Significant Stochastic Dependencies in Process Models

Sander J.J. Leemans[a,*], Lisa L. Mannel[a], Natalia Sidorova[b]

[a] *RWTH, Aachen, Germany*
[b] *Eindhoven University of Technology, the Netherlands*

Abstract

Process mining aims to artificially derive meaningful information from event logs recorded from information systems that support business processes in organisations. In many real-life processes, decisions do not only depend on the current state of the model, but also on decisions made earlier in the process. Such dependencies challenge process mining techniques: Petri nets are limited to hard dependencies (non-free-choice constructs). In this paper, we study stochastic dependencies, which model that the likelihood of decisions in a process may change dynamically based on earlier decisions. We introduce a modelling formalism that supports stochastic dependencies by extending stochastic labelled Petri nets, we study symmetries of this formalism, introduce a stochastic process discovery technique that discovers such models and we adapt two conformance checking techniques to validate the discovered models. The techniques have been implemented, and evaluated on computational feasibility and applicability. Finally, we show that the quality of our new models can compete with existing discovery techniques, and we show that stochastic dependencies are present in existing real-life logs and may lead to new types of insights.

Keywords: process mining, stochastic process mining, stochastic process discovery, model dependencies, long-distance dependencies

1. Introduction

Many processes in organisations are supported and monitored by information systems. These systems track the process executions and store extensive information on various aspects of the process. Such data can be extracted in the form of an *event log* – a collection of sequences of events that represent the execution of process activities for cases in the process. The research area of process mining focuses on automatically or semi-automatically deriving insights and intelligence from event logs which can be used to better understand and to improve the process, or to predict ongoing cases. Particularly, in the

*Corresponding author
Email addresses: s.leemans@bpm.rwth-aachen.de (Sander J.J. Leemans), lisa.mannel@rwth-aachen.de (Lisa L. Mannel)

sub-field of process discovery the goal is to automatically identify relevant relations between the activities of the process, such as choices, concurrency and loops, and represent them in a process model. A process model can then be used in further analysis, for instance for performing root-cause analysis using automated simulation or for recommending interventions in ongoing cases to optimise outcomes.

For both of these examples, it is not sufficient that the model only contains the control flow, that is, the possible sequences of activities. Additionally, the model should also express the likelihood of behaviour: the impact of performance issues in a process depends on the relative frequency of the issue; simulations require knowledge of the distribution of choices; and recommending interventions requires knowledge of the likelihood of outcomes further down the process. Stochastic process models, such as stochastic labelled Petri nets (SLPNs), express such a stochastic perspective: whenever the model must make a choice between executing two transitions, the relative weight of each transition indicates their likelihood of occurring next. Stochastic process discovery techniques aim to discover a stochastic process model from an event log.

Most existing stochastic process models, including SLPNs, and most discovery techniques assume that decision points in the process are independent, i.e., a decision made at one point in the process does not impact later decisions. However, in real-life processes, this assumption is unrealistic: one cannot assume all decisions in the process to be completely independent of each other. For example, in a manufacturing process, the likelihood of detecting the first defect at an inspection might be lower than the likelihood of detecting a second defect after the first defect is repaired; in a sales process, actions taken by the organisation can influence the likelihood of making a sale; and in a process of a student attempting exams for a course, passing an exam a second time might have a lower probability than passing the exam the first time.

In Petri nets, complex dependencies can be modelled using non-free choice constructs. These constructs represent *hard dependencies*: rigid constraints of the control-flow, where making a decision restricts the possible choices that can be made afterwards. Several discovery algorithms have been proposed to discover Petri nets including such constructs, e.g. [1, 2], however these techniques provide no indication of the reliability of their conclusions and do not provide statistical guarantees of the discovered constructs. Furthermore, real world dependencies are not always hard: A student who successfully passed an exam might still take part in the re-examination in order to improve the grade, even if the probability of choosing for a re-examination is low for this student (and he will pass the re-examination with a high probability). A student who failed the exam will most likely take part in the re-examination, and the probability he will pass the re-examination is not that high. Soft dependencies (or *stochastic dependencies*) describe influences that a decision or an outcome of an earlier action has on the likelihood of future behaviour.

Models describing stochastic dependencies, particularly dynamic changes of the related likelihoods, can enable a process owner to gain valuable insights which traditional stochastic models are unable to provide. For instance, con-

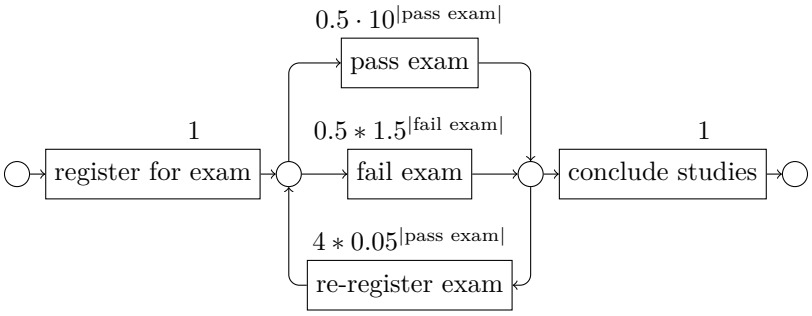


Figure 1: Motivational example for stochastic dependencies.

sider the motivational example shown in Figure 1, which models the study path of students for a particular exam. In general, students may take the exam arbitrarily often: they may re-register for another attempt both after failing (in order to pass the course) and after passing (to try improving their grade). Finally, pass or fail, a student can decide to conclude their studies of this subject. Most traditional control-flow discovery techniques will have no trouble to discover a model representing this behaviour, and stochastic discovery algorithms can assign likelihoods to the various paths, e.g. by showing that in 40% of cases the registration for exam is followed by passing it and in 60% by failing.

Instead of suggesting that the probability to pass the exam is the same for a student who is taking it for the first time as the one of the student who already passed it, the model shown in Figure 1 includes stochastic dependencies which reveal correlations between events. It shows, e.g., that passing the exam decreases the likelihood of re-registering (every execution of 'pass exam' decreases the base probability of re-registration, which is 4, due to multiplication with $0.05^{|\text{pass exam}|}$) but strongly increases the likelihood of passing the exam again. Similarly, according to this model, the more often a student fails, the lower the probability of that student to pass this exam. Clearly, such dependencies carry information that can lead to valuable insights and contribute significantly to the understanding of the underlying process, but they cannot be derived by existing discovery techniques or stochastic process discovery techniques.

In this paper, we introduce several techniques to study stochastic dependencies. We propose stochastic labelled Petri nets with stochastic dependencies (SLPN-SD) – a new modelling formalism that extends the idea presented in [3] to represent stochastic dependencies in stochastic labelled Petri nets and investigate its inherent symmetries. Such SLPNs describe a stochastic language, which describes not only which traces the net supports, but also how likely each trace is. We prove that the class of stochastic languages that can be modelled by SLPN-SDs is strictly larger than the class expressible by SLPNs.

Furthermore, we introduce a discovery algorithm that returns an SLPN-SD based on a given event log and a Petri net. The symmetries studied before are used to not only improve the performance of this algorithm but also to reduce the

complexity of the discovered model without changing the stochastic language it represents. Notably, we ensure the stochastic dependencies included in the final model are indeed significant by including statistical significance testing within the discovery procedure.

The proposed discovery algorithm is implemented in the ProM framework [4] and evaluated twofold on real-life data. First, we qualitatively demonstrate the new types of insights that can be derived using our approach. Second, we quantitatively study the computational feasibility of applying our approach to real-life event logs, and the quality of the discovered SLPN-SDs compared to existing stochastic discovery techniques. The quality of the models is measured using two adapted stochastic conformance checking techniques, as well as using simplicity. We will find that new types of statistically significant insights can be gained that cannot be derived from SLPNs, and that our approach is competitive with existing approaches.

In summary, key contributions of this paper include:

- A new stochastic process modelling formalism (SLPN-SD);
- A study of symmetries in SLPN-SDs;
- A stochastic process discovery technique that discovers SLPN-SDs from an event log and a Petri net;
- Two adapted conformance checking techniques that compare an SLPN-SD to an event log;
- A publicly available implementation of all techniques.

In the remainder of this paper, we discuss related work in Sect. 2 and provide preliminaries in Sect. 3. In Sect. 4 we introduce our new modelling formalism and analyse its symmetries. The discovery method is described in Sect. 5. We evaluate the approach in Sect. 6 and provide a discussion on the approach in Sect. 7; Sect. 8 concludes the paper.

2. Related Work

The representation and discovery of dependencies between past and future behaviour of process executions has been investigated before in several settings, resulting in a variety of approaches with different prerequisites, goals, advantages and limitations.

Hard dependencies, also called long-term, implicit, non-local or indirect dependencies as well as non-local constraints, have been investigated quite extensively and in the context of standard Petri nets they can be expressed using non-free choice constructs. While they are related to our research in that they find and model dependencies between past and future behaviour, it is clear that the correlations we aim to express go beyond that. Hard dependencies can be interpreted as a subset of stochastic dependencies where probabilities are set to either 0 or 1. In contrast, the combination of standard stochastic Petri nets

and hard dependencies could be used to model some stochastic dependencies: certain transitions may enable other transitions which, using non-free-choice constructs, influence the stochastic perspective of later decisions. However, this is unlikely to be readable by human analysts, and does not support loops.

Hard dependencies are supported by several existing discovery algorithms. For instance, heuristic approaches such as Heuristic Miner [5], Fodina [6] and Fusion Miner [7], as well as genetic approaches [8]. By their nature, state-based [9, 10, 11]) or language-based [12] region theory are able to discover hard dependencies. Other algorithms able to mine hard dependencies are certain α-Miner variants [13], eST-Miner [14], approaches based on T-Invariants [15], Maximum Pattern Miner [16], local process mining [17] and AGNEs Miner [18]. Approaches that repair a given Petri net such as [19, 20] may add hard dependencies to existing Petri nets.

Stochastic process discovery techniques include techniques that add a stochastic perspective to an existing model, such as the GDT_SPN Miner [21] and the weight estimators [22]. Toothpaste Miner [23] discovers a stochastic model from scratch. These techniques express the likelihood of behaviour, but do this statically: earlier decisions have no influence on the distribution of later decisions.

The idea of incorporating historical information to adjust the probabilities of future behaviour, particularly based on which activities have been executed and how often, has been a topic in some previous works.

The approach proposed in [3] adds guards which allow an enabled transition to fire only if the current history of the trace satisfies definable requirements. However, these guards represent hard dependencies, i.e., they lack the stochastic component. Furthermore, silent transitions are not supported in [3].

In [24] the authors suggest a similar idea of transition probabilities based on history and [25, 26] apply it in the context of discovery of simulation models. A number of abstractions on history, like property projection, event projection, window abstraction, multiset, set and cardinality abstractions are proposed. The generalized logit model for multinomial response is used to predict the probabilities. However, a limited class of Petri nets is considered there: free-choice and without silent transitions.

Declarative process models, such as Declare [27] can inherently express hard dependencies. A recent extension adds probabilities: a constraint must hold in a certain fraction of behaviour [28], and as such, can express some stochastic dependencies. This is one of the closest approaches to our work, though, unlike [28], our formalism is imperative and can express dependencies on loops.

