Stochastic Process Discovery By Weight Estimation

Adam Burke (✉), Sander J. J. Leemans, and Moe Thandar Wynn

Queensland University of Technology, Brisbane, Australia,
at.burke@qut.edu.au,s.leemans@qut.edu.au,m.wynn@qut.edu.au

Abstract. Many algorithms now exist for discovering process models from event logs. These models usually describe a control flow and are intended for use by people in analysing and improving real-world organizational processes. The relative likelihood of choices made while following a process (i.e., its stochastic behaviour) is highly relevant information which few existing algorithms make available in their automatically discovered models. This can be addressed by automatically discovered stochastic process models.
We introduce a framework for automatic discovery of stochastic process models, given a control-flow model and an event log. The framework introduces an estimator which takes a Petri net model and an event log as input, and outputs a Generalized Stochastic Petri net. We apply the framework, adding six new weight estimators, and a method for their evaluation. The algorithms have been implemented in the open-source process mining framework ProM. Using stochastic conformance measures, the resulting models have comparable conformance to existing approaches and are shown to be calculated more efficiently.

Key words: Stochastic Petri nets, process mining, stochastic process mining, stochastic process discovery

1 Introduction

The world abounds in information systems, generating data about the processes they mediate, execute, or observe. Using this data to compute and analyze process models is the concern of process mining [3], within the field of Business Process Management (BPM). BPM studies the impact and improvement of processes in organizations. Automatic process discovery is one aspect of process mining concerned with finding a formal process model computationally from an input event log.

To understand a process, we often want to know how likely an event is. If we travel to work, a journey where our train reliably arrives on time is different from one where the train sometimes breaks down, is sometimes replaced by a bus, or is often so crowded that it's quicker to ride a bike. A highly contagious disease with rare side effects differs importantly from one difficult to transmit but with severe side effects, even if observable symptoms are similar. Detecting fraud in financial transactions depends on recognizing certain client actions happening more frequently than usual. Existing process mining techniques already recognize this: where noise or probability is considered in creating control flows (e.g. [30, 19]), they acknowledge the importance of likelihood in process modeling. Better stochastic representations and stochastic-aware techniques have been flagged as a key research challenge for process mining [2].

Process discovery techniques have become quite sophisticated at determining causal relationships between activities from event logs, and representing that in process models. There are far fewer techniques for discovering relative probabilities (discussed in Section 5). We introduce a framework in Section 3 which leverages this by allowing transformation of models with only control flows into stochastic process models. This extends an existing stochastic process discovery technique by Rogge-Solti et al (RSD) [25, 26], in two ways. Firstly, it generalizes one estimation algorithm to a general class of *weight estimators*. Secondly, it specializes the possible outputs from general probability distributions to Generalized Stochastic Petri Nets (GSPNs) [4]. The framework does not prescribe whether the estimation calculation is deterministic, uses stochastic simulation, or other techniques, and our introduced estimators include both deterministic and non-deterministic types.

We describe our approach as a form of Stochastic Process Discovery, as it takes an event log input and produces a GSPN output. In decoupling weight estimation from control flow discovery, the technique also shares some features with process model enhancement for time and probability [3, p290]. Unlike enhancement techniques, estimators can potentially change control flows when producing a stochastic process model. Stochastic process models have a corresponding, emerging, set of stochastic process conformance measures [20, 21, 16]. Consequently, the algorithms and models presented here are evaluated, in Section 4, as stochastic process discovery algorithms, using stochastic process conformance measures. Evaluation, which also includes performance, is against real-life event logs, multiple control flow discovery algorithms, and RSD [25].

In the next section, we introduce existing concepts. In Section 3, we describe the weight estimation framework and instantiate it by introducing novel estimators. In Section 4, the results of using the estimators on real-world event logs are presented. Related work is reviewed in Section 5, and Section 6 concludes the paper.

2 Preliminaries

Petri nets and Generalized Stochastic Petri Nets are well-established formalisms for modelling processes and a number of good overviews exist [4, 8]. We use notations from the process mining literature [3, 21].

