Investigating the Influence of Data-aware Process States on Activity Probabilities in Simulation Models: Does Accuracy Improve?

Massimiliano de Leoni[1], Francesco Vinci[1], Sander J.J. Leemans[2], and Felix Mannhardt[3]

[1] University of Padua
massimiliano.deleoni@unipd.it, francesco.vinci.1@phd.unipd.it
[2] RWTH Aachen
s.leemans@bpm.rwth-aachen.de
[3] Eindhoven University of Technology
f.mannhardt@tue.nl

Abstract. Business process simulation enables analysts to run a process in different scenarios, compare its performances and consequently provide indications on how to improve a business process. Process simulation requires one to provide a simulation model, which should accurately reflect reality to ensure the reliability of the simulation findings. An accurate simulation model passes through a correct stochastic modelling of the activity firings: activities are associated with the probability of each to fire. Literature determines these probabilities by looking at the frequency of the activity occurrences when they are enabled. This is a coarse determination, because this way does not consider the actual process state, which might influence the probabilities themselves (e.g., a thorough loan assessment is more likely for larger loan requests). The process state is as a faithful abstraction of the process instance execution so far, including the process-variable values, the activity firing history, etc. This paper aims to investigate how process states can be leveraged to improve activity firing probabilities. A technique has been put forward and compared with the baseline where basic branching probabilities are employed. Experimental results show that, indeed, business simulation models are more accurate to replicate the real process' behavior.

Keywords: Process Simulation · Stochastic Models · Branching Probabilities · Process Mining

1 Introduction

Business process simulation refers to techniques for the simulation of business process behavior on the basis of a process simulation model, a process model extended with additional information for a probabilistic characterization of the different run-time aspects (case arrival rate, activity durations and probabilities, roles, etc.). Simulation provides a flexible approach to analyse and improve business processes. Through simulation experiments, various 'what if' questions can

be answered, and redesigning alternatives can be compared with respect to some key performance indicators. The main idea of business process simulation is to carry out a significantly large number of runs, in accordance with a simulation model. Statistics over these runs are collected to gain insight into the processes, and to determine the possible issues (bottlenecks, wastes, costs, etc.). By applying different changes to the simulation model, one can assess the consequences of these changes without putting them in production, and consequently can explore dimensions to possible process improvements.

A successful application of business process simulation for process improvement relies on a simulation model that reflects the real process behavior; conclusions drawn on an unrealistic simulation model lead to process redesigns that may not yield improvements, or even may worsen the performances. A large body of research work has already focused on accurately creating process models and several of their run-time aspects (cf. Section 5), including case arrival rate, activity durations, and roles. However, no recent work has focused on accurately estimating, given a set of enabled activities at run time, the probability of each to occur. Currently, this determination is solely based on the branching probabilities. Of course, this is a course determination, which does not consider that probabilities of activities to occur may vary as function of the characteristics of the process instances, the activities that previously occurred in the same process instance, time-related information, etc. Consider, for example, a loan application process: the probability of executing an activity about a thorough assessment grows, e.g., with the amount of the requested loan, and decreases with annual salary of the applicant. Also, the probability of rejecting an application may grow with the number of requests to the applicant of providing further documents.

This paper introduces the concept of process state, which is a faithful abstraction of a case, and investigates the research question how a proper choice of this abstraction allows a more accurate estimation of the activity firing probabilities of simulation models, with respect to the simple branching probabilities. More accurate firing probabilities lead to more accurate simulation models.

In order to answer this research question, the paper builds upon Petri nets, and discovers a so-called weight function for each transition, which is defined over a process state, which, e.g., can include process variables, and the transition-firing history. Then, for the cases that are simulated, the current process state is computed and used to evaluate the weight function of the transitions that are enabled at that point. The probability to fire a transition is thus obtained as the ratio of its weight and the sum of the weights of all enabled transitions. It follows that the higher is the weight of a transition, the higher is the probability of that transition to fire. Since the case characteristics that may influence the probability may depend on the specific process that is being simulated, we propose a framework where the process-state abstraction can be customized to include or exclude certain characteristics. In Section 2, some examples for process states are provided. In this paper, we report on the use of logistic regression to learn

the weight function, but other approaches could be alternatively employed, such as regression trees.

The research question is finally answered by applying the aforementioned framework to two real-life processes, each with a real-life event log. For each process, five different definitions of process states have been considered to compute weights of the Petri-net models of the two processes to be simulated. The results show that the simulation using models with transition probabilities based on process states allows obtaining simulation results that are more accurate, if compared with simulation models based on branching probabilities.

Section 2 starts introducing the preliminary concepts of event logs, and then continues (i) introducing the novel notion of the process-state abstraction and (ii) their usage with stochastic labelled Data Petri nets, a Petri-net extension to associate weights to transitions. Section 3 discusses how to compute the transition weights as function of a customizable abstraction of the process state. Section 4 illustrates how Stochastic labelled Data Petri nets can be represented via Coloured Petri nets in CPN Tools, and reports on the benefits for simulation models to use transition probabilities based on process states. Section 5 reports on the related works, and Section 6 concludes the paper.

