due to the inherent nondeterministic nature of EMSC's unfolding and the multithreadedness of the implementation, these results are indicative only, especially for lower m; and that these graphs have logarithmic y-axes.

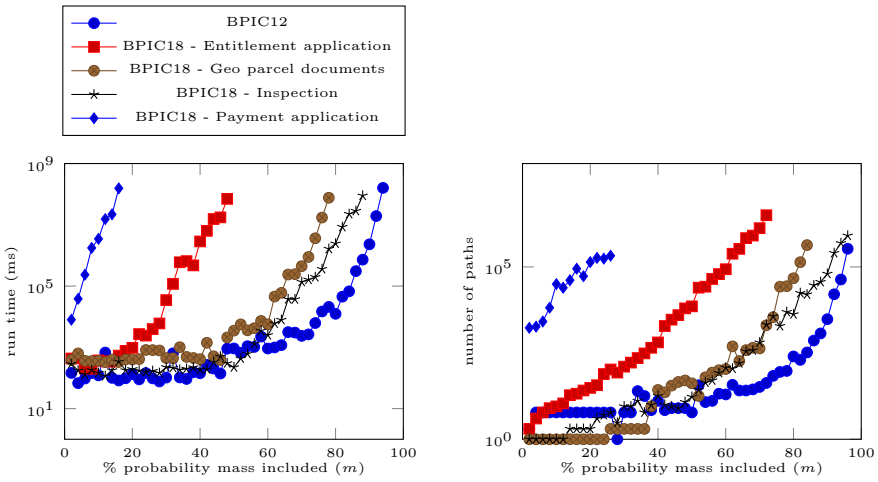
Some of the values could not be obtained: for `BPIC18 Parcel Document` with $m = 88$, the unfolding step finished quickly, however the computation of the reallocation matrix did not finish within 4 days. The discovered model for the Sepsis log contains transitions with a weight of 0, which are consequently never executed by EMSC and not part of the path language of the model.

For 3 out of 6 logs for which discovery was successful, 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 Reference alignment`, computation could take up to 20 minutes. 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.



(a) Run time.

(b) Unfolded language size.



(c) Run time.

(d) Unfolded language size.

Figure 9: Run time and size of unfolded languages with varying mass unfolding parameters (m) on real-life logs for logs where discovery was successful (a, b) and logs for which we used a baseline model (c, d).

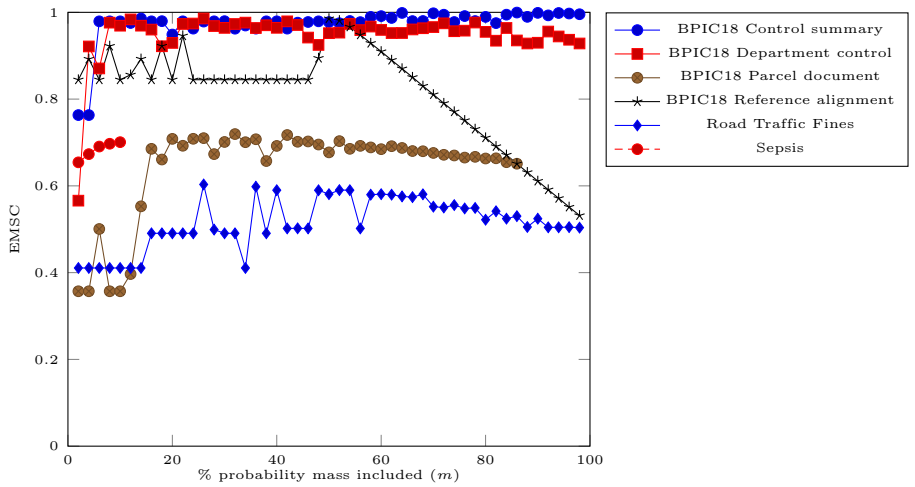
For any stochastic model with loops, with m approaching 100%, a "full" unfolding would need to consider an unbounded number of traces, and thus would suffer from an infinite run time. Inherently, every implementation is going to have its boundaries. The implementation of EMSC running on our machine seems to be capbable of handling around 10^6 traces with corresponding run times of around 30 minutes.

Values. The EMSC values are shown in Figure 10, split over the logs for which the discovery technique [50] was successful (Figure 10a) and the logs for which we used a baseline discovery technique (Figure 10b).

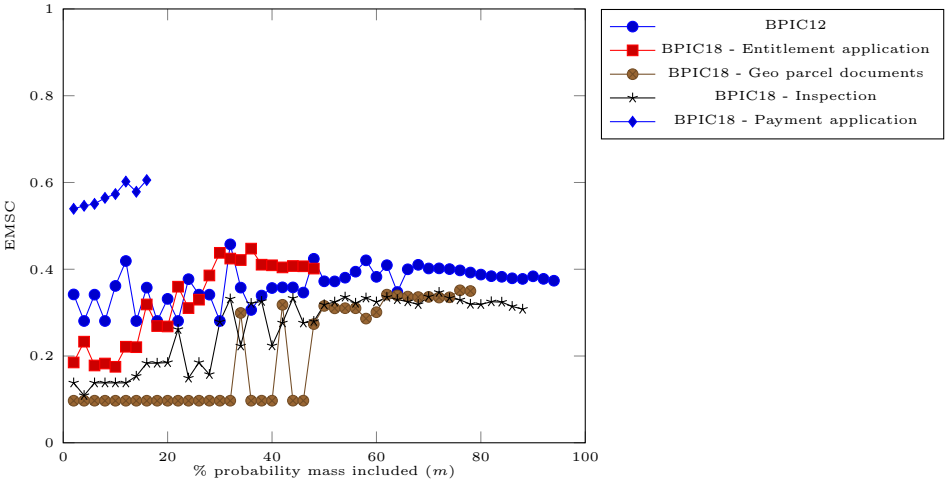
Most logs show expected behaviour with increasing m: due to nondeterminism of the unfolding step, for lower m they vary considerably, but stabilise with m approaching 100%. An exception is `BPIC18 Reference alignment`, which increases to 0.98 for $m = 52$, after which it decreases almost linearly. A manual inspection revealed that the GLSPN 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 EMSC drops.