A Petri net is a tuple $PN = (P, T, F, M_0)$, where P is a finite set of places, T is a finite set of transitions, and $F : (P \times T) \rightarrow (T \times P)$ is a flow relation. A *marking* is a multiset of places $\subseteq P$ that indicate a state of the Petri net, with M_0 the initial marking. A transition is enabled if every incoming place contains a token. A transition fires by changing the marking of the net to consume incoming tokens and producing tokens for its outgoing transitions. For a node $n \in P \cup T$, we define $\bullet n = \{y \mid (y, x) \in F\}$ and $n\bullet = \{y \mid (x, y) \in F\}$.

A *Generalized Stochastic Petri Net (GSPN)* is a tuple $(P, T, F, M_0, W, T_i, T_t)$ such that $T_i \subseteq T$, $T_t \subseteq T$ and $T_i \cap T_t = \emptyset$. Weight function $W\colon T \rightarrow \mathbb{R}^+$ assigns each transition a weight. T_i is a set of immediate transitions. If multiple transitions $T_e \subseteq T_i$ are enabled in a particular marking, the probability of a transition $t \in T_i$ firing is given by $\frac{W(t)}{\Sigma_{t' \in T_e} W(t')}$. T_t is a set of timed transitions. Immediate transitions take priority over timed transitions. A timed transition, if enabled, fires according to an exponentially distributed wait time. Given a set of enabled timed transitions $T_e \subseteq T_t$, a particular transition t fires first with probability $\frac{W(t)}{\Sigma_{t' \in T_e} W(t')}$ [4].

Event logs. A process consists of activities from the set $\mathcal{A}$. A trace is a non-empty sequence of activities, and an event log L is a finite multiset of traces observing the underlying process. Partial function $\lambda : T \rightarrow \mathcal{A}$ designates labels for Petri net transitions that represent log activities. The number of traces in a log L is denoted with $|L|$, while the the number of events is denoted with $||L||$.

Control Flow Process Discovery. A process discovery algorithm for Petri Nets is then defined by $cfd : L \rightarrow (P, T, F, M_0)$.

Sequence operations. A finite sequence over $\mathcal{A}$ of length n is a mapping $\sigma \in \{1..n\} \rightarrow \mathcal{A}$ and denoted by $\sigma = \langle a_1, a_2, ..., a_n \rangle$ where $\forall_i a_i = \sigma(i)$. Concatenation operator $+$ appends one sequence to another such that $\langle a_1, ..., a_n \rangle + \langle b_1, ..., b_m \rangle = \langle a_1, ...a_n, b_1, ..., b_m \rangle$. The tail function is then $tail(\langle a \rangle + \sigma) = \sigma$.

Subsequence. Function ct returns the number of times a subsequence is present in a sequence: $\mathrm{ct}(\varsigma, \sigma) = \begin{cases} 0 & \text{if } \sigma = \langle\rangle \\ 1 + \mathrm{ct}(\varsigma, tail(\sigma)) & \text{if } \sigma = \varsigma + x \\ \mathrm{ct}(\varsigma, tail(\sigma)) & \text{if } \sigma \neq \varsigma + x \end{cases}$

Alignments. An alignment [1] represents paired paths between a log and a model. That is, a move is a tuple where (a, t) represents a synchronous move on activity a in a trace and a transition t in the model (with the same label: $\lambda(t) = a$), $(a, \perp)$ represents a log move, and $(\perp, t)$ represents a model move. For our purposes, we assume that a function γ is available taking a Petri net, a set of final markings and an event log, and that γ returns a sequence of move tuples that represent all moves necessary to align every trace in the log.

3 Stochastic Process Model Weight Estimation

In this section, we first introduce our framework to transform a Petri net into a GSPN using an event log. Then, we introduce six estimators using the framework, which we will illustrate using the running example shown in Figure 2. Estimators are a large solution space with many potential algorithms. Our six estimators are chosen to emphasize broad applicability of inputs, computational tractability, using the implicit causal information in control flow models, and reapplying established process mining concepts.

3.1 A Framework for GSPN Discovery

The framework defines functions which together transform an event log into a GSPN, as shown in Figure 1.

A stochastic process discovery algorithm for GSPNs (*mine_spn*) is a function $mine\_spn : L \rightarrow (P, T, F, M_0, W, T_i, T_t)$. Our framework considers functions of the form $mine\_spn = est(cfd(L), L)$. Functions $est : L \times (P, T, F, M_0) \rightarrow (P, T, F, M_0, W, T_i, T_t)$ are termed *estimators*.