In the context of process mining, causal analysis aims to derive the influence that decisions or interventions in a process have on other decisions [29], time performance [30, 31] or process outcomes [32, 33] of processes. The causal influence of decisions on later decisions has been studied in [29] for directly follows models and process trees, however focuses on insights for users, considers only these sub-classes of Petri nets, and does not define a modelling formalism for stochastic dependencies. Nevertheless, it is intuitive to combine stochastic dependencies with causal dependencies, and future research may reveal how to interpret causal and associational relations.

In [32, 34] the input format is a so-called *Action-Response-Logs*. Causal relations between pairs of action and response type events and future behaviour are investigated. Our work focuses on traditional event logs and correlation rather than causation, however, the ideas presented in [32] may become relevant for future extensions.

In the context of the discovery algorithm presented in this work, we apply a series of statistical tests to verify the significance of the discovered dependencies. A notable related technique is *lasso* [35], a variable selection and regularization method, which could be applied instead of the significance test to select an interesting subset of dependencies.

3. Preliminaries

In this section, we define existing concepts and introduce notation.

3.1. Multisets

Given a set of elements Σ, a multiset $\mathcal{M}\colon \Sigma \to \mathbb{N}$ maps the elements of Σ to the natural numbers. For instance, $[a^2, b^3]$ is a multiset having two a's and three b's.

Let $\mathcal{M}_1$, $\mathcal{M}_2$ be multisets, then $(\mathcal{M}_1 \uplus \mathcal{M}_2)(e) = \mathcal{M}_1(e) + \mathcal{M}_2(e)$ is the multiset union; $(\mathcal{M}_1 \setminus \mathcal{M}_2)(e) = \max(0, \mathcal{M}_1(e) - \mathcal{M}_2(e))$ is the multiset difference; $\mathcal{M}_1 \subseteq \mathcal{M}_2 \equiv \forall_e \mathcal{M}_1(e) \leq \mathcal{M}_2(e)$; and $|\mathcal{M}_1| = \sum_e \mathcal{M}_1(e)$ is the size of the multiset. Furthermore, by considering the set of elements having a non-zero mapping, the standard set-based relations $\subset$, $\subseteq$, $\setminus$ and $\in$ apply. Let $\mathbb{M}_\Sigma$ denote the set of all multisets over Σ.

3.2. Petri nets

A *labelled Petri net* is a tuple $(P, T, F, \Sigma, \lambda, M_0)$, in which P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, F is a flow relation $F \subseteq (P \times T) \cup (T \times P)$, Σ is a finite alphabet of activities such that $\tau \notin \Sigma$, $\lambda\colon T \to \Sigma \cup \{\tau\}$ is a transition labelling function, $M_0 \in \mathbb{M}_P$ is an initial marking (a multiset of places denoting the state of the net).

For a transition t, let $t^\bullet$ denote the multiset of outgoing places of t: $t^\bullet = [p \mid (t, p) \in F]$. Symmetrically, ${}^\bullet t = [p \mid (p, t) \in F]$.

We assume the default semantics of Petri nets here: the net starts in the initial marking M_0 which indicates *tokens* on places. A transition $t \in T$ is *enabled* in a marking M if ${}^\bullet t \subseteq M$. Let $E(M)$ be the set of enabled transitions in marking M. An enabled transition t can *fire* in a marking M, which brings the system to a new marking $M \uplus t^\bullet \setminus {}^\bullet t$ atomically. On firing, if $\lambda(t) \neq \tau$, then the activity $\lambda(t)$ is emitted. A *path* is a sequence of transitions that brings the net from M_0 to a marking in which no transition is enabled; the corresponding projection using transition labels λ and leaving out transitions mapped to τ is a *trace*. The set of all traces supported by a net is its *language* [36]. Finally, in context of a particular Petri net, let R denote the set of reachable markings from M_0.

3.3. Coverability

Let $(P, T, F, \Sigma, \lambda, M_0)$ be a labelled Petri net. An *ω-marking* is a mapping of places to the natural numbers and ω, which indicates a number that can be arbitrarily large. Using $\omega + 1 = \omega$, $\omega - 1 = \omega$, and $n < \omega$ for all natural numbers n, the enabling and transition firing rules extend to ω-markings, as well as the multiset relations $\trianglelefteq$, $\triangleleft$ and $\in$. A *coverability graph* consists of ω-markings as nodes. For any path such that $M_0 \xrightarrow{t_1} \dots \xrightarrow{t_k} M_k \xrightarrow{t_{k+1}} \dots \xrightarrow{t_n} M_n$ such that $M_k \triangleleft M_n$, the node $\bar{M}_n$ of the coverability graph corresponding to marking M_n is defined as follows: $\forall_{p \in P \wedge M_k(p) < M_n(p)} \bar{M}_n(p) = \omega$, and $\bar{M}_n(p) = M(p)$ for all other places. An edge $M \xrightarrow{t} M'$ for ω-markings M, M' and $t \in T$ is present if $t \in E(M)$ and $M' = M \uplus t^\bullet \setminus {}^\bullet t$. A coverability graph is thus a finite abstract representation of the entire state space of a potentially unbounded Petri net, in which the number of tokens in unbounded places is summarised using ω.

3.4. Stochastic Petri Nets

A *stochastic labelled Petri net* (SLPN) $(P, T, F, \Sigma, \lambda, M_0, w)$ extends a labelled Petri net with a function w that indicates the likelihood that a transition fires. One option to define w is by $w \colon T \to \mathbb{R}^+$. In a given marking M with the set $E(M)$ of enabled transitions, the likelihood that $t \in E(M)$ fires is $\frac{w(t)}{\sum_{t' \in E(M)} w(t')}$ [36]. The probability of a path is the product of the probabilities of all its transitions, and lifted to traces, the weighted set of all traces of an SLPN is its *stochastic language*.

An SLPN has close resemblances with stochastic Petri nets, which express an exponentially distributed time delay on their transitions. In this paper, we only consider the weights of transitions (which equal the parameter of the exponential distribution). A main conceptual difference between stochastic Petri nets and SLPNs is that SLPNs support silent transitions, which do not take time but do have a weight.

We assume that from each marking that is reachable from the initial state with non-zero probability, a final marking (that is, a marking in which no transitions are enabled) can be reached with non-zero probability. That is, we only consider SLPNs without *livelocks*.

4. Stochastic labelled Petri nets with stochastic long-distance dependencies

In this section, we first introduce our variant of SLPNs with stochastic dependencies (SLPN-SD), after which we discuss symmetries in the formalism, discuss loops, and show that it strictly increases expressivity over SLPNs.

4.1. SLPN-SDs

We take SLPNs and modify the definition of their weight function to introduce the dependency of the weight on the execution sequence that resulted in the current marking. Intuitively, these weight functions express for a transition

t a *base weight* Φ_t, multiplied by *weight adjustments* $\varphi_{t,t'}$ for each execution of a transition t'. A multiset abstraction is applied to the execution sequence to define this new weight function.

Definition 1 (Stochastic labelled Petri net with stochastic dependencies)**.** *A* stochastic labelled Petri net with stochastic dependencies *(SLPN-SD) is a tuple* $(P, T, F, \Sigma, \lambda, M_0, w)$*, in which* $(P, T, F, \Sigma, \lambda, M_0)$ *is a labelled Petri net and* $w\colon T \times \mathbb{M}_T \to \mathbb{R}$ *is a weight function of the form* $w(t, \mathcal{M}) = \Phi_t \cdot \prod_{t' \in T} \varphi_{t,t'}^{\mathcal{M}(t')}$ *with* $\Phi_t > 0$ *and* $\varphi_{t,t'} > 0$ *for any* $t, t' \in T$ *and* $\mathcal{M} \in \mathbb{M}_T$*.*

The semantics of SLPN-SDs resembles the semantics of SLPNs [36], the only difference being that one needs to keep track of the executed transitions [3]. Thus, the state of an SLPN-SD is the marking M and a multiset $\mathcal{M}_p$ of executed transitions, i.e. of the path prefix that led to M.

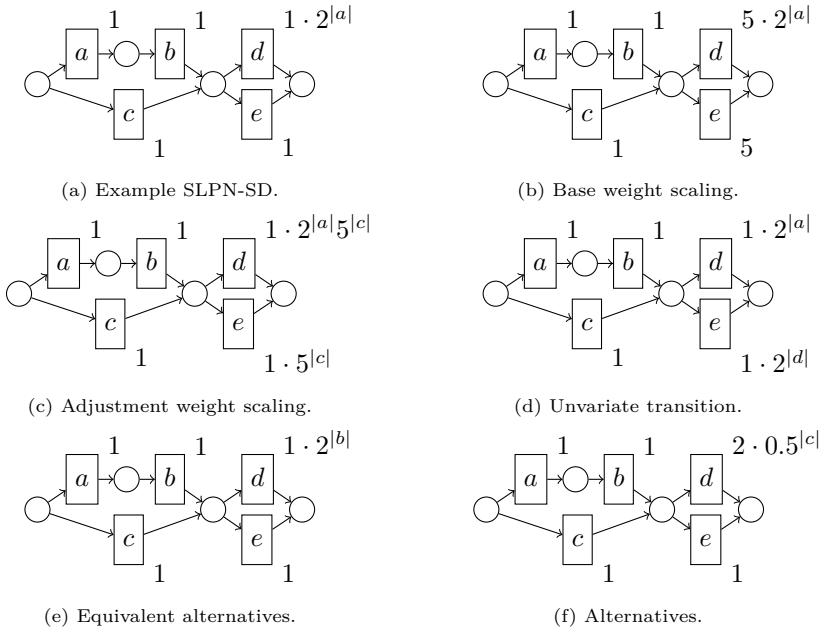
The SLPN-SD starts in the state $(M_0, [])$. In a state $(M, \mathcal{M}_p)$, the probability of firing transition $t \in E(M)$ is $\frac{w(t,\mathcal{M}_p)}{\sum_{t' \in E(M)} w(t',\mathcal{M}_p)}$. Firing t leads to the state $(M \uplus t^\bullet \setminus {}^\bullet t, \mathcal{M}_p \uplus [t])$. Note that M in $(M, \mathcal{M}_p)$ reached from $(M_0, [])$ is uniquely defined by $\mathcal{M}_p$.

Fig. 2a shows an example; parameters equal to 1 may be omitted. The transition d has a stochastic dependency on transition a: the base weight Φ_d is 1, while the weight adjustment $\varphi_{d,a}$ is 2 and the other weight adjustments equal 1. We write, e.g., $2^{|a|}$ at transition d to indicate that $\varphi_{d,a} = 2$. That is, the weight of d is 1 if d got enabled after executing transition c, but it equals 2 if transitions a and b were executed first. Formally, the weight function w of transition d is $w(d, [a^{|a|}, b^{|b|}, c^{|c|}, d^{|d|}, e^{|e|}]) = 1 \cdot 2^{|a|} \cdot 1^{|b|} \cdot 1^{|c|} \cdot 1^{|d|} \cdot 1^{|e|}$. The stochastic language of this net is $[\langle a, b, d\rangle^{\frac{1}{3}}, \langle a, b, e\rangle^{\frac{1}{6}}, \langle c, d\rangle^{\frac{1}{4}}, \langle c, e\rangle^{\frac{1}{4}}]$.