2 Event Logs and Stochastic Data Petri Nets

The determination of the weights is obtained from an analysis of an event log of the process that aims to be simulated:

Definition 1 (Events). *Let $\mathcal{A}$ be the set of process' activities. Let $\mathcal{V}$ the set of process attributes. Let $\mathcal{W}_\mathcal{V}$ be a function that assigns a domain $\mathcal{W}_\mathcal{V}(x)$ to each process attribute $x \in \mathcal{V}$. Let $\overline{\mathcal{W}} = \cup_{x \in \mathcal{V}} \mathcal{W}_\mathcal{V}(x)$. An event is a tuple $(a, v) \in \mathcal{A} \times (\mathcal{V} \not\rightarrow \overline{\mathcal{W}})$ where a is the event activity, v is a partial function assigning values to process attributes with $v(x) \in \mathcal{W}_\mathcal{V}(x)$.*

A trace is a sequence of events, the same event can potentially occur in different traces, namely attributes are given the same assignment in different traces. This means that potentially the entire same trace can appear multiple times. This motivates why an event log is to be defined as a multiset of traces:[4]

Definition 2 (Traces & Event Logs). *Let $\mathcal{E} = \mathcal{A} \times (\mathcal{V} \not\rightarrow \overline{\mathcal{W}})$ be the universe of events. A trace σ is a sequence of events, i.e. $\sigma \in \mathcal{E}^*$. An event-log $\mathcal{L}$ is a multiset of traces, i.e. $\mathcal{L} \subset \mathbb{B}(\mathcal{E}^*)$.*

In this paper, simulation models are provided in form of so-called Stochastic Labelled Data Petri Nets (SLDPNs). While SLDPNs are not able to represent every aspect relevant for simulation models, they are simple, yet sufficient to discuss and formalize the concepts behind activity probabilities. In SLDPNs, a transition firing consists in executing a transition and assigning values to some process attributes. The sequence of transition firings determines the process state:

[4] $\mathbb{B}(X)$ indicates the set of all multisets with the elements in set X.

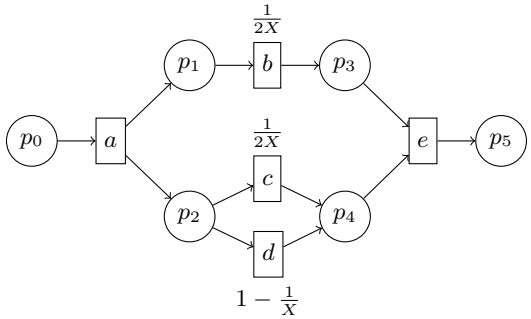

Fig. 1: Example of a Stochastic Data Petri Net. Transitions are annotated with the weights, when the latter are present.

Definition 3 (Process State). *Let T be a set of transitions. Let $\mathcal{V}$ the set of process attributes. Let $\mathcal{W}_\mathcal{V}$ be a function that assigns a domain $\mathcal{W}_\mathcal{V}(x)$ to each process attribute $x \in \mathcal{V}$. Let $\overline{\mathcal{W}} = \cup_{x \in \mathcal{V}} \mathcal{W}_\mathcal{V}(x)$. Let Δ be the set of process states. A process-state function maps a sequence of transition firings to a process state: $\mathcal{S}_\Delta : (T \times (\mathcal{V} \not\rightarrow \overline{\mathcal{W}}))^* \rightarrow \Delta$.*

Note that the marking is not part of the process state (see below). A process-state function can be customized, according to the specific domain. For instance, if one wants to account for the process attributes in the set $\mathcal{V}$, the set $\overline{\Delta}$ of possible process states consists of tuples $(x_1, \ldots, x_n)$ where x_i is the value assigned to variable $v_i \in \mathcal{V}$, after defining an ordering of the attributes in $\mathcal{V}$. In particular, for a sequence $\sigma = \langle (t_1, f_1), \ldots, (t_m, f_m) \rangle$ of transition firings, $S_{\overline{\Delta}}$ returns a tuple $(x_1, \ldots, x_n)$ in which x_i is the latest value assigned to v_i in σ, namely there is a transition firing $(t_j, f_j) \in \sigma$ such that $f_j(v_i) = x_i$ and, for all $j < k \leq m$, v_i is not in the domain of f_k.

In SLDPNs, each transition is associated to a weight function that is dependent on a process state.

Definition 4 (Stochastic Labelled Data Petri Net - syntax). *Let A be a set of activities, and Δ the set of possible process states. A stochastic labelled data Petri net (SLDPN) is a tuple $(P, T, F, \lambda, \Delta, M_0, \mathbb{w})$, such that (P, T, F) is a Petri Net, $\lambda : T \not\rightarrow \mathcal{A}$ be a labelling function, M_0 is the initial marking, and $\mathbb{w}\colon T \times \Delta \rightarrow \mathbb{R}^+$ is a weight function.*

Example. Figure 1 shows an example of an SLDPN. The control flow of this SLDPN consists of an AND split followed by the parallel executions of b and a choice between c and d. The transitions are annotated with weight functions depending on the continuous variable X.