For the other logs and models, EMSC yields a consistent ranking from $m = 60\%$, thus using only around 60% of the behaviour of the models. Finally, we can conclude that the discovery technique Stochastic Miner [50] discovers models of which the stochastic behaviour is much closer to the respective event logs, compared to the baseline technique.

We conclude that our technique is feasible on most tested real-life event logs, and that a high setting for m was not always necessary to rank models.

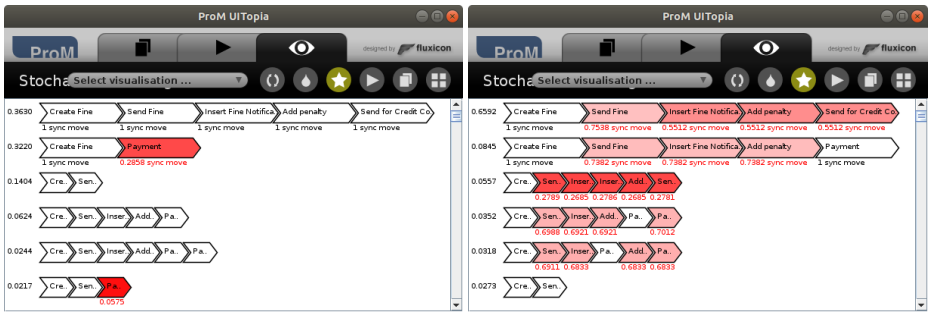


(a) Logs on which discovery was successful.



(b) Logs for which we used a baseline discovery technique.

Figure 10: EMSC values for our real-life logs.



(a) Fine amount ≤ 200 ($R_{\leq}$). (b) Fine amount > 200 ($R_{>}$).

Figure 11: EMSC applied to two sub-logs of the Road Traffic Fine Management Process.

6.4. Sub-Log Comparison

To illustrate the applicability of EMSC, we applied it to an event log of a road fine management process, in which fines are processed, payments received, and, if necessary, penalties applied and referred for credit collection. We split this log into two sub-logs: one with fines of EUR 200 and lower (143955 traces, $R_{\leq}$), and one for the other fines (6415 traces, $R_{>}$). The control flow followed in both logs is very similar. However, the stochastic perspective differs considerably, which is reflected by the EMSC of 0.67. The computation took several seconds.

To study these differences in more detail, the log projections on both sub-logs are shown in Figure 11. In the $R_{\leq}$ log, over 36% of the traces follow a path where a penalty is added before the fine is sent for credit collection (Figure 11a). This trace contains no stochastic differences with the $R_{>}$ log, that is, all events are always synchronous moves, which indicates that at least 36% of the traces of $R_{>}$ also followed this path. The second trace of Figure 11a, in which a fine is paid, appeared over 32% in $R_{\leq}$. However, the second event (`Payment`) is a synchronous move, that is, present in the traces of $R_{>}$ that were mapped to this trace, in only 29% of the times this trace appeared in $R_{\leq}$. Thus, the log projections can be used to get very detailed insights into differences in the stochastic perspective of sub-logs, which could be used to, for instance, split the process into two more specialised variants, based on the fine amount.

6.5. A Use Case: Municipalities

As an illustration of the flexibility of our technique and to show another potential use case, we consider the BPI Challenge logs of 2015. Each of these 5 logs describes a building permit approval processes in a different Dutch municipality. All processes serve the same function and have the same juridical foundation based in national laws, however each municipality executes the process in its own unique way, yielding similar but different processes [14]. Key questions are where the processes differ, and where they coincide. These differences of behaviour include paths that are infrequent in one municipality and mainstream in another. The complexity of the logs is described in Table 5.

log	traces	events	activities
BPIC15-1	1199	52217	398
BPIC15-2	832	44354	410
BPIC15-3	1409	59681	383
BPIC15-4	1053	47292	356
BPIC15-5	1156	59083	389

Table 5: BPIC 2015 event logs complexity.

log	BPIC15-1'	BPIC15-2'	BPIC15-3'	BPIC15-4'	BPIC15-5'
BPIC15-1'	0.800	0.530	0.764	0.694	0.560
BPIC15-2'	0.482	0.567	0.431	0.458	0.560
BPIC15-3'	0.761	0.511	0.829	0.724	0.551
BPIC15-4'	0.736	0.482	0.752	0.854	0.535
BPIC15-5'	0.477	0.567	0.419	0.459	0.631

(a) Cross-Organizational Comparison Framework (obtained from [14]).

log	BPIC15-1	BPIC15-2	BPIC15-3	BPIC15-4	BPIC15-5
BPIC15-1	1.0000	-	-	-	-
BPIC15-2	0.4752	1.0000	-	-	-
BPIC15-3	0.5432	0.4697	1.0000	-	-
BPIC15-4	0.4781	0.4781	0.4868	1.0000	-
BPIC15-5	0.4607	0.5011	0.4840	0.5077	1.0000

(b) A pairwise comparison using EMSC (this paper).