4.2. Symmetries

SLPN-SDs with equivalent control-flow structures may still have equivalent stochastic languages even if they have certain syntactical differences in their stochastic dependencies. Our aim is to discover SLPN-SDs that enable better process comprehension. Including less dependencies in the resulting SLPN-SD improves model simplicity, and hence model comprehension. Therefore, we are interested in reducing the number of dependencies. To achieve that, we consider models with control flow patterns that yield symmetries in stochastic behaviour and we steer the discovery process to a simpler model with equivalent stochastic behaviour. In this section, we study control flow constructs for which base weights and weight adjustments can be modified without changing the stochastic behaviour to make use of them in model discovery in Section 5.

We describe several such symmetries, using the concept of *groups of transitions*: any two transitions t and t' belong to the same group iff there is a reachable state such that t and t' are both enabled. Formally speaking, a set $G \subseteq T$ of transitions is called a *group* if it is a minimal nonempty set such that for every reachable marking M, $E(M) \subseteq G$ or $E(M) \cap G = \emptyset$. Two groups are either disjoint or they are equal, implying that groups form a partitioning of the



(a) Example SLPN-SD.

(b) Base weight scaling.

(c) Adjustment weight scaling.

(d) Unvariate transition.

(e) Equivalent alternatives.

(f) Alternatives.

Figure 2: Stochastic-language equivalent SLPN-SDs.

set T of transitions. In our example SLPN-SD Fig. 2a, the groups are $\{a, c\}$, $\{b\}$ and $\{d, e\}$. We do not aim to, and we cannot, be exhaustive in symmetries, as these may arise from the model's structure.

Symmetry S1 The base weight Φ_t of all transitions $t \in G$ in a group G can be scaled by a positive number f. This does not change the stochastic behaviour, as for any state $(M, \mathcal{M})$, the probability of firing a transition $u \in G$ does not change: If $E(M) \subseteq G$ then $\frac{f\,\Phi_u(\varphi_{u,t})^{\mathcal{M}_p(t)}}{\sum_{v\in E(M)}(f\,\Phi_v(\varphi_{v,t})^{\mathcal{M}_p(t)})} = \frac{f\,\Phi_u\,\varphi_{u,t}^{\mathcal{M}_p(t)}}{f\sum_{v\in E(M)}(\Phi_v\,\varphi_{v,t}^{\mathcal{M}_p(t)})} = \frac{\Phi_u\,\varphi_{u,t}^{\mathcal{M}_p(t)}}{\sum_{v\in E(M)}(\Phi_v\,\varphi_{v,t}^{\mathcal{M}_p(t)})}$, otherwise $E(M) \cap G = \emptyset$ and no $u \in G$ is enabled. Fig. 2b shows an example: $\{d, e\}$ is a group of transitions and the base weights of d and e have been scaled by 5.

Symmetry S2 Given an arbitrary transition t, a group G and a positive number f, the adjustment weight $\varphi_{u,t}$ of all transitions $u \in G$ can be scaled by f without changing the stochastic behaviour of the model, as for any reachable state $(M, \mathcal{M}_p)$ and a transition $u \in E(M) \cap G$, the probability to fire u is $\frac{\Phi_u(f\,\varphi_{u,t})^{\mathcal{M}_p(t)}}{\sum_{v\in E(M)}\Phi_v\cdot(f\,\varphi_{v,t})^{\mathcal{M}_p(t)}} = \frac{\Phi_u\,f^{\mathcal{M}_p(t)}\,\varphi_{u,t}^{\mathcal{M}_p(t)}}{f^{\mathcal{M}_p(t)}\sum_{v\in E(M)}\Phi_v\,\varphi_{v,t}^{\mathcal{M}_p(t)}} = \frac{\Phi_u\,\varphi_{u,t}^{\mathcal{M}_p(t)}}{\sum_{v\in E(M)}\Phi_v\,\varphi_{v,t}^{\mathcal{M}_p(t)}}$. Fig. 2c shows an example: $\varphi_{d,c}$ and $\varphi_{e,c}$ have both been scaled with 5.

Symmetry S3 Consider two transitions t, t'. If there is a number c such that before every enablement of t, t' is executed precisely c times, a symmetry is

(a) Negative feedback loop.

(b) Positive feedback loop.

Figure 3: Examples of (different) SLPN-SDs with loops.

present. Formally, given $t, t' \in T$, if $\exists_{c \in \mathbb{N}}$ such that for every reachable state $(M, \mathcal{M}_p)$, $t \in E(M) \Rightarrow \mathcal{M}_p(t') = c$, then changing Φ_t and $\varphi_{t,t'}$ to Φ'_t and $\varphi'_{t,t'}$ such that $\Phi_t(\varphi_{t,t'})^c = \Phi'_t(\varphi'_{t,t'})^c$ does not change the stochastic behaviour of the model. Fig. 2d shows an example: $\varphi_{e,d}$ is 2, but as d is never executed before e (that is, c of our formalisation is 0), $\varphi_{e,d}$ has no influence.

Symmetry S4 Given transitions t, u, v, if u and v are executed equally many times in any reachable marking enabling t, then the adjustment weights $\varphi_{t,u}$ and $\varphi_{t,v}$ are interchangeable. Formally, given transitions t, u, v, if $\mathcal{M}_p(u) = \mathcal{M}_p(v)$ for any reachable state $(M, \mathcal{M}_p)$ such that $t \in E(M)$, then changing $\varphi_{t,u}, \varphi_{t,v}$ to $\varphi'_{t,u}, \varphi'_{t,v}$ such that $\varphi_{t,u}\,\varphi_{t,v} = \varphi'_{t,u}\,\varphi'_{t,v}$ does not change the stochastic behaviour of the model. Fig. 2e shows an example: d has a dependency on a, but obviously this can be (even partially) shifted to a dependency on b.

Symmetry S5 Consider a transition t and a set of transitions X such that $\sum_{x \in X} \mathcal{M}_p(x) = 1$ for any reachable state $(M, \mathcal{M}_p)$ in which $t \in E(M)$, i.e. exactly one of the transitions from X must be executed once before executing t. Then we can scale Φ_t by an arbitrary positive number f, while scaling $\varphi_{t,x}$ by $\frac{1}{f}$ for every $x \in X$, without changing the stochastic behaviour of the model. Thus $\Phi'_t = f\,\Phi_t$ and $\varphi'_{t,x} = \frac{\varphi_{t,x}}{f}$ for all $x \in X$, and for a state $(M, \mathcal{M}_p)$ with $\mathcal{M}_p(u) = 1$ for some $u \in X$ and $\mathcal{M}_p(x) = 0$ for all $x \in X \setminus \{u\}$ we have:

$$\Phi'_t \prod_{x \in X} (\varphi'_{t,x})^{\mathcal{M}_p(x)} = f\,\Phi_t\,\frac{\varphi_{t,u}}{f} \prod_{x \in X \setminus \{u\}} (\varphi'_{t,x})^0 = \Phi_t \prod_{x \in X} \varphi_{t,x}^{\mathcal{M}_p(x)}$$

In particular, by taking $f = \varphi_{t,u}$ for some $u \in X$, we can replace Φ_t with $\Phi_t\,\varphi_{t,u}$, and $\varphi_{t,x}$ with $\frac{\varphi_{t,x}}{\varphi_{t,u}}$ for every $x \in X$, thus obtaining $\varphi_{t,u} = 1$.

Fig. 2f shows an example where this symmetry is applied: to reach a state enabling d, either a or c needs to be executed exactly once. Therefore, we can change $\varphi_{d,a}$ from 2 to 1 (dividing by 2), multiply Φ_d by 2 and divide $\varphi_{d,c}$ by 2, thus changing it from 1 to 0.5.

4.3. Loops

Our definition of SLPN-SDs allows for stochastic dependencies in loops: transitions may depend on transitions that form a loop with them. Fig. 3 shows two examples. In each SLPN-SD, the initial weight of b is 1, and for every execution of b, its weight decreases by half (Fig. 3a) or increases twofold

(Fig. 3b). Thus, these SLPN-SDs model negative resp. positive feedback loops: each time the loop is taken, the probability to execute b another time changes.

Intuitively, stochastic dependencies in loops can be used to model rework in business processes: we would argue that it is highly *un*likely that the probability of performing more rework would be *in*dependent of previously performed rework. For instance, if exiting a loop reflects passing a quality control check, then it would be highly unlikely that performing a repair has no influence whatsoever on the quality control afterwards (and thus on the probability of exiting the loop after rework).

The stochastic language of the SLPN-SD with a positive feedback loop in Fig. 3b is

$$[\langle a\rangle^{\frac{1}{1}\cdot\frac{1}{2}}, \langle a,b\rangle^{\frac{1}{1}\cdot\frac{1}{2}\cdot\frac{1}{3}}, \langle a,b,b\rangle^{\frac{1}{1}\cdot\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{1}{5}}, \ldots] \text{ or } [\langle a,b^n\rangle^{\frac{1}{2^n+1}\cdot\prod_{k=1}^{n}\frac{2^k}{2^k+1}} \mid n\in\mathbb{N}]$$

The probability to exit the loop decreases exponentially with the number of the executions of b. Theoretically, it never becomes 0, however, it becomes extremely unlikely to finish in practice. This model may challenge simulation tools that randomly find a path through the model by weighted chance until an end state is reached.

4.4. Expressiveness

Next, we prove that SLPN-SD is a proper extension of SLPN. That is, the stochastic language of every SLPN can be expressed by a SLPN-SD, but there are SLPN-SDs whose stochastic language cannot be expressed by an SLPN.

Theorem 1 (SLPN $\rightarrow$ SLPN-SD)**.** *For every SLPN, there is an SLPN-SD with the same stochastic language.*

Proof. We transform the SLPN with weight function w into a SLPN-SD by defining $\Phi_t = w(t)$ and $\varphi_{t,t'} = 1$ for any $t, t' \in T$. □

Theorem 2 (SLPN-SD $\not\rightarrow$ SLPN)**.** *There are SLPN-SDs for which there is no SLPN with the same stochastic language.*

Proof. There is no SLPN with the same stochastic language as our SLPN-SD in Fig. 3. The weight of b changes to a new, larger value after each execution of b. Thus, one needs to model an infinite number of weights for transition b. Infinitely many copies of transition b would be needed in an SLPN to mimic this behaviour, since every transition has a fixed weight. As an SLPN has a finite set of transitions, there cannot be such an SLPN. □

5. Discovery

To discover an SLPN-SD from an event log and a Petri net, we perform the steps illustrated in Figure 4: we first align the log to obtain the sequences of transitions that were most likely executed for the log. Second, we reconstruct the choices made in the traces. Third, we select which parameters are necessary

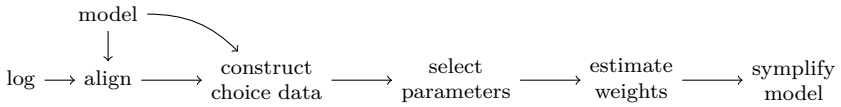


Figure 4: Discovery approach.

to express the weight functions in the SLPN-SD. Fourth, we estimate the weight function. Fifth, we simplify the SLPN-SD.