The state of an SLDPN is the combination of a marking and an process state $d \in \Delta$. Hereafter, when clear from the context, the process state is simply referred to as state. The marking determines which transitions are enabled, while the process state influences the probabilities of transitions:

Definition 5 (Stochastic Labelled Data Petri Net - semantics). *Let* $N = (P, T, F, \lambda, \Delta, M_0, \mathbb{w})$ *be an SLDPN. Let* $\sigma \in (T \times (\mathcal{V} \not\to \overline{\mathcal{W}}))^*$ *be a sequence of transition firings, leading to marking* M*. Let* S_Δ *be the process-state function, and* $E(M) \subseteq T$ *be the set of transitions enabled at marking* M *of Petri net* (P, T, F)*. The probability to fire* t *after* σ *is:*

$$Pr_N(t, M, \sigma) = \frac{\mathbb{w}(t, S_\Delta(\sigma))}{\sum_{t' \in E(M)} \mathbb{w}(t', S_\Delta(\sigma))}.$$

3 Framework for Determination of Weights

In this section, we introduce a framework that, given an event log $\mathcal{L}$, a Petri net (P, T, F), a labelling function λ and an initial marking M_0, can be used to determine the weights of the transitions, thereby transform the Petri net into an SLDPN. The framework can be instantiated for a process-state function $\mathcal{S}_\Delta$, generalising the proposal in [10], and a parameterised weight function $\mathbb{w}$, and consists of four steps:

Step 1 For each trace $\sigma \in \mathcal{L}$, reconstruct the corresponding path of transitions that σ took through the model. This reconstruction is performed using a sequence of *moves*: a synchronous move combines an event (a, v) in the log trace with a transition t on the model path such that $\lambda(t) = a$; a model move is a transition on the model path; while a log move is an event in the log trace. Such a sequence of moves, where the projection of the sequential synchronous and log moves yields the trace, and the projection of the sequential synchronous and model moves yields the path, is called an *alignment* [1].
For a given trace of the event log, an optimal alignment is an alignment with a minimal number of log and model moves[5], over all paths in the model. Note that this alignment does not need to take the data values or weight functions into account and can be computed solely based on the regular Petri net and each trace of the event log.

Step 2 For one optimal alignment of each trace in the event log, we use the process-state function $\mathcal{S}_\Delta$ to reconstruct the sequence of process states Δ. By definition, any path of the model starts in the initial marking M_0 and the process state $\mathcal{S}_\Delta(\langle\,\rangle)$. For each synchronous or model move m in the optimal alignment, we have a transition t available. As the moves are sequential, we can take the partial function assigning values to process attributes (v) from the last synchronous move in the alignment, before m. If no such last move exists, we take an empty function.
As such, we obtain a sequence of transition firings (t, v). Through $\mathcal{S}_\Delta$, this sequence yields a sequence of process states $\delta \in \Delta$. Similarly, the sequence of markings can be derived from the model and the sequence of transition firings.

[5] For the minimalisation, we do not count model moves on unlabelled transitions.

Step 3 For each transition t in the model, we gather the observations made in each trace σ of the log related to this transition: each time that t was enabled and fired in the sequence of transition firings, the associated process state before firing t is recorded as positive observation. Each time that t was enabled but another transition fired a negative observation using the process state before firing that transition is recorded. Given these collected multisets of positive process states $\Delta_{t^+} \subset \mathbb{B}(\Delta)$ and negative process states $\Delta_{t^-} \subset \mathbb{B}(\Delta)$ we build a training set to learn the influence of process states on the weight of transition t as:

$$\biguplus_{\delta \in \Delta_{t^+}} [(\delta, 1)] \cup \biguplus_{\delta \in \Delta_{t^-}} [(\delta, 0)]$$

Step 4 We leverage any suitable machine learning model that supports the process state representation chosen as input to serve as parameterised weight function $\mathbb{w}$. Such a model should assign higher weights, e.g., the 1 in the training set, for those process states in which the transition t was observed to occur opposed to those in which another transition was observed. We can obtain the overall parameterised weight function by fitting a separate model for each transition t since the weights obtained for enabled transitions in the SLDPN are not required to sum up to 1.

As an example, we instantiate our framework with a process state function $\mathcal{S}_\Delta$ that takes into account (i) the event attribute values observed for the first event in a trace and (ii) the count of the activity occurrences in the history of the process instance:

$$\mathcal{S}_\Delta(\langle (t_1, v_1), \ldots, (t_n, v_n) \rangle) = (v_1, [t_1, \ldots t_n]).$$

Our process state is, thus, defined as $\Delta = (\mathcal{V} \not\rightarrow \overline{\mathcal{W}}) \times \mathbb{B}(T)$. We use the logistic function over the same attribute values and history as parameterised weight function $\mathbb{w}$.