Table 6: Two procedures to find similarities in the different BPIC 2015 logs.

Using the method described in [14] (Cross-Organizational Comparison Framework (COCF)), first a process model is discovered from each log using an automated process discovery technique (the Evoluationary Tree Miner [15]). Second, the fitness of each process model with respect to each event log is reported, that is, what part of the behaviour of the event log is represented by the model. The result of this procedure is shown in Table 6a. Please note that due to the size and complexity of the event logs, in [14], the event logs were filtered to contain only the 47 most frequent activities accross all municipalities and the analyses were performed on these simplified logs. From these results, in a workshop setting, clusters of similar processes were identified and studied in more detail.

Alternatively, using the technique described in this paper applied to two logs rather than to a log and a model, a similar comparison procedure can be performed, by comparing each pair of event logs directly. The result of this procedure is shown in Table 6b, and its run times are shown in Table 7.

Afterwards, trace alignments could be used to study differences in more detail, for instance using the log projection of the most-different sub-logs BPIC15-2 and BPIC15-3 shown in Figure 12. In this projection of BPIC15-3, the most-occurring trace appears in 1% of all cases. Four moves (the two first and the two last ones) coincide with corresponding traces in BPIC15-2, however the three events in the middle differ considerably, appearing in BPIC15-2 as well with a

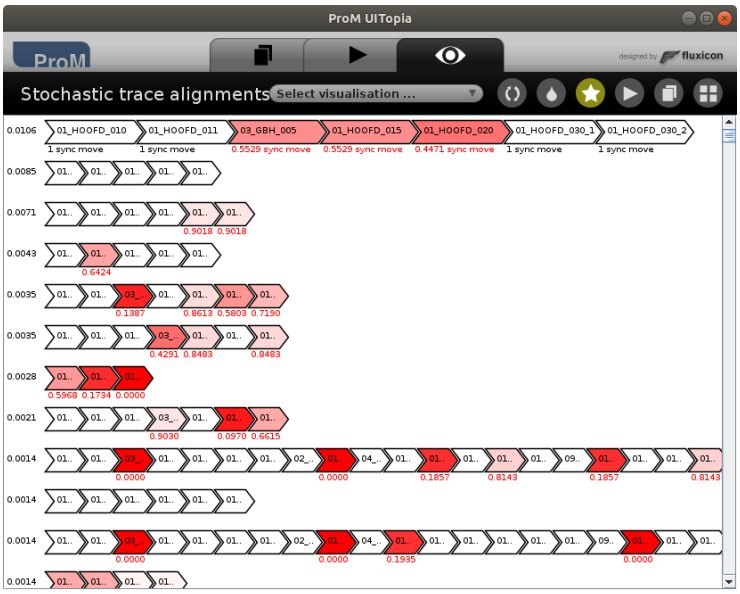


Figure 12: Log projection of stochastic trace alignments of logs BPIC15-2 and BPIC15-3, which were shown to be most different by EMSC.

likelihood of only 0.4471 (`01_hoofd_020`).

The log projection enables the study of details, however aggregating the trace alignments to a higher level remains future work: visualising a process consisting of 410 activities remains a challenge in itself.

Discussion. Both procedures identify groups of similar municipalities: for COCF, this was (BPIC15-1, BPIC15-3 and BPIC15-4), while for EMSC this were (BPIC15-1, BPIC15-3) and, weaker, (BPIC15-2, BPIC15-4, BPIC15-5). Using either result [14] could be applied to gain insights into more detailed common aspects and differences of the municipalities.

First, the diagonals of Table 6 show the comparison of each event log with itself, and clearly shows a weakness of the COCF approach: first, a process discovery technique is applied, which introduces some error, and this is reflected in the diagonals ranging from 0.567 to 0.854. Furthermore, due to the discovery step, the results are not symmetric, e.g. BPIC15-1 compared to BPIC15-2 yields a different value than BPIC15-2 compared to BPIC15-1. As the discovery step is avoided by using the measures introduced in this paper, each event log compared with itself has an EMSC of 1 (as shown in Lemma 1). EMSC is symmetric by definition, which eases the analysis.

Second, typical conformance checking techniques that are not stochastic aware still take the stochastic perspective of the event log into account, by accounting for traces that appear more often in event logs. However, as a process model is discovered, this stochastic information is lost in COCF. Thus, for each pair of event logs, COCF only takes the stochastic perspective of one of them

Log	BPIC15-1	BPIC15-2	BPIC15-3	BPIC15-4	BPIC15-5
BPIC15-1	1626	-	-	-	-
BPIC15-2	2373	1132	-	-	-
BPIC15-3	4153	2763	2117	-	-
BPIC15-4	2658	2055	3193	1474	-
BPIC15-5	3354	2503	3852	2764	1885

Table 7: Run time (ms) of EMSC on BPIC 2015 event logs.

into account, which is another source of asymmetry in the table. Consequently, EMSC not only considers which paths through the process were executed in the logs (the control flow), but also how often this happened.

Finally, we argue that the EMSC technique introduced in this paper, and by extension the pairwise-comparison method it enables, are feasible on complex real-life logs: the entire suite of comparisons combined took 175 seconds on our machine (i7-9700K CPU, deliberately limited to 1GB RAM), and log simplification was not necessary. The longest individual comparison, and application of EMSC, took around 4 seconds.

6.6. Reproducibility

All experiments (unless indicated otherwise) were performed on a single machine with i7-9700K CPU and 55GB RAM available for each experiment process, running a fully patched Ubuntu 18.04 in February 2020. All logs are public and available on `https://data.4tu.nl/repository/collection:event_logs_real`. The source code is available on `https://svn.win.tue.nl/repos/prom/Packages/EarthMoversStochasticConformanceChecking/`. We used SVN revision 43084.