Functions $se : L \times (P, T, F, M_0) \rightarrow T \times \mathbb{R}^+$ are *simple weight estimators* and use the control flow of the input Petri net intact in the output Petri net, such that for discovered control flow model $cfd(L) = (P_d, T_d, F_d, Md_0)$,

$$\exists_{pe \in est} pe = (P_d, T_d, F_d, Md_0, se(L, (P_d, T_d, F_d, Md_0)), T_d, \emptyset)$$

The estimators discussed next are of this simpler form.

Specific estimators may have further restrictions on their inputs, or provide guarantees on their outputs. For example, estimators discussed below do not distinguish transitions with duplicate labels. A challenge common to several estimators is treatment of silent transitions, as those transitions in a discovered model serve a structural role and do not directly represent an activity in the log. Assigning such a transition a weight of zero in a stochastic net is equivalent to deleting the transition, and all subsequent model paths. To avoid this impact, default values are assigned to silent transitions where the calculation would otherwise result in zero weights. In general, estimators make no distinction between silent transitions and transitions without a corresponding activity in the log. In the remainder of this section, we introduce several examples of estimators that instantiate this framework.

3.2 Frequency Estimator

The first estimator, w_{freq}, straightforwardly uses how often each transition t appeared in the event log L:

$$w_{freq}(L, t) = \max(1, \Sigma_{\sigma \in L} \operatorname{ct}(\langle \lambda(t) \rangle, \sigma))$$

Silent transitions are assigned the arbitrary weight of 1, equivalent to a single observation in the log. The complexity of this estimator is linear in the number of events in the log. Figure 2c shows the results of this estimator on our running example, e.g. $w_{freq}(EL, b) = 15$.

3.3 Activity-Pair Frequency Estimators

An Activity-Pair Estimator uses the frequency of pairs of successor activities to better reflect the constraints of more general Petri nets. These are *edge-structured estimators*, in that Petri net edges inform the weighting.

We first introduce some frequency definitions. The functions q_I and q_F capture how often an activity appears as the first/last in a trace. The function q_P captures the frequency of activity pairs in the log, that is, where the two given activities follow one another directly in the log:

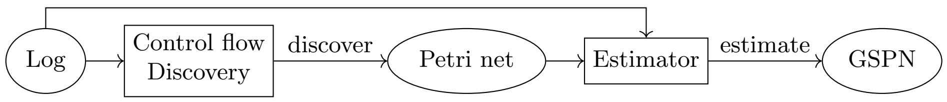
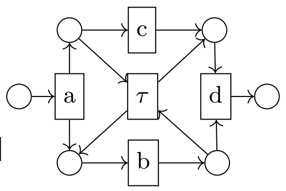


Fig. 1: Our framework for GSPN Discovery.

$[\langle a,b,c,d\rangle^5,$
$\langle a,c,b,d\rangle^4,$
$\langle a,b,b,d\rangle^2,$
$\langle a,b,c,b,d\rangle]$

(a) Log EL.



(b) Petri net EPN.

	w_{freq}	w_{lhpair}	w_{rhpair}	$w_{pairscale}$	w_{fork}	w_{align}
a	12	12	12	$1\frac{11}{49}$	12	12
b	15	8	7	$\frac{35}{49}$	8	14
c	10	4	5	$1\frac{1}{49}$	$4\frac{10}{11}$	9
d	12	12	12	$1\frac{11}{49}$	$11\frac{6}{13}$	12
τ	1	1	1	1	$\frac{129}{143}$	0

(c) Six example estimators.

Fig. 2: Running example of an event log and a Petri net, and the estimators.

$$q_I(L,t) = |[\langle \lambda(t), \ldots\rangle \in L]|$$
$$q_F(L,t) = |[\langle \ldots, \lambda(t)\rangle \in L]|$$
$$q_P(L,s,t) = \Sigma_{\sigma \in L}\, \mathrm{ct}(\langle \lambda(s), \lambda(t)\rangle, \sigma)$$

There are both left-handed and right-handed variants of the Activity-Pair estimator, depending on whether weights are informed by successor or predecessor transitions, defined as:

$$w_{lhpair}(L,t) = \max(1, q_I(L,t) + q_F(L,t) + \sum_{s \in \bullet(\bullet t)} q_P(L,s,t))$$

$$w_{rhpair}(L,t) = \max(1, q_I(L,t) + q_F(L,t) + \sum_{s \in (t\bullet)\bullet} q_P(L,t,s))$$

There are no restrictions on input Petri nets and they can be calculated in time $O(||L|||F|)$, that is, the number of events times the number of model edges.