Assume the example trace $\sigma = \langle a^{X=3}, d^{X=4}, d^{X=5}, b^{X=3}, e^{X=5} \rangle$ with a single process attribute X. We align σ in Step 1 to the model shown in Figure 1. This alignment is:

Log	$a^{X=3}$	$d^{X=4}$	$d^{X=5}$	$b^{X=3}$	$e^{X=5}$
Model	a	d	-	b	e

In Step 2, we transform this alignment into a sequence of process states:

$$\langle (X = 3, [a]), (X = 3, [a, d]), (X = 3, [a, b, d]), (X = 3, [a, b, d, e]) \rangle$$

Then, Step 3 constructs the observations:

$$\begin{aligned}
\Delta_{a^+} &= [(X = \bot, [])] & \Delta_{a^-} &= [] \\
\Delta_{b^+} &= [(X = 3, [a, d])] & \Delta_{b^-} &= [(X = 3, [a])] \\
\Delta_{c^+} &= [] & \Delta_{c^-} &= [(X = 3, [a])] \\
\Delta_{d^+} &= [(X = 3, [a])] & \Delta_{d^-} &= [] \\
\Delta_{e^+} &= [(X = 3, [a, b, d])] & \Delta_{e^-} &= []
\end{aligned}$$

In the final Step 4 the weight function is then approximated using logistic regression for each of the transitions using the training sets build from positive and negative observations. We use logistic regression since it provides white-box explanations and is more usable for simulators such as CPN Tools. Moreover, white-box simulators can be used for what-if analysis, whereas deep learning models cannot [6]. Logistic regression, and many other machine learning models, requires input variables to be numeric. Thus, we need to transform the multiset of activity occurrences into several variables, one for each activity in the process model. Similarly, we could use one-hot encoding for categorical variables. Finally, we obtain the coefficients for the logistic function and obtain the final SLDPN including the learned weight function.

4 Experiments

The experiment focuses on verifying the similarity between the original event logs and those obtained from simulation. Our probability model of activity firing is only supported by CPN Tools, which includes simulation features.

Section 4.1 illustrates the case studies that are used to assess the validity of the proposed approach. Section 4.2 discusses the experimental setup, including how CPN Tools has been employed to define our probability model. Finally, Section 4.3 reports the results, and discusses the findings.

4.1 Case studies

The approach has been evaluated in two different case studies for which a public event log and a reference model is available: Road-Traffic Fines and Sepsis (see [9] for details).

The Road Traffic Fines event log describes the process of managing road traffic fines by a local police force in Italy. The event log contains 150370 traces and 11 different activities, with 12 data attributes. Sepsis case study is a real-life event log obtained from an Enterprise Resource Planning (ERP) system of a regional hospital in The Netherlands. It contains 1050 traces, 16 different activities and several data attributes, most of them binary.

Figures 2 and 3 depicts the Petri Net models used for our evaluations, which are shown in Figure 12.8 and Figure 13.3 of [9]. The Road-Traffic Fines model

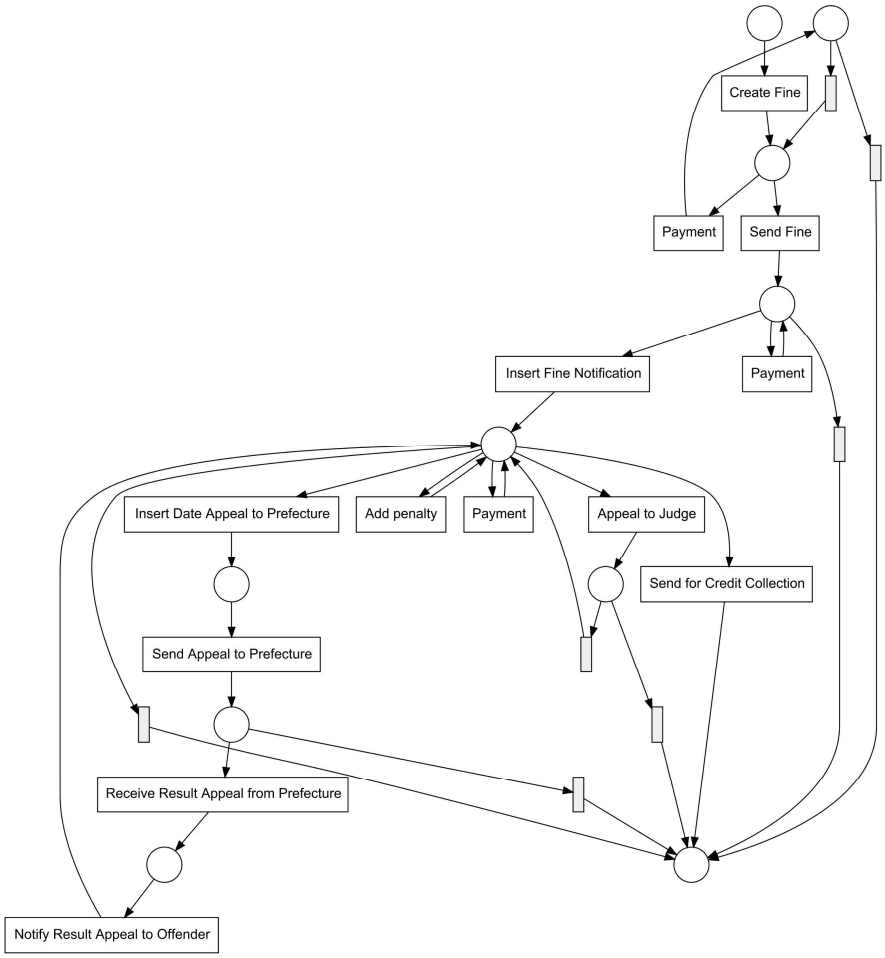

Fig. 2: Road-Traffic Fines Petri Net model.

allows executions with only one activity, i.e. `Create Fine`, however, the event log does not contain any trace such that. This led us to improve the model: a place and a transition after the first `Payment` are added to maintain the loop of it and remove the possibility of ending the trace after `Create Fine`.