7. Discussion

In this section, we illustrate the applicability of EMSC by reporting on a couple of case studies that were performed using EMSC and we describe several remaining open challenges.

In [13], concept drift detection is extended beyond control flow with the stochastic perspective (using EMSC), as to detect drift points in processes where exceptional behaviour becomes mainstream or mainstream behaviour becomes exceptional. The authors propose to complement the stochastic languages by additional perspectives, e.g. time, and exploit flexibly chosen reallocation distances to detect multi-perspective drifts. In addition to detecting general control flow drifts, this method allows to detect drifts in different perspectives, for example activity duration or sojourn time. Moreover, comprehensive drifts, which are difficult to detect considering a single perspective only, become detectable.

In [26], stochastic representations of behaviour have been used to study non-deterministic long-distance dependencies from event logs, where EMSC could be used to verify the quality of the detected dependencies. That is, a choice

early in the process influences a choice later in the process, but stochasticly and non-deterministically: if a particular option is chosen at the start of the process, then a particular option at the end of the process is more likely.

In [31], EMSC is used to identify the trace attributes (for instance, amount of a loan application, gender, mode of study, etc.) has the highest influence on the process that is being followed, and to quantify this influence. That is, this technique automatically recommends trace-level filters that maximise the differences between groups of students (measured using EMSC) having and not having a particular attribute and value. The applicability of this technique was illustrated in two case studies in Australian universities: one to study the processes of groups of students interacting with an online learning environment, where using the technique clear learning pathway differences between students were identified by instructors, and one that studied milestones of PhD students, which found differences in the processing of milestone e-forms (leave, confirmation, thesis submission, ...) between faculties that led to a university-wide change proposal. The second case study also found that there were virtually no differences in processing of e-forms related to demographic factors such as gender, mode of study and being domestic or international, which was a finding that was welcomed by stakeholders.

Next, we discuss remaining challenges.

Infinite behaviour. The use of reallocation matrices in the implementation inherently brings the requirement that the two considered stochastic (path) languages must be finite. As described in Section 5.7, process models with infinitely many traces, such as loops, can be unfolded to achieve this, at the cost of lowering the accuracy of the achieved results, as shown in Section 6. It would be interesting to research alternative ways to handle infite behaviour, or to prove bounds on the unfolding. Nevertheless, as shown in Section 6.2, it is sometimes possible to compute EMSC analytically for infinite stochastic path languages. It would be interesting to characterise the classes of models and logs for which this is possible.

Uniqueness. As shown in Section 5.1, an optimal reallocation matrix is not necessarily unique: several matrices with a minimal cost might exist. Similarly, for a given pair of traces and/or paths, multiple trace alignments might exist. In stochastic trace alignments, log and model moves might be swappable, however in our projections this difference is not visible (unlike some visualisations of alignments, such as in the Inductive visual Miner [30]). Finally, unfolding of infinite behaviour is conceptually not deterministic. Therefore, conceptually, the projected results of the EMSC (sections 5.4 and 5.5) are not deterministic either. This issue is common for conformance checking techniques: for instance, (non-stochastic) alignments [7] exhibit similar issues.

Visualisations. The projections introduced in sections 5.4 and 5.5 show only log moves for log projections, and only model moves for model projections. Even though trace alignments contain the information of the opposite (log moves for

model projection, and model moves for log projection) as well, we did not find a method to visualise this information in a succinct manner, and leave such visualisations for future work.

Summaries. When comparing an event log with a stochastic process model, the techniques described in this paper can succinctly visualise which parts of the model deviate from the log (Section 5.5). However, analysing the parts of the log that deviate from the model (Section 5.4 is tedious: all trace variants are visualised one-by-one, and for larger event logs, it is infeasible to derive information from this view. When we tested the EMSC measures described in this paper in a real-life industry setting with industry stakeholders, precisely this question arose, and we consider it an interesting area of future research, or a potential innovation factor for commercial process mining software vendors, to visualise the stochastic trace alignments in existing log visualisations, such as in the ubiquitous directly follows-based tools.

Distance measures. Finally, as mentioned in Section 5.1, the distance measure can be seen as a parameter of the EMSC approach. Therefore, the Levenshtein distance measure could easily be replaced with other measures, for instance to provide activity-label text matching, or to take the performance or time perspective into account. This would entail extending stochastic languages with information *how long* eac h event takes, and consequently in order to derive such a timed stochastic language from an event log would need (1) awareness of start and completion events, and either (2a) a control-flow process model or (2b) a partially ordered event log in order to take concurrency out of the equation in determining how long an activity execution took. Concepts from [53] could perhaps be used to add inter-case dependencies and sensitivity to scheduling.

An obvious catch in this regard is the curse of dimensionality: when adding dimensions to the distance measure, the expected distance between arbitrary events approaches 1, eventually making the measure useless.

8. Conclusion

The conformance checking technique presented in this paper considers the stochastic perspective as a *first-class citizen*. The main reason is to address the asymmetry between event logs and process models. A unique trace that cannot be replayed by the model is typically assumed to be less severe than a deviating trace that appears many times in the event log. Therefore, most conformance checking techniques take trace frequencies into account. Probabilities in process models can be seen as the counterpart of frequencies in event logs. However, most conformance checking approaches tend to abstract from probabilities. This explains why existing precision notions are problematic. They aim to penalise modelled behaviour that is not observed. However, process models with loops allow for infinitely many traces that are all considered equally important. Since the event log only contains example behaviour, one cannot expect to observe all possible traces. These problems can be addressed by adding probabilities.