When using activity pair frequency data, two important types of path through the model are neglected for any given trace: paths from the initial place to the first transition, and the paths from the last transition to the final place. Traces of length one are also invisible from this perspective. To account for this, how often an activity appears as the initial or final activity in a trace is also included in the weight estimation. Note that not all activity pairs occurring in the log are used to calculate the resulting transition weights. For instance, where a given Petri net represents two transitions a and b as concurrent, the frequency of $\langle a, b\rangle$ will not be used. In our running example (see Figure 2c), $w_{lhpair}(EL, c) = 4$ and $w_{rhpair}(EL, c) = 5$.

3.4 Mean-Scaled Activity-Pair Frequency Estimator

The previous estimators depend on the size of the log. Two logs with the same traces in the same ratios will result in two models with two distinct sets of weights, which challenges human analysis. Though comparison and comprehensibility of stochastic process models appears not to have been directly addressed in the literature, it is consistent with research that finds "small variations between models can lead to significant differences in their comprehensibility" [24] and the usability principle of minimizing user memory load. The mean-scaled activity-pair estimator $w_{pairscale}$ mitigates this effect by scaling weights by average transition frequency $(\frac{||L||}{|T|})$ in the log L:

$$pairscale(L,T,t) = \frac{q_I(L,t) + q_F(L,t) + \sum_{s \in (t\bullet)\bullet} q_P(L,t,s)}{\frac{||L||}{|T|}}$$

$$w_{pairscale}(L,(P,T,F,M_0),t) = \begin{cases} pairscale(L,T,t) & \text{if } pairscale(t) \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

One effect of defaulting after scaling is that silent or unrepresented transitions are weighted more heavily, that is, the same as an activity of mean-frequency, rather than the equivalent of an activity occurring once in the log. In our running example of Figure 2c, $||L|| = 49$, $|T| = 5$ and the numerator of *pairscale* is equal to w_{rhpair} for a, b, c and d. Then, for instance $w_{pairscale}$ of c is $\frac{10}{\frac{49}{5}} = 1\frac{1}{49}$.

3.5 Fork Distribution Estimator

The Fork Distribution Estimator w_{fork} uses a two-stage approach: it first assigns weights to each place in a Petri net using activity-pair frequencies. Second, it distributes those weights to transitions according to the activity frequency in the event log.

$$pw(L,p) = \begin{cases} |L| & \text{if } p \in M_0 \\ \Sigma_{s \in \bullet p} \Sigma_{t \in p\bullet} q_P(L,s,t) & \text{otherwise} \end{cases}$$

$$placeWeights(L,p) = \max(1, pw(L,p))$$

$$w_{fork}(L,(P,T,F,M_0),t) = \Sigma_{p \in \bullet t} placeWeights(L,p) \frac{w_{freq}(t)}{\Sigma_{t'}^{p\bullet} w_{freq}(t')}$$

This estimator only applies to Petri nets which have at least one place without incoming edges, such as workflow nets [3, p81]. This is an edge-structured estimator informed by the structure of the input net. The complexity is $O(||L||\ |F|)$. The w_{fork} estimator shares similarities with the Alpha algorithm [3, p167], in that it treats a place as defining a neighbourhood of related activities represented as transitions. In our example (Figure 2), let p_1 be the top-right place and p_2 the bottom-right place. Then, $pw(EL,p_1) = q_P(c,d) + q_P(\tau,d) = 5$, $pw(EL,p_2) = q_P(\tau,d) + q_P(b,d) = 7$, $placeWeights(EL,p_1) = 5$, $placeWeights(EL,p_2) = 7$ and w_{fork} of $d = 5\frac{12}{12} + 7\frac{12}{13} = 11\frac{6}{13}$.