4.2 Experimental Setup

The assessment is based on comparing the similarity of the original event logs with those obtained via simulation. The simulation models are constructed using five different characterization of the process state: with data only, namely using the values of the process attributes; with history only, namely using the number of occurrences of process activities; and with data and history, as well

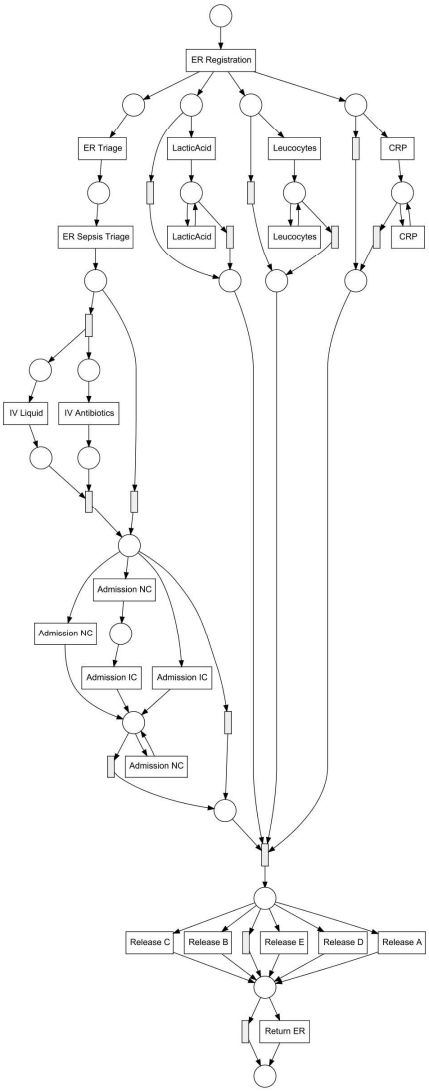

Fig. 3: Sepsis Petri Net model.

as with combinations of so-called binary history. History is also delineated in the binary version, where we only consider whether or not an activity has happened, irrespectively of the number of occurrences. As baseline of comparison, we also built the simulation model using branching probabilities, which were computed through the Multi-perspective Process Explorer [11]. The comparison of the original event log and those obtained via simulation is computed through the Earth-Movers' Stochastic distance introduced in [8]. This measure considers

the stochastic characteristics of the event logs: which activities are executed in which order, and how often a particular order of activities is executed.

For evaluating our approach, we divided the original logs into training and test sets using a temporal split: 70% to the training set and 30% to the test set. The training set was utilized to calculate the weights of the corresponding SLDPNs, and also the branching probabilities of the comparison baseline. By comparing the results of them, we can determine the effectiveness of our approach and assess the impact of different processing techniques on performance.

We instantiated the framework for determining the weights described in Section 3 in ProM by encoding the process state obtained into a set of attributes and implemented the parameterised weight function as a set of logistic regression models over that set of attributes and the binary dependent response variable. We use ridge regression as implemented in WEKA 3.8 [17] and implemented an export functionality to obtain the resulting coefficients $\beta_0, \ldots, \beta_n$ of the logistic model that are sufficient to determine the weight. More complex models may be added to the implementation in the future as long as their parameters can be used to compute weights based on process states. After exporting the logistic regression coefficients, we use them in the simulation models.

The simulation models are implemented using CPN Tools. In fact, one of the key advantages of CPN Tools is the ability to model and analyze complex systems. Additionally, the Standard ML programming language can be used to implement custom functions, making it possible to adapt the model to the specific needs of the simulation.

We illustrate how SLDPNs can be represented via CPN Tools through a simple example. In particular, we focus on the SLDPN in Figure 1. The CPN model consists of several parts, each with a specific function. The black part represents the Petri Net underneath, while the blue part focuses on the simulation of the CPN Tools: `n_sim` process instances are simulated one at time until the previous instance is completed. Note that this does not affect the event log similarity since the evaluation is based on control-flow and not on time-related measurements. This is done here for the sake of simplicity: it is trivial to extend it to allow for multiple process instance executions at the same time.

The brown part is related to generating the values of the process attributes for the different simulated traces. The literature proposes to assign a suitable statistical distribution to each process attribute (cf. Section 5): values are then sampled from those distributions. However, this might introduce noise in the experiments, if suitable distributions are not found. This ultimately leads to an unfair comparison of the original event log and that obtained from simulation. Recall that we simulate as many process instances, i.e. traces, as those in the original event log. We leverage on this, and, for each trace of the original event log, we use the same set of process-attribute values for one process instance that is simulated.