In this section, we will use a running example consisting of a log with 300 traces $[\langle a,b,d\rangle^{99}, \langle a,b,e\rangle^{50}, \langle c,d\rangle^{75}, \langle c,e\rangle^{75}, \langle a,b,d,d\rangle^{1}]$, and the Petri net in Fig. 2a (assuming we do not know the weight parameters).

5.1. Computing Alignments

We apply alignments [37] to the log and model, which will find a least-edits way to replay each trace from the log on the model. Thus, for each log trace, alignments find a corresponding path through the model. In the following, we will use these paths: sequences of transitions. For our running example, we obtain $[\langle a,b,d\rangle^{100}, \langle a,b,e\rangle^{50}, \langle c,d\rangle^{75}, \langle c,e\rangle^{75}]$. Please, note that these are paths rather than traces: the steps are transitions of the model and not their labels.

5.2. Constructing Choice Data

The second step is to reconstruct the choices made in the model. To this end, we construct all states reached when executing the log, using prefixes of all paths resulting from the alignments computation; we abstract the prefix to a multiset $\mathcal{M}_p$ of transitions. Due to the marking equation, $\mathcal{M}_p$ uniquely defines the marking M of the state $(M, \mathcal{M}_p)$. For each $\mathcal{M}_p$ present in the alignments, we record $E = E(M)$ of the corresponding marking M as well as the multiset $\mathcal{M}_n$ of transitions executed in the log immediately after this prefix. For our running example, the choice data is shown in Table 1a.

Formally, we start with the empty choice data $\mathrm{C} = \emptyset$, go along every alignment and update the choice data as follows: for each prefix p followed by the execution of a transition t, we consider the multiset abstraction $\mathcal{M}_p$ of p with the corresponding set of enabled transitions E, and add $(\mathcal{M}_p, E, [t])$ to C if $(\mathcal{M}_p, E, \mathcal{M}_n)$ was not present in C for any $\mathcal{M}_n$, otherwise we update the value of this $\mathcal{M}_n$ to $\mathcal{M}_n \uplus [t]$.

5.3. Selecting parameters

In an SLPN-SD, for every transition t there is a base weight parameter Φ_t and for every pair of transitions t, t' there is an adjustment weight parameter $\varphi_{t',t}$. As the aim is to discover SLPN-SDs, all of this quadratic number of parameters needs to be assigned a value. To limit computational complexity (for the discovery algorithm) and model complexity (for human analysts), it is beneficial to reduce the number of parameters that need to be considered. In this section, we describe several strategies that reduce the number of parameters, without limiting the expressive power of the SLPN-SD's stochastic perspective.

prefix $\mathcal{M}_p$	enabled E	next transition $\mathcal{M}_n$
$[]$	$\{a, c\}$	$[a^{150}, c^{150}]$
$[a]$	$\{b\}$	$[b^{150}]$
$[c]$	$\{d, e\}$	$[d^{75}, e^{75}]$
$[a, b]$	$\{d, e\}$	$[d^{100}, e^{50}]$

(a) Choice data.

$.$	$\Phi_.$	$\varphi_{.,a}$	$\varphi_{.,b}$	$\varphi_{.,c}$	$\varphi_{.,d}$	$\varphi_{.,e}$
a	R1	R1	R1	R1	R1	R1
b	R1	R1	R1	R1	R1	R1
c	-	R2	R2	R2	R2	R2
d	R1	R1	R1	R1	R1	R1
e	-	-	R3	-	R2	R2

(b) Parameters, and which reduction removes it.

a executed:	0	1
e chosen	75	50
e not chosen	75	100

c executed:	0	1
e chosen	50	75
e not chosen	100	75

(c) χ^2 observations.

$\mathcal{M}_p$	$\mathcal{M}_n$	t	equality
$[]$	$[a^{150}, c^{150}]$	a	$\frac{150}{300} = \frac{\Phi_a}{\Phi_a + \Phi_c}$
		c	$\frac{150}{300} = \frac{\Phi_c}{\Phi_a + \Phi_c}$
$[a]$	$[b^{150}]$	b	$\frac{150}{150} = \frac{\Phi_b}{\Phi_b}$
$[c]$	$[d^{75}, e^{75}]$	d	$\frac{75}{150} = \frac{\Phi_d \cdot \varphi_{d,c}}{\Phi_d \cdot \varphi_{d,c} + \Phi_e \cdot \varphi_{e,c}}$
		e	$\frac{75}{150} = \frac{\Phi_e \cdot \varphi_{e,c}}{\Phi_d \cdot \varphi_{d,c} + \Phi_e \cdot \varphi_{e,c}}$
$[a, b]$	$[d^{100}, e^{50}]$	d	$\frac{100}{150} = \frac{\Phi_d \cdot \varphi_{d,a} \cdot \varphi_{d,b}}{\Phi_d \cdot \varphi_{d,a} \cdot \varphi_{d,b} + \Phi_e \cdot \varphi_{e,a} \cdot \varphi_{e,b}}$
		e	$\frac{50}{150} = \frac{\Phi_e \cdot \varphi_{e,a} \cdot \varphi_{e,b}}{\Phi_d \cdot \varphi_{d,a} \cdot \varphi_{d,b} + \Phi_e \cdot \varphi_{e,a} \cdot \varphi_{e,b}}$

(d) Equations; removed by R5; fixed parameters.

Table 1: Discovery steps for our running example.

Technically, when we 'remove' a parameter, we replace it with a constant 1. For an adjustment parameter $\varphi_{t,t'} = 1$, this means that there is no dependency modelled as t' has no influence on t. Consequently, we do not need to show these parameters to the user. For a base weight parameter $\Phi_t = 1$, we argue that this makes it easier to interpret the base weight parameters of *other* transitions: the user would not need to perform mental multiplications to determine the weight.

In the following, we describe several reductions for parameters, in which C denotes the choice data.

5.3.1. Groups

In a typical Petri net, there might be transitions that are never enabled together, thus belonging to different groups as described in Sect. 4.2. For each group of transitions, a weight function can be defined in isolation.

To determine whether two transitions t, t' may be enabled at the same time, several options apply:

1. From the model, t and t' are independent if they belong to two different groups of transitions, as defined in Sect. 4.2. Groups of transitions can be computed on an adjusted coverability graph, where we change the condition on introducing a new node M to the graph: normally, it is the existence of M' in the graph already such that $M' \trianglelefteq M$. Instead, we also require that there exists $M \trianglelefteq M$ such that $E(M') = E(M)$. For (bounded or unbounded) nets without arc weights, this ensures that we see every E.

2. From the choice data C, $\exists_{(\mathcal{M}_p,E,\mathcal{M}_n)\in \mathrm{C}}(t \in E \wedge t' \in E)$ is a sufficient condition to conclude that t and t' are not independent. If we assume that the choice data C contains all sets E at least once, then it is a necessary condition as well. In safe nets, the assumption can be weakened by taking ${}^\bullet t \cap {}^\bullet t' = \emptyset$ as a necessary condition.

For our running example, the groups $\{a, c\}$, $\{b\}$ and $\{d, e\}$ can be identified and considered in isolation.

5.3.2. Symmetries

We can leverage some symmetries described in Sect. 4.2 to further reduce the number of parameters, and thus limit the search space of weight function discovery.

Reduction R1 Using symmetries S1 and S2, from each group G we may select one arbitrary transition t and fix all its parameters:

$$\text{for one } t \in G \tag{1}$$
$$\text{do } \Phi_t := 1 \wedge \forall_{t'} \varphi_{t,t'} := 1 \tag{2}$$

In our example, this reduction allows us to fix the parameters indicated by R1, shown in Table 1b.

Reduction R2 If in any prefix $\mathcal{M}_p$ in which transition t' is enabled, a transition t is executed precisely a fixed number of times, then by Symmetry S3 we can fix $\varphi_{t,t'}$:

$$\text{for all } t, t' \tag{3}$$
$$\text{such that } \exists_{c\in\mathbb{N}} \forall_{(\mathcal{M}_p,E)}(t' \in E \Rightarrow \mathcal{M}_p(t) = c) \tag{4}$$
$$\text{do } \varphi_{t,t'} := 1 \tag{5}$$

To determine the domain of $(\mathcal{M}_p, E)$, two options apply:

(a) We can check the property by constructing the coverability graph of the model. For each node in this graph, we keep the number of times t was executed in reaching that node. If we add an edge in the coverability graph, we check whether the two to-be connected nodes have the same number of ts annotated (or, the target has one more if the edge denotes the execution of t). If this makes the node inconsistent, we mark it as such and propagate the inconsistency. Whenever in a node t' is enabled, the number of times t was executed should be consistent, and equal to c.

(b) From the choice data, we can consider all encountered $(\mathcal{M}_p, E, \mathcal{M}_n) \in \mathrm{C}$. This assumes that enough tuples $(\mathcal{M}_p, E)$ have been observed to derive c and conclude that at enablement of t', t is always executed c times.

In our example, this reduction allows us to fix the parameters indicated by R2, shown in Table 1b.

Reduction R3 If for every execution of a transition t, we observe that two transitions u, v are always executed the same number of times, and this number is always either 0 or 1, then by Symmetry S4 we can fix either $\varphi_{t,u}$ or $\varphi_{t,v}$:

$$\text{for all } t, \{u, v\} \tag{6}$$

$$\text{such that } \forall_{(\mathcal{M}_p,E)} t \in E \Rightarrow \mathcal{M}_p(u) = \mathcal{M}_p(v) \wedge \mathcal{M}_p(u) \in \{0, 1\} \tag{7}$$

$$\text{do } \varphi_{t,u} := 1 \tag{8}$$

While Symmetry S4 targets any number of times, as long as u and v are executed the same number of times. If this number is 0 or 1, then $\varphi_{t,v}$ could be fixed by any other reduction without issues. Thus, by limiting this reduction to the 0 or 1 case, we can apply it independently of the other reductions.

To determine the domain of $(\mathcal{M}_p, E)$, two options apply:

(a) From the model, we can obtain the property by constructing a coverability graph. In this graph, we keep an extra variable in each node indicating the difference between the number of executions of u and v. If an edge is added to an existing node, the annotated differences are compared; if they are inconsistent, the node is marked as such and the inconsistency is propagated. In every coverability node in which t is enabled, this difference must be consistent and equal to 0.

(b) From the choice data, we can consider all encountered $(\mathcal{M}_p, E, \mathcal{M}_n) \in \mathrm{C}$ and verify the property directly. This assumes that enough tuples $(\mathcal{M}_p, E)$ have been observed to derive the property.

In our example, this reduction allows us to fix the parameters indicated by R3, shown in Table 1b.

The symmetries relevant for these reductions show that the parameter that gets fixed is redundant in respect to some other parameter. Then, it is not a

problem if the other parameter is fixed as well by another reduction, as any behaviour related to the fixed parameter can be fully expressed by using the other parameter. Therefore, these reductions can be arbitrarily combined, though not repeated. In future work, it may even be a good idea to reduce the search for parameters to fix based on the parameters that are already fixed.

5.3.3. Significance

In order to ensure that all reported dependencies are significant, we apply a set of statistical tests.