3.6 Alignment Estimator

The estimator w_{align} applies alignments [1] to estimate weights. To this end, it counts the number of times a transition t appears either as a model move or as a synchronous move in the alignments:

$$w_{align}(L,PN,M_F,t) = |[(x,t) \in \gamma(PN,M_F,L)]|$$

This algorithm only applies to Petri nets with at least one final marking. The time complexity is $O(|T|\ |\gamma|)$ plus the time to compute γ. The alignment estimator has similarities with RSD [25], which fits duration distributions to aligned logs. In our example of Figure 2, the last trace of log EL does not fit the model EPN, as b is executed a second time and c is executed. Thus,

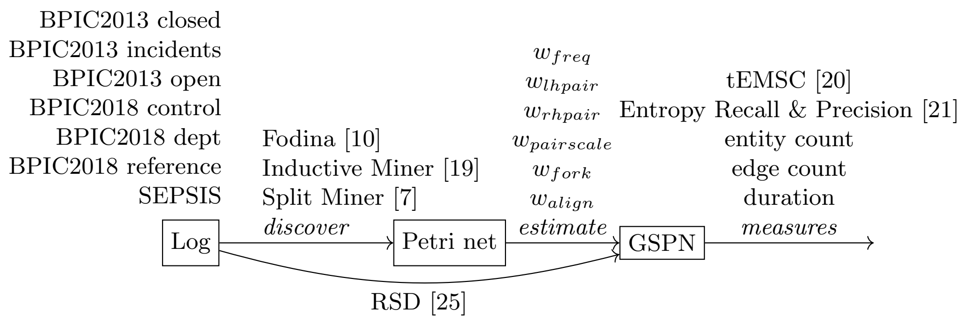


Fig. 3: Set-up of the evaluation.

alignments will (based on a cost function, or if that does not discriminate the options an arbitrary choice) include a log move on either b or a log move on c. If the alignments choose a b for a log move, then $w_{align}(EL, EPN, M_F, b) = 14$ and $w_{align}(EL, EPN, M_F, \tau) = 0$. Alignments are not always deterministic, and consequently neither is w_{align}.

4 Implementation and Evaluation

4.1 Evaluation Design

The six estimators introduced in Section 3 were implemented in the ProM framework [13][1]. For our evaluation, a discovery algorithm was applied to an event log. Where necessary, the result was converted to a Petri net. Each estimator was invoked on the resulting Petri net, resulting in a GSPN. Finally, the conformance of the resulting GSPN was measured against the original log. For comparison, an existing stochastic discovery algorithm by Rogge-Solti et al [25] (RSD) was also applied to the log. This direct discovery algorithm also outputs GSPNs, and the same conformance measures were applied. The implementation of this plugin in ProM 6.9 uses the Inductive Miner internally as an initial control flow discovery step, which has been updated from the gradient-descent procedure described in [25]. Algorithms, reference event logs and conformance measures are summarized as Figure 3.

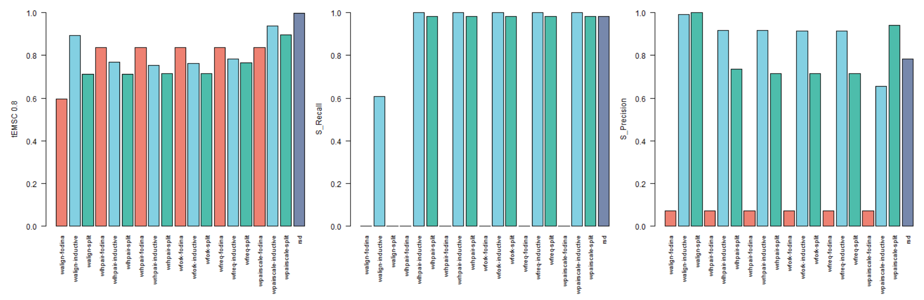
Measures include (1) Truncated Earth Movers' Distance (tEMSC) [20] provides a measure expressing the cost of transforming the distribution of activity traces from one stochastic language into another. We use a minimum probability mass parameter setting of 0.8 for feasibility. (2) Entropy Precision and Recall [21], are stochastic conformance measures based on the entropy of equivalent automata constructed from a given log or model. (3) Petri net entity count (places and transitions) and (4) edge count are used as structural simplicity measures, ensuring that conformance quality has not been achieved by sacrificing model simplicity and comprehensibility. Entity and arc counts have existing uses in process model evaluation [14, 17], and were preferred here over behavioural simplicity measures [16], though these measures also have limitations, including specificity to Petri nets, and insensitivity to the stochastic perspective of GSPNs.