The green part of the model checks the enabled transitions and the trace history. Finally, the purple part is responsible for computing the probabilities of

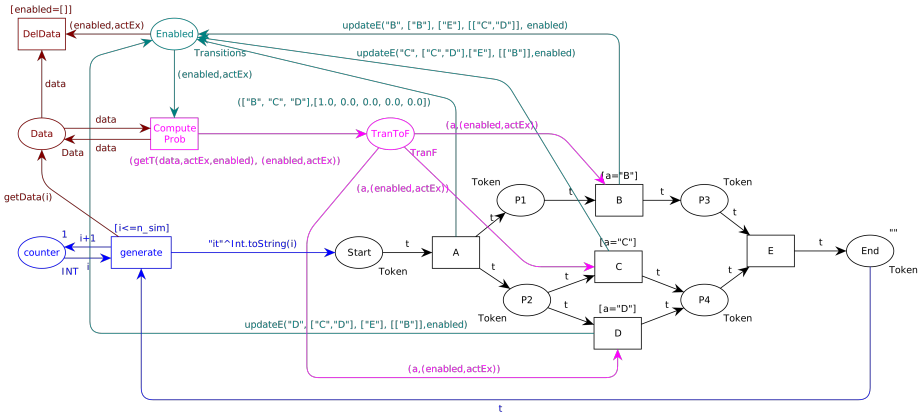

Fig. 4: Implementation of the SLDPN in Figure 1 in CPN Tools.

firing the enabled transitions using the weights obtained by the logistic regression models.

The arc inscriptions in the CPN Tools model example are used to implement the SLDPN models: the `getData` function returns a list of real value representing the `i`-th data state; the `updateE` function returns a tuple `(enabled, actEx)`, where `enabled` is the list of strings of the enabled transitions after having executed a certain transition, and `actEx` is a list of real values of length number of activities in which each value represents how many time the correspondent activity has been performed; `getT` returns the transition to fire according to the probabilities computed using a function that implements the formula in Definition 5, where $\mathfrak{w}$ is the logistic regression using the coefficients exported from the Prom Plug-In.

This procedure is then applied to the Road-Traffic Fine and Sepsis SLDPNs. Each simulated model has been used to generate as many traces as those in the original event log (respectively, 150370 and 1050 traces). The simulation reports generated by CPN Tools have been converted to obtain event logs in XES files using a parser which we implemented in Python.[6]

4.3 Experimental Results

For each case study, we have thus computed six simulation models, using probabilities based on the branching probabilities and on the weights with five different process-state abstractions. Each simulation model has been run ten times, to mitigate the stochasticity of simulation.

Table 1 shows the final results for each simulation model and case study. The reported measures are the average of the Earth Movers' distances between each simulated event log and the real one. The value in brackets are the maximum

[6] `https://github.com/franvinci/InfluenceofDataawareProcessStatesonActivityProbabilities`

Case Study	Branching Prob.	Data	History	Data & History	Bin. History	Data & Bin. History
Road-Traffic Fines	0.8793 (±0.001)	0.8831 (±0.001)	0.9526 (±0.001)	0.9542 (±0.001)	0.9352 (±0.001)	0.9427 (±0.001)
Sepsis	0.5309 (±0.008)	0.6245 (±0.009)	- -	- -	0.5906 (±0.005)	0.6201 (±0.006)

Table 1: Average Earth Movers' distance (with absolute error) for each configuration.

Fig. 5: Road-Traffic Fines absolute errors of the probabilities of occurrence of an activity a_j (columns) immediately after a_i (rows) between the real and the simulated event logs. Left: using branching probabilities. Right: using data and history. Darker colours indicate lower errors.

variation of similarity for each of ten simulated logs obtained for each configuration, wrt. the original event log. Note how the variation is negligible (less than 1%), thus positively hinting at the reliability of the results.

In the Road-Traffic Fines case study, it can be noticed a slight improvement in log similarity using only data features. However, by adding history, our approach outperforms the baseline of 7.5%: this suggests that, for this case, the data alone are insufficient for capturing the control flow stochasticity, while the history features provide valuable insights. Binary history also leads to good performance. However, configurations with history states perform better.

On the other hand, in the Sepsis case study, using the data features without history results in a 9.4% increase in baseline performance, proving that computing firing probabilities based on these features can lead to more accurate simulation models. We were unable to perform the experiments for the Sepsis case study when the process state abstraction includes the history. Indeed, the

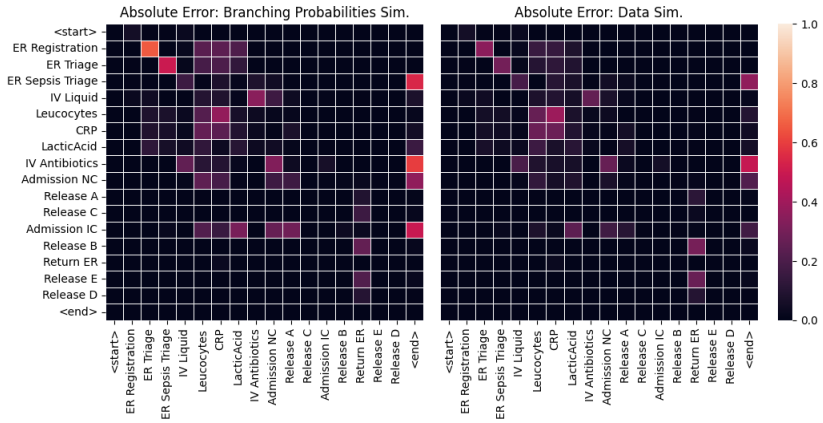

Fig. 6: Sepsis absolute errors of the probabilities of occurrence of an activity a_j (columns) immediately after a_i (rows) between the real and the simulated event logs. Left: using branching probabilities. Right: using data. Darker colours indicate lower errors.