An adjustment weight parameter $\varphi_{t,t'}$ indicates an association between transitions t and t'. Let X be a random variable indicating whether t was executed from a state enabling t, and let Y be a random variable indicated the number of times t' was executed along the path leading to this state. To verify that X and Y are indeed associated, we perform a statistical test, with null-hypothesis that X and Y are independent.

Each enabling of t in the choice data constitutes one observation for X and Y. Formally, the number of times that $X = \mathit{true}, Y = i$ is

$$\sum_{(\mathcal{M}_p,E,\mathcal{M}_n)\in C \wedge t\in E \wedge \mathcal{M}_p(t')=i} \mathcal{M}_n(t)$$

The number of times that $X = \mathit{false}, Y = i$ is

$$\sum_{(\mathcal{M}_p,E,\mathcal{M}_n)\in C \wedge t\in E \wedge \mathcal{M}_p(t')=i} |\,\mathcal{M}_n\,| - \mathcal{M}_n(t)$$

If for any $Y = i$, either $X = \mathit{true}$ or $X = \mathit{false}$ has less than 5 observations, that $Y = i$ is removed; if this leaves less than two $Y = i$, then the parameter $\varphi_{t,t'}$ is fixed, due to a lack of data.

On these observations, we perform a χ^2 test [38] with a user-chosen threshold α, divided by the total number of tests according to the Bonferroni correction [39].

Reduction R4 If the statistical test rejects the null-hypothesis, then there is sufficient evidence of the alternative hypothesis, which states that X and Y are dependent and thus that t and t' are associated. If not, we fix $\varphi_{t,t'}$.

For our running example, the observations are summarised in Table 1c. For both, the χ^2 value is 8.57; the p-value 0.003. For our adjusted α of $0.05/2 = 0.025$, both are significant. Hence, both a and c have a statistically significant impact on e and thus the $\varphi_{e,a}$ and $\varphi_{e,c}$ adjustment weight parameters should not be fixed.

5.4. Estimating the weight function

The next step is to use the remaining parameters to estimate the weight function of the SLPN-SD. We start from the choice data C: for each triple $(\mathcal{M}_p, E, \mathcal{M}_n)$, we add an equality that states that the likelihood of a transition

seen in $\mathcal{M}_n$ should match the weight as posed by the parameters of the weight function:

$$\forall_{(\mathcal{M}_p,E,\mathcal{M}_n)\in C}\forall_{t\in E}\frac{\mathcal{M}_n(t)}{|\,\mathcal{M}_n\,|} = \frac{\Phi_t \cdot \prod_{t'\in\mathcal{M}_p} \varphi_{t,t'}^{\mathcal{M}_p(t')}}{\sum_{t''\in\mathcal{M}_n} \Phi_{t''} \cdot \prod_{t'\in\mathcal{M}_p} \varphi_{t'',t'}^{\mathcal{M}_p(t')}} \tag{9}$$

The weight function is then given by a solution for which the equations hold as closely as possible. In determining this closeness, the size $|\,\mathcal{M}_n\,|$ is considered. This is a non-linear optimisation problem, for which standard solvers are available to approximate an optimal solution.

In (9), we use the information of the model of which transitions were enabled ($t \in E$). However, we obtain the prefixes ($\mathcal{M}_p$) from the choice data. These prefixes could be derived from the model using a state space exploration, however we cannot obtain the number of times each enabled transition was executed ($\mathcal{M}_p$) from a control-flow model: there is no such information available if the corresponding prefix was not observed in the log.

As a further reduction, we can observe that not all equations are necessary:

Reduction R5 The equations come in conjugated sets: for each $(\mathcal{M}_p, E, \mathcal{M}_n)$ in the choice data C, the equation for one arbitrary transition $t \in \mathcal{M}_n$ adds no further information and can be removed.

For our running example, the equations are shown in Table 1d; dropped equations and fixed parameters are indicated. Simplified, these equations are: $\frac{1}{2} = \frac{1}{1+\Phi_c}$; $\frac{1}{2} = \frac{1}{1+\Phi_e\cdot\varphi_{e,c}}$; and $\frac{2}{3} = \frac{1}{1+\Phi_e\cdot\varphi_{e,a}}$. An optimal solution is $\Phi_c = 1$, $\Phi_e = \frac{1}{3}$, $\varphi_{e,c} = 3$ and $\varphi_{e,a} = 1.5$.

5.5. Heuristic Model Simplification

Some of the symmetries identified in Section 4.2 are challenging to leverage before optimisation – in interplay with other symmetries, too many parameters fixed means that the SLPN-SD loses expressivity –, but can be used *afterwards* in post-processing reductions. The final reductions are applied to change the SLPN-SD without changing its behaviour; that is, each reduction transforms a set of parameters, in order to make as many parameters 1 as possible.

Reduction R6 Symmetry S5 can be used as follows: if before every enablement of t, precisely one of a set of transitions X is executed (11), then we can scale on weight adjustment parameter φ to 1, amongst scaling several other parameters. As the other affected parameters are scaled as well, we only apply this reduction if these are not 1 already (12).

$$\text{for all } t, X \tag{10}$$

$$\text{such that } \forall_{(\mathcal{M}_p,\mathcal{M}_n)\in C}\, \mathcal{M}_n(t) > 0 \Rightarrow \textstyle\sum_{t'\in X} \mathcal{M}_p(t') = 1 \tag{11}$$

$$\wedge\, \forall_{t'\in X}\, \varphi_{t,t'} \neq 1 \tag{12}$$

$$\text{pick a } v \in X \tag{13}$$

$$\text{and do } \forall_{u\in X, u\neq v}\, \varphi_{t,u} := {}^{\varphi_{t,u}}\!/_{\varphi_{t,v}} \wedge \Phi_t := \Phi_t\, \varphi_{t,v} \wedge \varphi_{t,v} := 1 \tag{14}$$

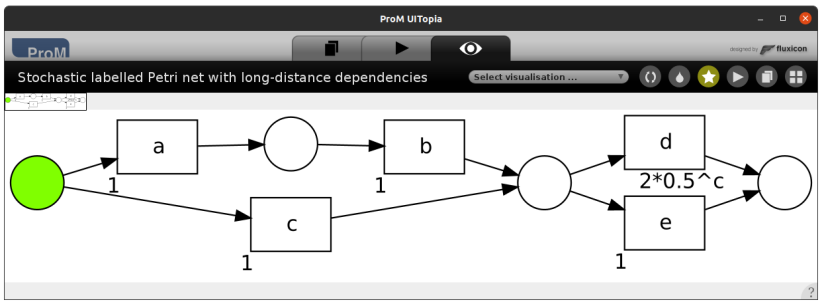


Figure 5: Screenshot of an SLPN-SD in the ProM framework.

For our running example, we apply R6 as follows: $t = e$, $X = \{a, c\}$. Then, we pick $v = c$ and end up with $\Phi_c = 1$, $\Phi_e = 1$, $\varphi_{e,c} = 1$ and $\varphi_{e,a} = \frac{1}{2}$.

6. Evaluation

In this section, we evaluate our approach threefold: we show its computational feasibility using an implementation, illustrate the new insights it may provide, and evaluate the practical applicability of the technique and the quality of the models with respect to other stochastic discovery techniques.

6.1. Implementation

SLPN-SDs, a discovery technique, a conformance checking technique, two visualisations, a file importer and a file exporter have been implemented as plug-ins of the ProM framework [4] in the `LongDistanceDependencies` package. The source code is available at `http://svn.win.tue.nl/repos/prom/Packages/LongDistanceDependencies/Trunk`.

Figures 5 and 6 show screenshots of the two visualisations. The first visualisation shows the weight parameters in full detail. The second visualisation abstracts from the precise values and only shows the base weights and the type of dependency: an incoming red arc indicates a lowering (< 1) adjustment weight, while an incoming green arc indicates an increasing (> 1) adjustment weight.

In our implementation of the discovery method, we use the choice data for the symmetries: no coverability graph computations are performed – these remain future work. The optimisation is likely non-convex, and therefore we can only approximate an optimal solution, using the Levenberg-Marquardt method. This method does not support constraints to require parameters to be positive, so on encountering of a negative value we truncate to 0. We apply the optimisation in two passes: first, only the base weight Φ parameters are estimated with an initialisation of 1. Second, the base weight Φ and adjustment weight φ parameters are estimated, initialised with the found Φ parameters. Intuitively, this will first find the "average" values of Φ in a smaller search space, after which the φ are fine-tuned to include the dependencies.

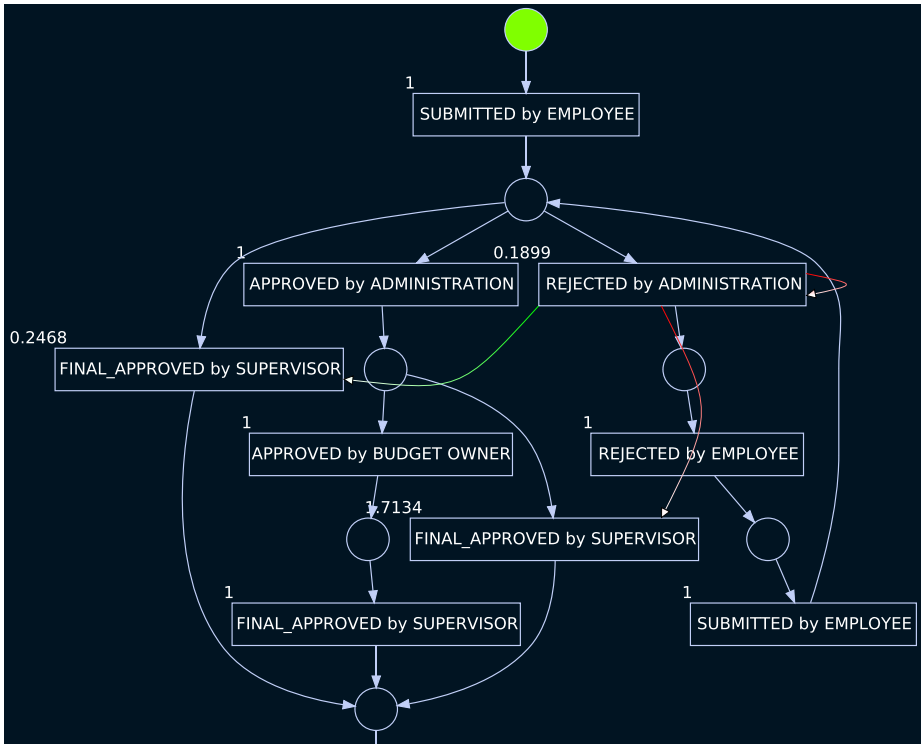


Figure 6: Part of SLPN-SD of the BPIC20 Request for Payment event log (activity labels abbreviated).

6.2. Insights

To illustrate the practical insights that can be obtained using stochastic dependencies, we applied our discovery technique to an event log (the BPIC20-Request for Payment log), and a control-flow model obtained by applying the Directly Follows Model Miner [40] to this log. The discovered model was converted to a Petri net and our stochastic dependencies discovery technique was applied.