[1] Source code is accessible via `https://github.com/adamburkegh/spd_we`

The duration of a discovery process was also captured, and direct discovery times are compared with combined runtimes for discovery and estimation.

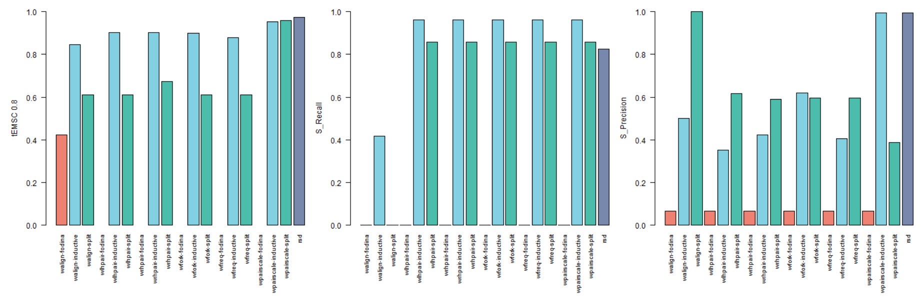
The experiments were run on a Windows 10 machine with 2.3GHz CPU and 50 Gb of memory allocated to each process on JDK 1.8.0_222. All logs are publicly available at `https://data.4tu.nl/`. The full results for these experiments are available in an accompanying technical report [11].

4.2 Results and Discussion



(a) tEMSC (b) entropy-recall (c) entropy-precision

Fig. 4: Results on BPIC 2018 Control log categorized by {*estimator*}-{*control flow algorithm*}, plus RSD.



(a) tEMSC (b) entropy-recall (c) entropy-precision

Fig. 5: Results on BPIC 2018 Reference log.

The estimators produced different, relevant, stochastic models when applied to a range of real-life logs. As seen in Figures 4 and 5, stochastic conformance for these models was comparable, but not uniformly better, than existing techniques, and was highly dependent on the discovery algorithm, and log.

The estimators combined well with the Inductive Miner and Split Miner control discovery algorithms. Frequency-based estimators combined poorly with

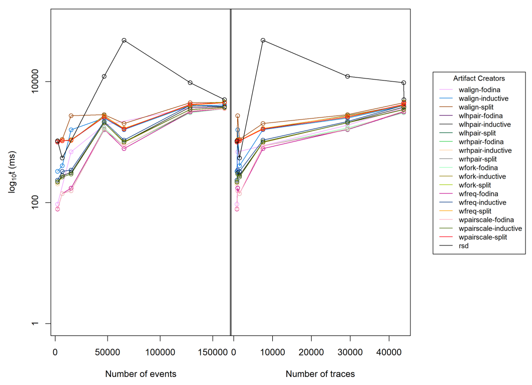


Fig. 6: Run times for control flow discovery and weight estimation by event and trace count. 12 hour time out for RSD [25] on sepsis log is excluded.

the Fodina discovery algorithm for some logs. This is at least partly due to Petri net representational bias in the presented framework. Fodina outputs a causal net, which was converted to a Petri net. The resulting Petri net includes a large number of silent transitions, often intermediating between transitions corresponding to activity pairs in the log. This can be seen distinctly in results for BPIC 2018 reference log in Figure 5, where w_{align} produces a stochastically relevant model on the output of a Fodina input, but no other estimator does. For Split Miner and Inductive Miner, though they use other representations internally, the Petri net model produced used fewer silent transitions and were less impacted by this property.

For the BPIC 2013 closed and incidents logs, Fodina returned a model without an initial place, to which w_{fork}, w_{align}, tEMSC and Entropy-Recall and Entropy-Precision conformance measures do not apply. For some algorithm-estimator combinations, these conformance measures could not be calculated due to soundness, time or memory constraints. Nevertheless, in these results it is clear that tEMSC 0.8 is more sensitive to the stochastic perspective produced by estimators than the Entropy Precision and Recall measures. Where RSD [25] produced a model on which measures could be calculated, the resulting models often conformed well to the logs, but not consistently better than the estimator-produced models. There were a number of event logs where RSD returned no model within the constraints of time (12 hour timeout) and machine memory, or where conformance measures were unable to be calculated within time (5 hour timeout) and memory constraints.