Given a model with probabilities, we expect the "highways" in the process model to occur frequently in the event log and anticipate that unlikely traces may not appear in the event log.

As demonstrated in this paper, we can only quantify the "fraction of modeled behaviour actually observed" if we add probabilities to models. Such probabilities are not just needed for conformance checking. Process models used for simulation, predictions and recommendations all require probabilities. Therefore, it is natural (and also not so difficult) to add the stochastic perspective. Moreover, treating all paths in the model equally does not only impact the diagnostics of conformance checking, but also limits progress in process discovery. Adding infrequent behaviour to a process model often yields a small improvement in fitness while lowering precision significantly. Hence, process-discovery techniques are tempted to simply leave out infrequent behaviour thus ignoring observed behaviour.

Stochastic conformance checking aims to address these problems by qualifying the "distance" between stochastic languages representing models and logs. In this paper, we presented our Earth Mover's Stochastic Conformance (EMSC) approach and the corresponding implementation. We introduced the so-called 'reallocation matrix' to deal with process models having silent and duplicate activities. Using stochastic trace alignments, we developed intuitive diagnostics projected on event logs and process models. The techniques have also be extended to model-model and log-log comparisons. The approach has been fully implemented in ProM and uses techniques for solving transportation problems to determine the earth movers' distance. This provides a considerable performance improvement as shown in our evaluation. Moreover, the practical applicability of the EMSC approach was demonstrated using 11 publicly available real-life logs and corresponding models.

Our findings show that stochastic conformance checking is practically feasible and helps to address the problems of existing conformance techniques related to precision measures and diagnostics. However, as discussed in Section 7, there are also several open challenges, such as dealing with infinite behaviour without unfolding, non-determinism when computing stochastic trace alignments, and providing aggregated diagnostics.

Future work extends in two directions. The first direction aims to use the approach for questions that go beyond conformance checking. For example, we have implemented concept-drift detection approaches using the same principles. Note that a major shift in the distribution of traces may not impact the control-flow model, e.g., the percentage of cases that skip a check or the number of customers that pay after delivery may grow from 10% to 60% without introducing new process variants. The second direction for future research aims to add additional perspectives covering time, resources, decisions, and costs. Although we focused on the ordering of activities, the exact same idea can be applied to traces and models extended with other perspectives. This requires additional distance measures to introduce penalties for events that are too late or executed by the wrong person. A baseline approach can be realised by making the additional features (time, resource, cost, etc.) discrete and use weights for the

different perspectives. For example, activity *Decide* is split into *Decide(slow)*, *Decide(normal)*, and *Decide(fast)*. In general it is non-trivial to define distance measures covering different perspectives, or to obtain them from event logs. However, the framework presented can be applied directly. Note that also for the other perspectives, probabilities matter.

Acknowledgments. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research, and Merih Seran Uysal for her useful comments on the description of the implementation and its proofs. Artem Polyvyanyy was in part supported by the Australian Research Council project DP180102839.

References

[1] *Minimal Cost Network Flows*, chapter 9, pages 453–512. John Wiley & Sons, Ltd, 2009.

[2] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.

[3] W.M.P. van der Aalst. Relating Process Models and Event Logs: 21 Conformance Propositions. In *Proceedings of the International Workshop on Algorithms and Theories for the Analysis of Event Data (ATAED 2018)*, volume 2115 of *CEUR Workshop Proceedings*, pages 56–74. CEUR-WS.org, 2018.

[4] W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[5] W.M.P. van der Aalst and W. Reisig, editors. *Advanced Tutorial on Petri Net Modelling of Business Processes (Satellite Event of ACSD'06 and ICATPN'06)*, Turku, Finland, June 2006.

[6] Hans Achatz, Peter Kleinschmidt, and Konstantinos Paparrizos. A dual forest algorithm for the assignment problem. In Bernd Sturmfels and Peter Gritzmann, editors, *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, pages 1–10. AMS & ACM, 1991.

[7] A. Adriansyah. *Aligning Observed and Modeled Behavior*. Phd thesis, Eindhoven University of Technology, April 2014.

[8] A. Adriansyah, B. van Dongen, and W.M.P. van der Aalst. Conformance Checking using Cost-Based Fitness Analysis. In C.H. Chi and P. Johnson, editors, *IEEE International Enterprise Computing Conference (EDOC 2011)*, pages 55–64. IEEE Computer Society, 2011.

[9] A. Adriansyah, J. Munoz-Gama, J. Carmona, B.F. van Dongen, and W.M.P. van der Aalst. Measuring Precision of Modeled Behavior. *Information Systems and e-Business Management*, 13(1):37–67, 2015.

[10] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.

[11] Søren Asmussen. *Applied Probability and Queues*, volume 51 of *Applications of mathematics*. Springer New York, 2nd edition, 2003.

[12] Adriano Augusto, Abel Armas-Cervantes, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Daniel Reißner. Abstract-and-compare: A family of scalable precision measures for automated process discovery. In *Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, pages 158–175, 2018.

[13] Tobias Brockhoff, Merih Seran, and Wil M. P. van der Aalst. Time-aware concept drift detection using the earth mover's distance. In *International Conference on Process Mining, ICPM 2020, Padova, Italy, October 6-9, 2020*, page to appear. IEEE, 2020.

[14] J. C. A. M. Buijs and Hajo A. Reijers. Comparing business process variants using models and event logs. In *Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings*, pages 154–168, 2014.

[15] J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. A Genetic Algorithm for Discovering Process Trees. In *IEEE Congress on Evolutionary Computation (CEC 2012)*, pages 1–8. IEEE Computer Society, 2012.