Fig. 6 shows the resulting SLPN-SD; activities have been abbreviated. The control-flow model by itself – in blue – shows the possible paths in the model. In this perspective, we can identify a rework loop of rejection by the administration. Adding the stochastic perspective on top – the base weights in white – yields insights into the likelihoods of behaviour. The stochastic perspective shows, for instance, that after submission by an employee, the likelihood of approval by the supervisor is $\frac{0.2468}{0.2468+1+0.1899} = 17\%$.

The approach of this paper adds stochastic dependencies *on top* of the stochastic perspective, thereby making the stochastic behaviour dynamic. In Fig. 6, a rejection by the administration correlates with several changes in probabilities further on in the process. That is, every time that a request gets rejected

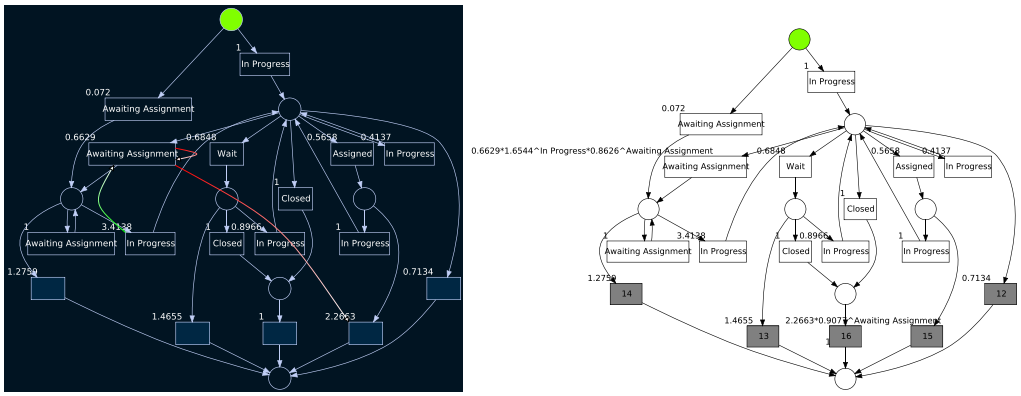


Figure 7: SLPN-SD of of the BPIC2013 - open problems event log.

by the administration: (1) another rejection by the administration becomes less likely (indicated by the red arc to itself), and (2) approval by a supervisor (instead of by the administration) becomes more likely (indicated by the green arc to final approval by supervisor). At another decision point in the model, after approval by administration, approval by the supervisor is less likely if the payment request was first rejected by the administration. An explanation, and a potential insight, would be that in case of a rejection by the administration, supervisors escalate to the budget owner before taking a decision themselves. This insight is not only statistically significant, but quantified as well in the alternative visualisation (not shown): the weight of approval by a supervisor gets multiplied by 0.74 for each rejection by the administration: without preceding rejection 63% is approved by the supervisor, with 1 preceding rejection this is 56%. In order to obtain this insight, it is necessary to not only consider the control flow perspective and the stochastic perspective, but also stochastic dependencies, as provided by our approach.

A second example is given in Figure 7. We take a log extracted from a problem management system of Volvo IT Belgium, which shows the lifecycle of reported IT incidents [41]. The control-flow model was discovered using the Directly Follows Model Miner, and translated to a Petri net; notice that the model contains many duplicated transitions in its Petri net translation. We applied our approach to this log and the Petri net; the results are visualised in Figure 7.

An interesting part of this SLPN-SD is the loop of `Awaiting Assignment` and `In Progress`: combined, these stochastic dependencies form a positive feedback loop of factor 1.654 * 0.863 = 1.427. A potential insight that could be derived from these stochastic dependencies is that problems that are 'shifted around' a lot do not get solved even after being assigned but instead go back to being 'shifted around': the more shifted around, the more likely to be even more shifted around.

This example also shows that positive feedback loops are not necessarily

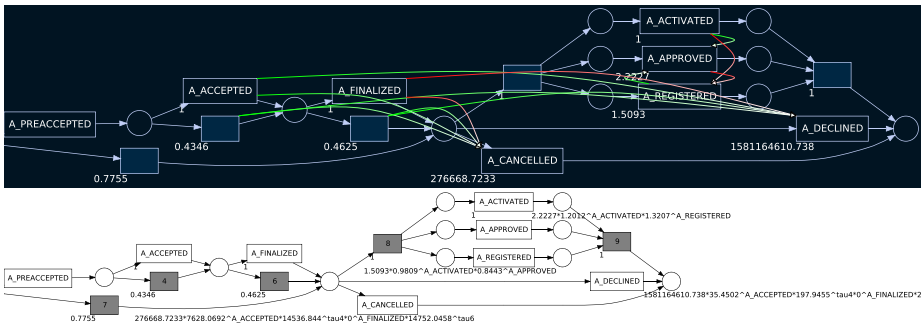


Figure 8: Part of SLPN-SD of the BPIC2012 - a log.

an inherent problem to SLPN-SD: even though `Awaiting Assignment` has a positive feedback loop, as part of the same loop `In Progress` does not become more likely, thus the exit probability (silent transition 14) does not get more unlikely with each execution of the loop.

A third example is shown in Figure 8. This log describes a loan application process, and the control flow model indicates several control-flow dependencies, such as that `A_ACCEPTED` can only be executed if `A_PREACCEPTED` was executed. From the SLPN-SD, we can observe that the decision to execute `A_ACCEPTED` and to execute `A_FINALIZED` have no significant stochastic dependency, as no green or red arc is drawn between these transitions. The main decision is the last one in the process: `A_DECLINED` and `A_CANCELLED` have *high* base weights, which make executing the parallel block of `A_ACTIVATED`, `A_APPROVED` and `A_REGISTERED` (with weight 1) extremely unlikely. However, both `A_DECLINED` and `A_CANCELLED` have a strong negative stochastic dependency: if `A_FINALIZED` is executed, then both options are multiplied by very low adjustment weight parameters ($3 \cdot 10^{-10}$ and $9 \cdot 10^{-12}$). This is a clear dependency, and where the control flow model does not show it, the stochastic dependencies of SLPN-SD do.

A further set of stochastic dependencies can be found in the parallel block: these activities are automatic, and even though they appear in all possible orders in the log, not only can weights be found – `A_APPROVED` is most likely to happen first –, but the likelihood of the second to-be executed activity changes with the first one.

These illustrations showed the types of insights that can be derived from SLPN-SDs, which are not derivable from the control flow or the stochastic perspective: SLPN-SDs show dynamic *changes* in the stochastic perspective.

6.3. Model quality

In this experiment, we evaluate whether the SLPN-SDs of this paper can compare with the SLPNs discovered by existing stochastic discovery techniques on stochastic model quality. Fig. 9 shows the experimental set-up, where Tab. 2 shows further details. We start with a set of real-life logs, from which we

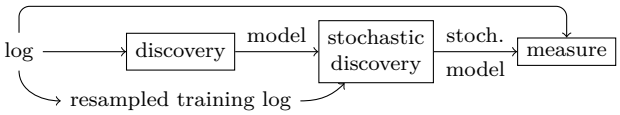


Figure 9: Quantitative evaluation set-up.

Table 2: Details of the experimental set-up.

(a) Logs

BPIC12-approvals [42]	BPIC12-a
BPIC13-closed problems [43]	BPIC13-cp
BPIC13-open problems [41]	BPIC13-op
BPIC17-offers [44]	BPIC17-O
BPIC20-domestic declarations [45]	BPIC20-dd
BPIC20-international declarations [45]	BPIC20-id
BPIC20-prepaid travel costs [45]	BPIC20-pt
BPIC20-request for payment [45]	BPIC20-rf
Example of Sect. 5	example log
MIMIC services [46]Appendix A	mimic-serv
MIMIC transfers [46]Appendix A	mimic-trans
Sepsis cases [47]	sepsis
BPIC2018-parcel document [48]	BPIC2018-6
Road fines [49]	roadfines

(b) Control-flow discovery.

BPIC12-a	Inductive Miner - infrequent (0.8) [50] transformed IMf to a Petri net
BPIC13-cp	
BPIC13-op	Directly Follows Model Miner (0.8) [40] transformed DFM to a Petri net
BPIC17-O	

(c) Stochastic discovery techniques.

Alignment-based estimator [23]	ABE
Frequency-based estimator [23]	FBE
This paper without reductions or dependencies	SDwr
This paper	SD

(d) Measures.

Number of transitions in control-flow model	transitions
Number of non-1 weights or dependencies	weights
Stochastic model discovering time (excl. control-flow)	time
Unit Earth Movers' Conformance Checking [36]	uEMSC

first discover control-flow models using standard control-flow process discovery techniques. Second, we resample each log to avoid evaluating on training data. From each of these resampled logs, combined with a discovered control-flow model, we discover a stochastic model using several stochastic discovery techniques. Lastly, we measure the quality of the models on the original logs using several stochastic-based measures, including a measure of stochastic quality (uEMSC [36]) and a measure of simplicity of the stochastic model (number of non-1 weights or dependencies). To nullify random effects, the procedure is repeated 10 times and averages are reported.

Computational Feasibility. The experiments were performed on an AMD EPIC 2GHz CPU with 40GB RAM available. The logs mentioned in Tab. 2 are those for which at least SDwr or SD could be computed for all 10 repetitions. The results have been included in Tab. B.4.

To provide a full picture of the computational feasibility of the technique, we also report on the attempted logs for which our techniques were not successful. Logs for which neither SDwr or SD could be computed for some or all of the 10 repetitions were:

- BPIC11 [51]; for this log, no alignments could be computed with our computing resources, and henceforth only FBE proved feasible.
- For the BPIC13-incidents [52] log, a couple of models were discovered (IMf-SD 1, IMf-SDwr 4, DFM-SD 2, DFM-SDwr 5); for the other repetitions,

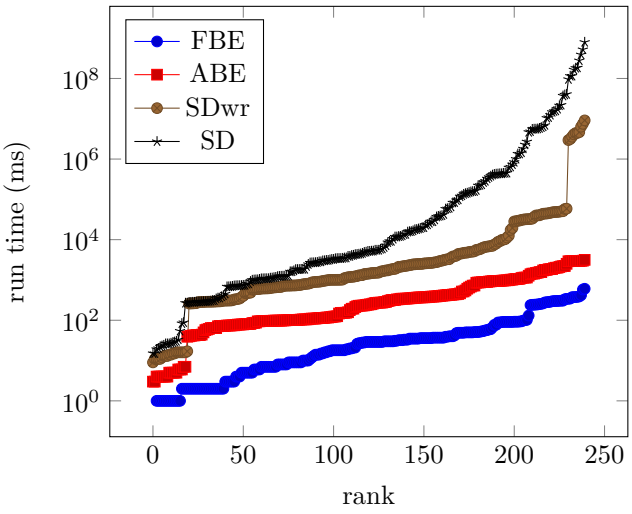


Figure 10: Run times in our qualitative experiment.

Table 3: Summary of our quantitative results.

stochastic algorithm	lowest weights	highest uEMSC	pareto-optimal
FBE	13	1	11
ABE	0	11	3
SDrw	8	9	16
SD	8	21	22

our optimiser did not converge, as e.g. one of the groups for DFM-SD had, with 7 transitions, 99 parameters to optimise.

- For the Sepsis [47] log, we could only obtain 3 SLPN-SDs for DFM-SD: time was the limiting factor here.