The run time of the estimators, which took never more than 10 seconds, was always comparable or better than RSD, orders of magnitude better in some cases, as shown in Figure 6. In the future, we aim to extend these experiments with larger logs containing more traces, events, and activities. However, even though our estimators returned results for each model and log combination quickly, the conformance measures were the limiting factors in these experiments in terms of time and memory, which indicates that future research should be directed towards more efficient stochastic conformance checking techniques.

In summary, our new estimators, even the alignment-based w_{align}, are able to handle real-life event logs and outputs from existing discovery techniques much faster than existing approaches. Depending on the applied discovery technique, they can also achieve higher stochastic quality, providing alternatives to the existing RSD discovery technique when analyzing control flow and stochastic perspectives.

5 Related Work

Significant work exists on performance analysis using process mining and Stochastic Petri Nets (SPNs) with pre-existing normative models. This includes improving parameters from an input SPN [22, 29, 26], from models in UML [9], and industrial case studies [26, 9]. These and other applications can benefit directly from automatic discovery of stochastic models.

RSD [25] is a technique, with publicly available implementation, for discovering Generally Distributed Transition Stochastic Petri Nets (GDT_SPNs), with some high level descriptions of techniques and algorithms preceding it [18, 15, 6]. RSD first discovers a control flow model in the form of a Petri net, then performs a fitness calculation, and attempts to repair the model if fitness is low. An alignment and replay calculation then informs the production of an output GDT_SPN. The distinction between control flow discovery and stochastic perspectives is extended by our proposed framework to many possible weight estimators. The post-control flow discovery steps in RSD are a *weight estimator*, but not a *simple estimator*, in our terminology.

In [27, 28], queues are discovered in stochastic process mining using two formalisms, Process Trees [28] and Queue-Enabling Colored Stochastic Petri Nets [27]. The Process Tree approach is informed by statistics theory and uses both Bayesian and Markov-Chain Monte-Carlo fitting.

Hidden Markov Models (HMMs) have seen some applications to stochastic process discovery [12, 5]. For instance, [12] constructs HMMs for resource usage using a variant of the Alpha algorithm [3, p167], an early process mining algorithm with known weaknesses on real-world event data. [5] uses event log data to prune unlikely paths from a HMM process model in the context of a semi-automated stochastic process discovery procedure.

Declarative process models describe a process in terms of constraints on behaviour. This contrasts with control-flow based process models, such as Petri nets used in our framework, which describe permitted behaviour. Techniques for automatic process discovery of probabilistic declarative models also exist [23]. Transforming the significant differences between the forms of control-flow and declarative models, and evaluating the result for stochastic conformance, put rigorous comparison beyond the scope of this paper.

6 Conclusion

The likelihood of an event is important information in understanding many real-world processes. Automatically discovered stochastic process models may then help analyze and improve organizations. In this paper we presented a framework for discovery of General Stochastic Petri Nets (GSPNs) from logs. The framework leverages existing control flow discovery algorithms, and introduces *estimators* which transform discovered Petri nets into GSPNs. We introduced six estimators; their implementation is publicly available, and evaluated against real-life logs using multiple stochastic conformance measures. The evaluation used three existing flow discovery algorithms, and an existing stochastic discovery technique, finding models of comparable quality, across a broader range of logs, in a generally shorter time.

The estimators presented here are not exhaustive, and we look forward to future research on novel, improved estimators. The estimator framework also implies the possibility of "direct stochastic discovery" algorithms which do not use a separate control flow algorithm, but produce a control flow model as a side-effect of a stochastic one. A simplicity measure sensitive to both structural representation and stochastic information in a process model would be a useful evaluation tool for work in this area, and is an avenue of future research.

Acknowledgement. Computational resources used included those provided by the eResearch Office at QUT.

References

[1] Wil M. P van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. "Replaying history on process models for conformance checking and performance analysis". In: *DMKD* 2.2 (2012), pp. 182–192.

[2] Wil M. P. van der Aalst. "Academic View: Development of the Process Mining Discipline". In: Springer, 2020, pp. 181–196.

[3] Wil van der Aalst. *Process Mining: Data Science in Action.* 2nd ed. Berlin Heidelberg: Springer-Verlag, 2016.