[16] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking: Relating Processes and Models*. Springer-Verlag, Berlin, 2018.

[17] Giovanni Chiola, Marco Ajmone Marsan, Gianfranco Balbo, and Gianni Conte. Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Trans. Software Eng.*, 19(2):89–107, 1993.

[18] M. de Leoni and W.M.P. van der Aalst. Aligning Event Logs and Process Models for Multi-Perspective Conformance Checking: An Approach Based on Integer Linear Programming. In F. Daniel, J. Wang, and B. Weber, editors, *International Conference on Business Process Management (BPM 2013)*, volume 8094 of *Lecture Notes in Computer Science*, pages 113–129. Springer-Verlag, Berlin, 2013.

[19] B.F. van Dongen, J. Carmona, and T. Chatain. A Unified Approach for Measuring Precision and Generalization Based on Anti-alignments. In M.

La Rosa, P. Loos, and O. Pastor, editors, *International Conference on Business Process Management (BPM 2016)*, volume 9850 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, Berlin, 2016.

[20] B.F. van Dongen, J. Carmona, T. Chatain, and F. Taymouri. Aligning Modeled and Observed Behavior: A Compromise Between Computation Complexity and Quality. In E. Dubois and K. Pohl, editors, *International Conference on Advanced Information Systems Engineering (Caise 2017)*, volume 10253 of *Lecture Notes in Computer Science*, pages 94–109. Springer-Verlag, Berlin, 2017.

[21] B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.

[22] Paul A. Gagniuc. *Markov Chains*. John Wiley and Sons Ltd, 2017.

[23] L. Garcia-Banuelos, N. van Beest, M. Dumas, M. La Rosa, and W. Mertens. Complete and Interpretable Conformance Checking of Business Processes. *IEEE Transactions on Software Engineering*, 44(3):262–290, 2018.

[24] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust Process Discovery with Artificial Negative Events. *Journal of Machine Learning Research*, 10:1305–1340, 2009.

[25] Frank L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20(1-4):224–230, 1941.

[26] Gert Janssenswillen, Benoît Depaire, and Christel Faes. Enhancing discovered process models using bayesian inference and mcmc. In *Business Process Management Workshops - BPM 2020 International Workshops, Sevilla, Spain, September 14, 2020, Revised Papers*, volume to appear of *Lecture Notes in Business Information Processing*.

[27] Anna Kalenkova and Artem Polyvyanyy. A spectrum of entropy-based precision and recall measurements between partially matching designed and observed processes. In *ICSOC*, 2020. In Press.

[28] M. Kerremans. Gartner Market Guide for Process Mining, Research Note G00353970. `www.gartner.com`, 2018.

[29] Sander J. J. Leemans and Artem Polyvyanyy. Stochastic-aware conformance checking: An entropy-based approach. In *Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings*, pages 217–233, 2020.

[30] Sander J. J. Leemans, Erik Poppe, and Moe Thandar Wynn. Directly follows-based process mining: Exploration & a case study. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*, pages 25–32. IEEE, 2019.

[31] Sander J. J. Leemans, Shiva Shabaninejad, Kanika Goel, Hassan Khosravi, Shazia Sadiq, and Moe T. Wynn. Identifying cohorts: Recommending drill-downs based on differences in behaviour for process mining. In *Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings*, volume to appear of *Lecture Notes in Computer Science*.

[32] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer-Verlag, Berlin, 2014.

[33] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, 17(2):599–631, 2018.

[34] S.J.J. Leemans, A.F. Syring, and W.M.P. van der Aalst. Earth Movers' Stochastic Conformance Checking. In T.T. Hildebrandt, B.F. van Dongen, M. Röglinger, and J. Mendling, editors, *Business Process Management Forum (BPM Forum 2019)*, volume 360 of *Lecture Notes in Business Information Processing*, pages 127–143. Springer-Verlag, Berlin, 2019.

[35] F. Mannhardt, M. de Leoni, H.A. Reijers, and W.M.P. van der Aalst. Balanced Multi-Perspective Checking of Process Conformance. *Computing*, 98(4):407–437, 2016.

[36] F. Mannhardt, M. de Leoni, H.A. Reijers, and W.M.P. van der Aalst. Measuring the Precision of Multi-perspective Process Models. In M. Reichert and H.A. Reijers, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2015)*, volume 256 of *Lecture Notes in Business Information Processing*, pages 113–125. Springer-Verlag, Berlin, 2016.

[37] M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalised Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[38] J. Munoz-Gama and J. Carmona. A Fresh Look at Precision in Process Conformance. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer-Verlag, Berlin, 2010.

[39] J. Munoz-Gama and J. Carmona. Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 184–191, Paris, France, April 2011. IEEE.

[40] Hoang Nguyen, Marlon Dumas, Marcello La Rosa, and Arthur H. M. ter Hofstede. Multi-perspective comparison of business process variants based on event logs. In *Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings*, pages 449–459, 2018.

[41] Charalampos Papamanthou, Konstantinos Paparrizos, and Nikolaos Samaras. Computational experience with exterior point algorithms for the transportation problem. *Appl. Math. Comput.*, 158(2):459–475, November 2004.

[42] Konstantinos Paparrizos. A non improving simplex algorithm for transportation problems. *RAIRO - Operations Research - Recherche Opérationnelle*, 30(1):1–15, 1996.

[43] Artem Polyvyanyy and Anna A. Kalenkova. Monotone conformance checking for partially matching designed and observed processes. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*, pages 81–88, 2019.