For the log mimic-serv, our discovery techniques successfully discovered all models. However, for IMf-SDwr and IMf-SD, the uEMSC measure did not complete and ran out of time.

Fig. 10 shows the distribution of stochastic discovery run times in the experiment. Observing the run time of the stochastic discovery techniques, we can observe that FBE is by far the fastest technique, as it does not compute alignments or perform any other type of optimisation. Second, the ABE technique applies alignments and then uses the frequencies of the observed transition firings as weights. As both SDwr and SD include alignments before starting the numerical optimisation, naturally these techniques require more time. While reaching a maximum of 9 days, half the models was discovered in 10 seconds or less.

Results. Tab. 3 summarises the results; for more details, please refer to Tab. B.4 in Appendix B. SD, our new technique that discovers SLPN-SDs, is the clear winner: of the tested techniques, it has the highest uEMSC the most often and discovers a pareto-optimal model in 22 of the 25 complete set-ups.

As SD considers a quadratic number of potential parameters in an SLPN-SD, one would expect over-fitting to be a potential challenge. In the experiments, we aimed to address over-fitting by not measuring uEMSC on the training data, but on a re-sampled event log. We see the effect of the re-sampling in the example log: a manual verification shows that the model discovered by IMf is our example control-flow model, however due to the re-sampling, uEMSC is 0.97 instead of 1, as the discovered SLPN-SD is close, but not entirely equal to, our example input model.

As a baseline, we included our technique without reductions and stochastic dependencies (SDwr): all adjustment weight parameters were fixed to 1, such that only the base weight parameters remain. We see two effects in Tab. 3: it reduces the number of weights unequal to 1 compared to ABE (the optimiser does not need all parameters to express the stochastic behaviour and leaves some at 1), and it does not always achieve an as high uEMSC as ABE (due to the optimiser not guaranteeing optimality and other factors, discussed in Sect. 7).

FBE often has the lowest number of weights, due to it not assigning weights to silent transitions at all; thus in any IMf model, it has the lowest number of non-1 weights. The 0.00 uEMSC values for FBE were double-checked: they are in the order of 10^{-7}, due to the technique assigning the number of times a transition was executed to each visible transition and 1 to each invisible transition. This yields a very low probability for any trace that must take a silent transition where also a visible transition is enabled, leading to the low scores on the trace-based uEMSC measure.

For Sepsis - IMf, we suspect a different reason for the uEMSC measures being quite low: the IMf model of the Sepsis log contains a large concurrent block (5 branches of which 2 contain loops). This yields such a high number of potential traces, each having a non-zero probability, that any trace seen in the log has a quite low probability of actually occurring according to the discovered stochastic models. SD here has the advantage of having more leverage to putting stochastic structure in the concurrent behaviour.

We conclude that our approach has the potential to outperform existing stochastic approaches, at the cost of longer run times.

7. Discussion

The stochastic dependencies reported by our discovery technique can be interpreted as follows. Take transitions a and b, and a reported adjustment weight parameter $\varphi_{a,b} \neq 1$, obtained using a certain significance level α. Then, a and b are dependent with $1 - \alpha$ certainty, and the best explanation of the relation between a and b is that with every execution of a, the weight of b increases $\varphi_{a,b}$-fold, under assumption of: (1) exponentially increasing weights

(Def. 1) (2) an optimal result of the optimiser used in our implementation, which is not guaranteed, (3) correctness of the input process model and event log, and (4) the event log containing enough information – where the precise notion of completeness is subject of future research. Please note that this is an associational result and not a causal statement: from the technique presented in this paper we can *not* conclude that by increasing the likelihood of a we can increase the likelihood of b.

Even though the base and adjustment weights are *a* best explanation, many degrees of freedom exist to express a single stochastic language, as shown in Sec. 4.2. As such, it may be challenging to interpret the values of the dependencies or to compare strengths of dependencies, even within the same SLPN-SD. Therefore, we chose not to indicate relation strength in our attemptly intuitive visualisation (Fig. 6). We argue that this challenging interpretability is shared with non-free choice constructs in Petri nets. Furthermore, please note that our method assumes a Petri net as input and adds complexity on top of it. Thus, for real-life cases for which an understandable Petri net is not available, our method will be challenged for understandability as well.

In Sec. 5.3, we introduced several reductions, which apply a symmetry to several parameters in order to fix a parameter and make the problem search space smaller. This yields a certain freedom in selecting parameters, which we argue can be leveraged using the following principles: (1) some of the parameters that may be fixed may already have been fixed by other reductions, thus these are best avoided to maximise the number of fixed parameters; (2) a weight of, or a dependency of or on, a silent transition is a preferable candidate to fix, as interpreting such dependencies or weights may be challenging: silent transitions are abstract modelling formalism constructs that may have only indirect interpretations in real-life processes.

Special care should be taken when applying the method proposed in this paper repeatedly: the statistical significance quantifies the uncertainty of the reported dependencies, but still the reported dependencies may be due to random chance. That is, spurious dependencies may be reported, especially when applying the approach of this paper repeatedly. As such, it remains important to validate the obtained dependencies using domain knowledge and apply Occam's razor: the simplest explanation is typically the best.

8. Conclusion

Considering the stochastic perspective in process models aids process optimisation and support efforts in organisations. The stochastic behaviour of real-life processes is often not static: the likelihood of alternatives in a process may depend on decisions taken earlier in the process. To capture dynamic stochastic perspectives we extended SLPNs with *stochastic dependencies* (SLPN-SDs), where the weight of a transition may depend on earlier-executed transitions. We described several symmetries that yield behaviour-equivalence classes of SLPN-SDs, and we showed how these can be used to reduce SLPN-SDs during and after discovery. We proved that SLPN-SDs have a strictly larger expressivity than

SLPNs, and introduced a technique to discover SLPN-SDs from an event log and a standard process model. As such, this technique is a stochastic process discovery technique. The discovery technique and two corresponding conformance checking techniques have been implemented in an open-source tool. The approach was evaluated on several real-life logs and shown to be able to provide insights that are not possible with standard SLPNs and to, in some cases, produce higher-quality models than existing techniques, in terms of EMSC with respect to test logs, and simplicity, in terms of relevant parameters.

Future work includes additional symmetries and simplifications to further increase readability, as well as improving visualisations or generating textual summaries of stochastic dependencies in SLPN-SDs with lots of dependencies. It would also be interesting to obtain the necessary information from a model rather than from a log, where possible. Further development of sampling techniques on positive-feedback loops will enable additional conformance checking techniques. Furthermore, it would be interesting to study different models of stochastic dependencies, for instance to support local semantics, additive weight functions or different types of influence models, such as the sigmoid function in combination with logistic regression; these influence models could also use other available data beyond the execution history. Neural networks could provide value for recommendation purposes. Finally, further study into causal dependencies may provide additional insights.

References

[1] A. Kalenkova, J. Carmona, A. Polyvyanyy, M. L. Rosa, Automated repair of process models using non-local constraints, in: R. Janicki, N. Sidorova, T. Chatain (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, 2020, pp. 280–300.

[2] J. Yuan, C. Duan, Q. Wei, A novel process mining algorithm to discover non-free choice construct from event logs, in: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, CSAE 2019, Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3331453.3360956.
URL https://doi.org/10.1145/3331453.3360956

[3] K. M. van Hee, A. Serebrenik, N. Sidorova, W. M. P. van der Aalst, Working with the past: Integrating history in Petri nets, Fundam. Informaticae 88 (3) (2008).

[4] B. F. van Dongen, et al., The ProM framework: A new era in process mining tool support, in: Petri nets, Vol. 3536 of LNCS, 2005.

[5] A. J. M. M. Weijters, J. T. S. Ribeiro, Flexible heuristics miner (FHM), in: CIDM, 2011.

[6] S. K. L. M. vanden Broucke, J. D. Weerdt, Fodina: A robust and flexible heuristic process discovery technique, Decis. Support Syst. 100 (2017).

[7] J. D. Smedt, J. D. Weerdt, J. Vanthienen, Fusion miner: Process discovery for mixed-paradigm models, Decis. Support Syst. 77 (2015).

[8] A. K. A. de Medeiros, A. J. M. M. Weijters, W. M. P. van der Aalst, Genetic process mining: A basic approach and its challenges, in: BPM Workshops, Vol. 3812, 2005.

[9] É. Badouel, L. Bernardinello, P. Darondeau, Petri Net Synthesis, Texts in Theoretical Computer Science. An EATCS Series, 2015.

[10] J. Carmona, J. Cortadella, M. Kishinevsky, A region-based algorithm for discovering Petri nets from event logs, in: BPM, Vol. 5240 of LNCS, 2008.

[11] W. M. P. van der Aalst, et al., Process mining: A two-step approach to balance between underfitting and overfitting, SoSyM 9 (2010).

[12] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, A. Serebrenik, Process discovery using integer linear programming, Fundam. Inf. 94 (3–4) (2009).

[13] L. Wen, W. M. P. van der Aalst, J. Wang, J. Sun, Mining process models with non-free-choice constructs, DMKD 15 (2007).

[14] L. L. Mannel, W. M. P. van der Aalst, Finding complex process-structures by exploiting the token-game, in: Petri nets, Vol. 11522 of LNCS, 2019.

[15] T. Tapia-Flores, E. López-Mellado, A. P. Estrada-Vargas, J.-J. Lesage, Discovering Petri net models of discrete-event processes by computing T-invariants, IEEE TASE 15 (3) (2018).

[16] V. Liesaputra, S. Yongchareon, S. Chaisiri, Efficient process model discovery using maximal pattern mining, in: BPM, Vol. 9253 of LNCS, 2015.

[17] N. Tax, N. Sidorova, R. Haakma, W. M. P. van der Aalst, Mining local process models, Journal of Innovation in Digital Ecosystems 3 (2) (2016) 183–196.

[18] S. Goedertier, D. Martens, J. Vanthienen, B. Baesens, Robust process discovery with artificial negative events, JMLR 10 (2009).

[19] A. A. Kalenkova, J. Carmona, A. Polyvyanyy, M. La Rosa, Automated repair of process models with non-local constraints using state-based region theory, CoRR abs/2106.15398 (2021).

[20] M. Chabrol, B. Dalmas, S. Norre, S. Rodier, A process tree-based algorithm for the detection of implicit dependencies, in: RCIS, 2016.

[21] A. Rogge-Solti, W. M. P. van der Aalst, M. H. Weske, Discovering stochastic Petri nets with arbitrary delay distributions from event logs, in: BPM Workshops, LNBIP, 2014.

[22] A. Burke, S. J. J. Leemans, M. T. Wynn, Stochastic process discovery by weight estimation, in: ICPM workshops, Vol. 406 of LNBIP, 2021.

[23] A. Burke, S. J. J. Leemans, M. T. Wynn, Discovering stochastic process models by reduction and abstraction, in: Petri nets, Vol. 12734 of LNCS, 2021.

[24] H. Schonenberg, N. Sidorova, W. M. P. van der Aalst, K. M. van Hee, History-dependent stochastic Petri nets, in: A. Pnueli, I. B. Virbitskaite, A. Voronkov (Eds.), PSI 2009, Vol. 5947 of LNCS, Springer, 2009.