[4] M. Ajmone Marsan et al. "The effect of execution policies on the semantics and analysis of stochastic Petri nets". In: *TSE* (1989).

[5] Amirah Mohammed Alharbi. "Unsupervised Abstraction for Reducing the Complexity of Healthcare Process Models". PhD thesis. University of Leeds, July 2019.

[6] Nikolas Anastasiou and William Knottenbelt. "Deriving coloured generalised stochastic petri net performance models from high-precision location tracking data". In: *PE.* 2013, pp. 375–386.

[7] Adriano Augusto et al. "Split miner: automated discovery of accurate and simple business process models from event logs". In: *KaIS* (2019).

[8] F. Bause and P.S. Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory.* Vieweg+Teubner Verlag, 2002.

[9] Simona Bernardi et al. "A systematic approach for performance evaluation using process mining: the POSIDONIA operations case study". In: QUDOS. 2016, pp. 24–29.

[10] Seppe K. L. M. vanden Broucke et al. "Fodina: A robust and flexible heuristic process discovery technique". In: *DSS* (2017), pp. 109–118.

[11] Adam Burke et al. *Report On Stochastic Process Discovery By Weight Estimation Experimental Results*. Tech. rep. `https://eprints.qut.edu.au/204662/`. Sept. 2020.

[12] Berny Carrera et al. "Constructing probabilistic process models based on hidden Markov models for resource allocation". In: *BPM*. 2014.

[13] Boudewijn F. van Dongen et al. "The ProM Framework: A New Era in Process Mining Tool Support". In: *Petri Nets*. 2005, pp. 444–454.

[14] Volker Gruhn and Ralf Laue. "Adopting the Cognitive Complexity Measure for Business Process Models". In: *CI*. 2006, pp. 236–241.

[15] Haiyang Hu, Jianen Xie, and Hua Hu. "A novel approach for mining stochastic process model from workflow logs". In: *JCIS* (2011).

[16] Anna Kalenkova et al. "A Framework for Estimating Simplicity of Automatically Discovered Process Models Based on Structural and Behavioral Characteristics". In: *ICPM*. 2020.

[17] Krzysztof Kluza et al. "Square Complexity Metrics for Business Process Models". In: *ABICT*. Springer, 2014, pp. 89–107.

[18] Edouard Leclercq et al. "Identification of timed stochastic Petri net models with normal distributions of firing periods". In: *IFAC* (2009).

[19] Sander J. J. Leemans et al. "Discovering block-structured process models from event logs-a constructive approach". In: *Petri nets*. 2013.

[20] Sander J. J. Leemans et al. "Earth movers' stochastic conformance checking". In: *BPM forum*. Springer, 2019, pp. 127–143.

[21] Sander J. J. Leemans et al. "Stochastic-Aware Conformance Checking: An Entropy-Based Approach". In: *CAiSE*. 2020, pp. 217–233.

[22] Chuang Lin et al. "Performance equivalent analysis of workflow systems based on stochastic petri net models". In: *CoopIS*. 2002, pp. 64–79.

[23] Fabrizio Maria Maggi, Marco Montali, and Rafael Peñaloza. "Probabilistic Conformance Checking Based on Declarative Process Models". en. In: *CAiSE*. 2020, pp. 86–99.

[24] Jan Mendling, Hajo A. Reijers, and Jorge Cardoso. "What Makes Process Models Understandable?" In: *BPM*. 2007, pp. 48–63.

[25] Andreas Rogge-Solti et al. "Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs". In: *BPM workshops*. 2014, pp. 15–27.

[26] Andreas Rogge-Solti et al. "Prediction of business process durations using non-Markovian stochastic Petri nets". In: *IS* (2015).

[27] Arik Senderovich et al. "Data-driven performance analysis of scheduled processes". In: *BPM*. 2016, pp. 35–52.

[28] Arik Senderovich et al. "Discovering Queues from Event Logs with Varying Levels of Information". In: *BPM workshops*. 2016, pp. 154–166.

[29] Loukas C. Tsironis et al. "Fuzzy Performance Evaluation of Workflow Stochastic Petri Nets by Means of Block Reduction". In: *ToS* (2010).

[30] A.J.M.M. Weijters and J.T.S. Ribeiro. "Flexible Heuristics Miner (FHM)". In: *CIDM*. 2011, pp. 310–317.