simulation was often blocked in a livelock: the weight function associated to the transition with label `Leucocytes` in the loop was typically returning weights close to 1 in most of process states, due to the history being included in the process state. This causes a loop to be infinitely repeating. This issue is certainly due to the nature of the method to synthesize the weight function, namely the use of logistic regression: it is likely the case that, because logistic regression is only able to synthesize weight function of a certain nature, the synthesis is unable to find a suitable weight function for transition with label `Leucocytes`. Thus, we decided to employ a binary history, namely that only accounts whether a transition has fired or not in the past, independently of the number of times. The configuration using only binary history outperforms the baseline of 6%. However, the results with the configuration with both data and binary history states are similar to those using only data: this could be due to correlations between the features.

To investigate which aspects of the simulation are improved in the best configuration for each case study compared to the baseline, we computed for each activity pair (a_i, a_j), the probability of a_j executing immediately after a_i, given that a_i has been executed. This probability has been computed as the ratio between the number of times a_j immediately follows a_i and the number of times a_i is executed: $|a_i \rightarrow a_j|/|a_i|$. When a given activity $\overline{a}$ is the first of the trace, we have a fictitious previous activity *start* to indicate this, namely a pair $(start, \overline{a})$. Similarly, we introduce a fictitious activity *end*. We then compared these probabilities with those from the real event log. Given any immediately follow relation $a_i \rightarrow a_j$, we computed the absolute error as $\left|p_{ij}^{real} - p_{ij}^{sim}\right|$, where p_{ij}^{real} and p_{ij}^{sim} are the probability described above for the real and simulated logs, respectively.

Figures 5 and 6 illustrate the absolute errors for the two cases. In particular, in the Road-Traffic Fines case study, our approach seems to work better when `Insert Fine Notification` and `Add penalty` are executed. Indeed, it can be noticed that the probabilities of `Add penalty`, `Send for Credit Collection` and `end` immediately occurring after `Insert Fine Notification` are more similar to the real ones using our approach, since the error is closer to zero. Similarly for `Add penalty`.

In the Sepsis case, our method is more accurate after the execution of `ER Registration` and `ER Triage`, and in the final activities: looking at the cells (`ER Registration`, `ER Triage`), (`ER Triage`, `ER Sepsis Triage`), and the cells of the `end` column, errors are smaller using data than using simply branching probabilities.

There are some cases in which the error is greater with our approach than with the basic one. This is probably due to the monotonicity of the logistic regression, which cannot control the loop parts of the model. However, the overall error decreases with our method.

5 Related Works

The determination of the activity probabilities in simulation model is related to stochastic process discovery [3,15,10]. However, they do not focus on building simulation models, nor do they verify whether or not their approaches are beneficial to increase the accuracy of business simulation models. In particular, Burke et al. use a similar concept of weights to determine probabilities [3], but, instead of using a more elaborated process state, they only consider the occurrences of previous activities, which our experiments show to not always be beneficial. Mannhardt et al. also use weights to determine probabilities [3], but they only focus on process attributes, which is similarly not always beneficial for more accurate simulation models. This paper brings together the works by Burket et al. and by Mannhardt et al. allowing both process-state abstractions, their combination, and also richer abstraction (e.g., includining resource and time information).

Related work also focuses on Markov-based stochastic models [2,16]. This class of models is, however, unable to feature concurrency, data and silent transitions, which are all crucial aspects of business process models, including when used for simulation.

On the other hand, several studies have been conducted in the simulation domain to discover and improve simulation models. However, none of these works have focused on discovering accurate activity firing probabilities. In [12], Pouarbafrani et al. proposed a framework with a simulation tool for enriched process trees, and in [14], the authors used historical execution data to provide simulation models. However, in these works, the probability of an activity occurring is not dependent on the data states. Camargo et al. proposed hybrid approaches using Deep Learning techniques to generate simulated logs in [6,7], but the control flow depends on constant branching probabilities that do not consider the

data and history. In [4,5], Simod has been introduced as a framework for the automated discovery of simulation models, but the activity firing probabilities are again based on branching probabilities. GenCPN [13] was proposed as a tool that extracts process parameters, including constant branching probabilities, directly from an event log and uses Process Mining techniques to convert them into a CPN model. However, these simulators use constant branching probabilities, which are not dependent on process states.