[25] J. Jian, Mining simulation models with correlations, Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2009).

[26] H. Schonenberg, J. Jian, N. Sidorova, W. van der Aalst, Business trend analysis by simulation, in: B. Pernici (Ed.), Advanced Information Systems Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 515–529.

[27] M. Pesic, H. Schonenberg, W. M. P. van der Aalst, DECLARE: Full support for loosely-structured processes, in: EDOC, 2007.

[28] F. M. Maggi, M. Montali, R. Peñaloza, A. Alman, Extending temporal business constraints with uncertainty, in: BPM, Vol. 12168 of LNCS, 2020.

[29] S. J. J. Leemans, N. Tax, Causal reasoning over control-flow decisions in process models, in: CAiSE, Vol. 13295 of LNCS, 2022.

[30] M. S. Qafari, W. M. P. van der Aalst, Case level counterfactual reasoning in process mining, in: S. Nurcan, A. Korthaus (Eds.), Intelligent Information Systems - CAiSE Forum 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings, Vol. 424 of Lecture Notes in Business Information Processing, Springer, 2021, pp. 55–63. doi:10.1007/978-3-030-79108-7_7.
URL https://doi.org/10.1007/978-3-030-79108-7_7

[31] Z. D. Bozorgi, I. Teinemaa, M. Dumas, M. L. Rosa, A. Polyvyanyy, Prescriptive process monitoring for cost-aware cycle time reduction, in: C. D. Ciccio, C. D. Francescomarino, P. Soffer (Eds.), 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021, IEEE, 2021, pp. 96–103. doi:10.1109/ICPM53251.2021.9576853.
URL https://doi.org/10.1109/ICPM53251.2021.9576853

[32] J. J. Koorn, X. Lu, H. Leopold, N. Martin, S. Verboven, H. A. Reijers, Mining statistical relations for better decision making in healthcare processes, in: International Conference on Process Mining, 2022.

[33] M. Shoush, M. Dumas, Prescriptive process monitoring under resource constraints: A causal inference approach, in: J. Munoz-Gama, X. Lu (Eds.),

Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers, Vol. 433 of Lecture Notes in Business Information Processing, Springer, 2021, pp. 180–193. doi:10.1007/978-3-030-98581-3_14.
URL https://doi.org/10.1007/978-3-030-98581-3_14

[34] J. J. Koorn, X. Lu, H. Leopold, H. A. Reijers, From action to response to effect: Mining statistical relations in work processes, Inf. Syst. 109 (2022) 102035. doi:10.1016/j.is.2022.102035.
URL https://doi.org/10.1016/j.is.2022.102035

[35] L. Breiman, Better subset regression using the nonnegative garrote, Technometrics 37 (1995) 373–384.

[36] S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, A. Polyvyanyy, Stochastic process mining: Earth movers' stochastic conformance, Inf. Syst. 102 (2021).

[37] W. M. P. van der Aalst, A. Adriansyah, B. F. van Dongen, Replaying history on process models for conformance checking and performance analysis, DMKD 2 (2) (2012).

[38] K. Pearson, X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 50 (302) (1900) 157–175.

[39] C. Bonferroni, Teoria statistica delle classi e calcolo delle probabilita, Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze 8 (1936) 3–62.

[40] S. J. J. Leemans, E. Poppe, M. T. Wynn, Directly follows-based process mining: Exploration & a case study, in: ICPM, 2019.

[41] W. Steeman, BPI Challenge 2013, open problems (4 2013). doi:10.4121/uuid:3537c19d-6c64-4b1d-815d-915ab0e479da.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_open_problems/12688556

[42] B. van Dongen, BPI Challenge 2012 (4 2012). doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204

[43] W. Steeman, BPI Challenge 2013, closed problems (4 2013). doi:10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_closed_problems/12714476

[44] B. van Dongen, BPI Challenge 2017 - Offer log (2 2017). doi:10.4121/uuid:7e326e7e-8b93-4701-8860-71213edf0fbe.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2017_-_Offer_log/12705737

[45] B. van Dongen, Bpi challenge 2020 (Mar 2020). doi:10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51.
URL https://data.4tu.nl/collections/BPI_Challenge_2020/5065541/1

[46] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, R. G. Mark, Mimic-iii, a freely accessible critical care database, Scientific data 3 (1) (2016) 1–9.

[47] F. Mannhardt, Sepsis Cases - Event Log (12 2016). doi:10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.
URL https://data.4tu.nl/articles/dataset/Sepsis_Cases_-_Event_Log/12707639

[48] B. van Dongen, F. F. Borchert, BPI Challenge 2018 (3 2018). doi:10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2018/12688355

[49] M. M. de Leoni, F. Mannhardt, Road Traffic Fine Management Process (2 2015). doi:10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.
URL https://data.4tu.nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249

[50] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: BPM Workshops, Vol. 171 of LNBIP, 2013.

[51] B. van Dongen, Real-life event logs - Hospital log (3 2011). doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54.
URL https://data.4tu.nl/articles/dataset/Real-life_event_logs_-_Hospital_log/12716513

[52] W. Steeman, BPI Challenge 2013, incidents (4 2013). doi:10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.
URL https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_incidents/12693914

[53] S. J. J. Leemans, Filtertree: a repeatable branching xes editor, in: International Conference on Process Mining - demo papers, 2022.

Appendix A. Log preparation

From the MIMIC [46] data, logs were constructed from the SERVICES.CSV and TRANSFERS.CSV files as follows, using the FilterTree [53] tool:

SERVICES.CSV:

```
CSV to XES | HADM_ID
copy event attribute | "CURR_SERVICE" "concept:name"
```

TRANSFERS.CSV:

```
CSV to XES | HADM_ID
combine event attributes | concept:name EVENTTYPE PREV_CAREUNIT
make event attribute a timestamp | OUTTIME "yyyy-M-d H:mm:ss"
copy event attribute | OUTTIME time:timestamp
annotate trace with sum of event attribute values | LOS
rename trace attribute | "LOS (sum)" cost:total
remove empty traces |
sort events |
sample traces with seed | 0.05 1
```

Appendix B. Quantitative experiment results

Table B.4: Results of the model-quality experiment.

Log	discovery	stochastic discovery	transitions	weights	time (ms)	uEMSC
BPIC12-a	IMf	FBE	15	10	110	0.19
		ABE	15	15	137	0.48
		SDwr	15	12	452	0.48
		SD	15	19	11726852	0.76
	DFM	FBE	16	13	32	0.73
		ABE	16	16	147	0.83
		SDwr	16	5	349	0.83
		SD	16	5	378	0.83
BPIC13-cp	IMf	FBE	15	6	32	0.01
		ABE	15	15	104	0.42
		SDwr	15	11	732	0.40
		SD	15	41	392908	0.36
	DFM	FBE	13	12	3	0.00
		ABE	13	13	108	0.65
		SDwr	13	10	1970	0.65
		SD	13	24	8611	0.71
BPIC13-op	IMf	FBE	16	5	2	0.01
		ABE	16	16	99	0.43
		SDwr	16	14	1281	0.41
		SD	16	51	1588670	0.02
	DFM	FBE	17	12	2	0.22
		ABE	17	17	43	0.75
		SDwr	17	16	1034	0.75
		SD	17	15	3461	0.75
BPIC17-o	IMf	FBE	9	8	94	0.54
		ABE	9	9	400	0.58
		SDwr	9	6	1536	0.58
		SD	9	8	1423	0.53
	DFM	FBE	10	7	120	0.84
		ABE	10	10	431	0.84
		SDwr	10	3	1492	0.84
		SD	10	2	1225	0.84

Table B.4: Results of the model-quality experiment.

Log	discovery	stochastic discovery	transitions	weights	time (ms)	uEMSC
BPIC20-dd	IMf	FBE	33	14	30	0.00
		ABE	33	32	265	0.80
		SDwr	33	24	2521	0.80
		SD	33	38	15677	0.78
	DFM	FBE	9	8	35	0.58
		ABE	9	9	134	0.81
		SDwr	9	3	336	0.81
		SD	9	2	1222	0.81
BPIC20-id	IMf	FBE	66	30	36	-0.00
		ABE	66	64	2972	0.27
		SDwr	66	47	30773	0.27
		SD	66	96	58834200	0.38
	DFM	FBE	51	48	40	0.00
		ABE	51	51	986	0.18
		SDwr	51	39	49653	0.18
		SD	51	199	108463726	0.57
BPIC20-pt	IMf	FBE	63	26	37	0.00
		ABE	63	62	879	0.22
		SDwr	63	50	10378	-
		SD	63	109	124893709	-
	DFM	FBE	31	29	44	0.16
		ABE	31	31	226	0.39
		SDwr	31	20	1733	0.40
		SD	31	40	9902	0.79
BPIC20-rf	IMf	FBE	29	15	52	-0.00
		ABE	29	28	114	0.63
		SDwr	29	20	948	0.63
		SD	29	37	19271	0.78
	DFM	FBE	12	11	18	0.55
		ABE	12	12	83	0.81
		SDwr	12	5	472	0.81
		SD	12	5	296	0.82
example log	IMf	FBE	5	5	40	0.92
		ABE	5	5	5	0.92
		SDwr	5	4	13	0.92
		SD	5	3	89	0.97
	DFM	FBE	8	6	1	0.79
		ABE	8	8	47	0.80
		SDwr	8	3	15	0.80
		SD	8	2	106	0.80
mimic-serv	IMf	FBE	55	19	112	0.00
		ABE	55	55	1648	0.53
		SDwr	55	52	5020753	-
		SD	55	292	9669738	-
	DFM	FBE	19	10	56	0.72
		ABE	19	19	420	0.81
		SDwr	19	10	758	0.81
		SD	19	9	819	0.81
mimic-trans	IMf	FBE	37	17	10	0.00
		ABE	37	36	463	0.17
		SDwr	37	32	4735	0.13
		SD	37	72	293417	0.25
	DFM	FBE	37	33	44	0.51
		ABE	37	37	273	0.59
		SDwr	37	31	42047	0.59
		SD	37	57	329444	0.65
Sepsis	IMf	FBE	29	14	47	-0.00
		ABE	29	29	2359	0.00
		SDwr	29	25	495648	0.00
		SD	29	86	29143995	0.04
	DFM	FBE	77	67	51	0.00
		ABE	77	77	2619	0.05
		SDwr	77	74	18110531	0.05
		SD	-	-	-	-
BPIC2018-6	IMf	FBE	24	9	86	0.00
		ABE	24	24	3075	0.07
		SDwr	24	19	2938526	0.08
		SD	24	105	126062338	0.14
	DFM	FBE	16	15	146	0.00
		ABE	16	16	1225	0.20
		SDwr	16	13	1363601	0.20
		SD	16	79	19686010	0.67

Table B.4: Results of the model-quality experiment.

Log	discovery	stochastic discovery	transitions	weights	time (ms)	uEMSC
Roadfines	IMf	FBE	24	11	309	0.01
		ABE	24	24	2025	0.29
		SDwr	24	19	7849	0.37
		SD	24	54	835911	0.88
	DFM	FBE	9	6	309	0.68
		ABE	9	9	1152	0.82
		SDwr	9	3	3636	0.82
		SD	9	2	3614	0.82