[44] Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. An entropic relevance measure for stochastic conformance checking in process mining. In *ICPM*, 2020. In Press.

[45] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing complexity of large EPCs. In *MobIS*, volume 141, pages 195–207. GfI, 2008.

[46] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. 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.

[47] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, sep 1963.

[48] A. Rogge-Solti, W.M.P. van der Aalst, and M. Weske. Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 15–27. Springer-Verlag, Berlin, 2014.

[49] A. Rogge-Solti, A. Senderovich, M. Weidlich, J. Mendling, and A. Gal. In Log and Model We Trust? A Generalized Conformance Checking Framework. In M. La Rosa, P. Loos, and O. Pastor, editors, *International Conference on Business Process Management (BPM 2016)*, volume 9850 of *Lecture Notes in Computer Science*, pages 179–196. Springer-Verlag, Berlin, 2016.

[50] Andreas Rogge-Solti, Wil M. P. van der Aalst, and Mathias Weske. Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In *BPM Workshops*, pages 15–27, 2013.

[51] A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.

[52] A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Simulation Models. *Information Systems*, 34(3):305–327, 2009.

[53] Arik Senderovich, Matthias Weidlich, Liron Yedidsion, Avigdor Gal, Avishai Mandelbaum, Sarah Kadish, and Craig A. Bunnell. Conformance checking and performance improvement in scheduled processes: A queueing-network perspective. *Inf. Syst.*, 62:185–206, 2016.

[54] Anja F. Syring, Niek Tax, and Wil M. P. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Transactions on Petri Nets and Other Models of Concurrency*, pages 192–221, 2019.

[55] N. Tax, X. Lu, N. Sidorova, D. Fahland, and W.M.P. van der Aalst. The Imprecisions of Precision Measures in Process Mining. *Information Processing Letters*, 135:1–8, 2018.

[56] A.N. Tolstoi. Methods of removing irrational shipments in planning. *Sotsialisticheskii Transport*, 9:28–51, 1939.

[57] Seppe K. L. M. vanden Broucke, Jochen De Weerdt, Jan Vanthienen, and Bart Baesens. Determining process model precision and generalization with weighted artificial negative events. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1877–1889, aug 2014.

[58] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs. *Information Systems*, 37(7):654–676, 2012.

[59] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A Robust F-measure for Evaluating Discovered Process Models. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 148–155, Paris, France, April 2011. IEEE.

[60] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske. Process Compliance Measurement Based on Behavioral Profiles. *Information Systems*, 36(7):1009–1025, 2011.

Appendix A. Exterior Point Simplex for Log-Model Comparisons

In the following, we will use terminology and notation introduced in [42]. The proof can be divided into two parts. First, in order to apply the algorithm, we introduce an artificial demand node on the model side which normalises the total demand. We show that this node will have zero incoming flow as long as there are demand nodes with rest capacity. Secondly, we show that the flow is optimal for the unfolded model problem immediately before the first usage of the new demand node.

We start the proof with five observations that follow directly from the algorithm and its proof in [42]. The first observation concerns the costs of augmenting flow along the path P_t which is created between the two roots of the surplus and deficit trees of the incoming edge in step t.

Observation 1 (Flow Augmentation Cost)**.** *Given the path P_t, δ_t corresponds to the primal costs of augmenting one unit of flow along P_t.*

Proof. This follows from the definition and update for the dual variables and is a general relation between dual slack variables and their corresponding primal variables in simplex type algorithms. □

Notice that augmenting flow along P_t might only violate the flow constraints at the root nodes. The second observation establishes the monotonicity of δ_t.

Observation 2 (Montonicity of δ_t)**.** *For iterations t, t' with $t < t'$, it holds that $0 \leq \delta_t \leq \delta_{t'}$.*

Proof. First, $0 \leq \delta_0$ immediately follows from he AKP forest initialisation method [41] since the resulting forest is dual feasible. Furthermore, $\delta_t \leq \delta_{t+1}$ holds independently from the initialisation method [42, p. 9,13]. □

For the second part of the proof we observe that for every edge in the forest the corresponding dual constraint is tight.

Observation 3 (Complementariy slackness wrt. w_{ij})**.** *Given $(i,j) \in F_t$, it holds that $0 = w_{ij}(F_t) = c_{ij} - u_i(F_t) - v_j(F_t)$.*

Proof. This is a typical relation for simplex type algorithms and follows from the AKP intitialisation method and the update equations of the dual variables of the algorithm. □

The fourths observation concerns the primal flow constraints in the individual subtrees.

Observation 4 (Subtree Flows)**.** *For each node in F_t that is not a root its total outgoing flow (supply node) or total incoming flow (demand node) constraints are satisfied.*

Proof. This follows from the AKP initialisation method and how flows are augmented in the update step. □

Finally, we observe that every root node in F_t corresponds to a demand node.

Observation 5 (Root Nodes)**.** *Given a root node $n \in F_t$, it holds that $n \in J$, i.e., n is a root node.*

Proof. For $n \in F_0$, this follows from the AKP intialisation method. For $n \in F_t, t > 0$, this follows from the forest update routine that only reduces the number of root nodes but never changes a root node nor adds a new tree. □

Given these observations, in order to apply the algorithm, we introduce an additional artificial demand (model) node n_a which normalises the total demand capacity, i.e., $b_{n_a} = 1 - \sum_{j=1}^{n} b_j$. The distance to each of the supply(/log) nodes is defined to be at least twice the total distance, i.e, $c_{in_a} = 2 \sum_{i \in I} \sum_{j \in J} c_{ij} + 1$, with $i = 1 \dots m$.

By introducing this artificial demand node, we reformulate the problem as a general TP problem with equality demand constraints and can then thus apply the algorithm.

Next, we show that the algorithm will not select an edge incident to n_a as long as there are deficit trees remaining, i.e., $F_t^D \neq \varnothing$. To this end, notice that the initial greedy AKP initialisation method will not attach a supply node to n_a. Furthermore, by the definition of c_{in_a}, $i = 1, \dots, m$, the primal costs of augmenting flow along a path between two root nodes will always be less than the costs for augmenting flow along a path that contains an edge (i, n_a) (for $i = 1, \dots, m$). Given that the potential primal cost improvement of augmenting one unit of flow is naturally bounded by the total sum of the distances, a path containing an edge to n_a will have a cost of at least $2 \sum_{i \in I} \sum_{j \in J} c_{ij} + 1 - \sum_{i \in I} \sum_{j \in J} c_{ij}$. Accordingly, using Observation 1, δ_t would not be minimal for an edge (i, n_a), $i = 1, \dots, m$ if $F_t^D \setminus \{n_a\} \neq \varnothing$ holds.

We stop the algorithm if there is no deficit tree left except the artifical demand node, i.e., $F_t^D = \{n_a\}$, and remove n_a. Let F_{t_f} denote the resulting forest. For this solution, we prove that it is optimal for the LP for unfolded model path languages. To this end, we prove primal feasibility, dual feasibility and complementary slackness, which together imply optimality.

Primal Feasibility. By Observation 4 all non-root nodes satisfy their constraints. Furthermore, by Observation 5 every root node is a demand node and the stopping criterion ensures that there is a surplus at the respective root of the tree. Thus, the solution is primal feasible.

Dual Feasibility. In order to show dual feasibility, we start with $w_{kl}(F_t) \geq 0$, $k \in F_0^S, l \in F_0^D$, which follows directly from Observation 2 and the relation:

$$0 \leq \delta_0 \leq \delta_t = \min\{w_{ij}(F_t) | i \in F_t^S, j \in F_t^D\} \leq w_{kl}(F_t)$$

In addition, we have to show that for the last forest F_{t_f}, it holds that $v_j(F_{t_f}) \geq 0, j \in J$. To this end, let $j \in J$ and consider the sequence of all changes of v_j, namely $\left(v_j(F_{t_0^j}), v_j(F_{t_1^j}), \ldots, v_j(F_{t_{f_j}^j})\right)$. That is, for $k, l \in \{0, \ldots, f_j\}, k < l$ and $t_k^j \leq t < t_l^j$, it holds that $v_j(F_{t_k^j}) = v_j(F_t) \neq v_j(F_{t_l^j})$. Since the value of v_j may only change if and only if node j transitions between a surplus tree and a deficit tree, we have $v_j \in F^S_{t_k^j}$ ($v_j \in F^D_{t_k^j}$) if and only if $v_j \in F^D_{t_{k+1}^j}$ ($v_j \in F^S_{t_{k+1}^j}$). Moreover, due to the AKP initialisation method, we have $v_j(F_0) = 0$. In order to prove that $v_j(F_{t_f}) \geq 0$, we consider the cases $j \in F_0^D$ and $j \in F_0^S$, and prove that $j \in F_{t'}^S$ implies that $v_j(F_{t'}) \geq 0$ for $t' = 0, \ldots, t_f$.

- Case $j \in F_0^D$. First, observe that $j \in F_{t'}^S$ if and only if $t' \in \{t_1^j, t_3^j, \ldots, t_{f_j-2}^j, t_{f_j}^j\}$.

  Notice that $t_{f_j}^j$ is included as the algorithm terminates when there is no deficit tree left. The claim follows by induction on $k = 1, 3, \ldots$ proving that $v_j(F_{t_k^j}) \geq 0 \Rightarrow v_j(F_{t_{k+2}^j}) \geq 0$.

  Induction base $k = 1$: by the update rules we have $0 = v_j(F_0) = v_j(F_{t_1^j - 1}) \overset{\delta_{t_1^j} \geq 0}{\leq} v_j(F_{t_1^j - 1}) + \delta_{t_1^j} = v_j(F_{t_1^j})$.

  Induction step $k \to k+2$: applying the update rules using $v_j \in F^S_{t_k^j}$, $v_j \in F^D_{t_{k+1}^j}$ and $v_j \in F^S_{t_{k+2}^j}$ yields

  $$v_j(F_{t_{k+2}^j}) = v_j(F_{t_{k+1}^j}) + \delta_{t_{k+1}^j} = v_j(F_{t_k^j}) \underbrace{-\delta_{t_k^j} + \delta_{t_{k+1}^j}}_{\geq 0\,(\delta_{t_k^j} \leq \delta_{t_{k+1}^j})} \geq v_j(F_{t_k^j}) \geq 0\,.$$

  Thus, the case holds by induction.

- Case $j \in F_0^S$. Analog to $j \in F_0^D$ with induction over $k = 0, 2, \ldots$.

As the algorithm terminates when there is no deficit tree left and hence all demand nodes are in a surplus tree, the solution is dual feasible.

Complementary Slackness. Finally, we show complementary slackness of the primal and dual solution, i.e.,

$$x_{ij}(F_{t_f}) w_{ij}(F_{t_f}) = x_{ij}(F_{t_f}) \left(c_{ij} - u(F_{t_f}) - v(F_{t_f})\right) = 0$$

for $i \in I$ and $j \in J$. This holds for every intermediate forest $F_i, i = 0, \ldots, t_f$ by Observation 3 and the fact that only tree edges have a nonzero flow.