6 Conclusions

Business process simulation is a flexible approach to analyze operational processes, and, when different sorts of issues are observed, to evaluate alternative scenarios wit the aim to overcome them. The advantage of business process simulation is that the evaluation is performed through the digital twin of the actual process, and hence these alternative scenarios are assessed without risking them in real production environments. Of course, the requisite of valuable applications of process simulations is that this digital twin is an accurate representation of the real process. The digital twin is modelled through a process simulation model, which hence needs to be accurate in order to profit from the advantage of simulation.

A process simulation model consists of the actual model of the process to be simulated that is extended with the run-time characterization of the process (case arrival rate, resources and roles, activity durations, etc.). While a large body of research has recently been carried out to create accurate simulation models, an accurate stochastic modelling of activity firings in these models has been totally overlooked (cf. Section 5): it is simply assumed that these probabilities can be well determined by looking at the frequency of occurrence of these activities in the log (e.g., the so-called branching probabilities). Unfortunately, this is a coarse determination because the activity firing probabilities usually depend on characteristics of the single process instances that are simulated, and these characteristics might significant change from instance to instance. For example, the chance of executing the activity to reject a loan application might depend on the amount requested and/on the number of further clarification requests.

This paper aims to assess whether our intuition that a more accurate modelling of the activity firing probabilities passes on having them depend on the process state, which models the characteristics of the cases. To answer this question, we compute the weights of activities as function of the process state, where these weights are proportional to the activity firing probability. The technique has been operationalized using stochastic labelled Data Petri nets, a simple language, yet sufficiently expressive to model weights, probabilities, and process states. Each Petri-net transition is associated to a weight function: weights are then computed by evaluate the weight functions on the current process state.

Since the process state is in fact an abstraction, we make it customizable as function of the process to simulate. Indeed, these probabilities are possibly linked to different process characteristics, depending on the process to simulate. We

conducted experiments with two different process and five alternative process-state definition to answer our business questions. The results confirm that our weight-based estimation of activity-firing probabilities allowed us to build more accurate simulation models, and that the best process-state abstraction changes with the process to simulate.

Currently, weight functions are synthesized using logistic regression, which imposes a certain shape of the function. This has also caused the livelock problem in simulation discussed in Section 4.3, because it was not possible to synthesize a suitable weight function when the process state includes the history. As a future work, we aim to replace the synthesis based on logistic regression with alternative synthesis methods, which possibly can return better approximations of the real weight function. Also, we aim to asses how fitness and precision of the process models influence the quality of the simulations.

Acknowledgement. The research is financially supported by MUR (PNRR) and University of Padua, by the Department of Mathematics of University of Padua, through the BIRD project "Data-driven Business Process Improvement" (code `BIRD215924/21`), and by the "Smart Journey Mining project" funded by the Research Council of Norway (grant no. `312198`).

References

1. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining Knowl. Discov. **2**(2), 182–192 (2012)
2. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
3. Burke, A., Leemans, S.J.J., Wynn, M.T.: Stochastic process discovery by weight estimation. In: Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 406, pp. 260–272. Springer (2020). https://doi.org/10.1007/978-3-030-72693-5_20
4. Camargo, M., Dumas, M., González, O.: Automated discovery of business process simulation models from event logs. Decision Support Systems **134**, 113284 (2020). https://doi.org/10.1016/j.dss.2020.113284
5. Camargo, M., Dumas, M., Rojas, O.G.: Simod: A tool for automated discovery of business process simulation models. In: Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019. CEUR Workshop Proceedings, vol. 2420, pp. 139–143. CEUR-WS.org (2019), `https://ceur-ws.org/Vol-2420/paperDT5.pdf`
6. Camargo, M., Dumas, M., Rojas, O.G.: Discovering generative models from event logs: data-driven simulation vs deep learning. PeerJ Comput. Sci. **7**, e577 (2021). https://doi.org/10.7717/peerj-cs.577
7. Camargo, M., Dumas, M., Rojas, O.G.: Learning accurate business process simulation models from event logs via automated process discovery and deep learning. In: Advanced Information Systems Engineering - 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6-10, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13295, pp. 55–71. Springer

(2022). https://doi.org/10.1007/978-3-031-07472-1_4, https://doi.org/10.1007/978-3-031-07472-1_4
8. Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 360, pp. 127–143. Springer (2019), https://doi.org/10.1007/978-3-030-26643-1_8
9. Mannhardt, F.: Multi-perspective process mining. Ph.D. thesis, Mathematics and Computer Science (Feb 2018), proefschrift
10. Mannhardt, F., Leemans, S.J.J., Schwanen, C.T., de Leoni, M.: Modelling data-aware stochastic processes - discovery and conformance checking. In: PetriNet 2023. Springer (2023), https://leemans.ch/publications/papers/pn2023mannhardt.pdf, to Appear.
11. Mannhardt, F., de Leoni, M., Reijers, H.A.: The multi-perspective process explorer. In: Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015), Innsbruck, Austria, September 2, 2015. CEUR Workshop Proceedings, vol. 1418, pp. 130–134. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1418/paper27.pdf
12. Pourbafrani, M., van der Aalst, W.M.P.: Interactive process improvement using simulation of enriched process trees. In: Service-Oriented Computing – ICSOC 2021 Workshops. pp. 61–76. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14135-5_5
13. Pourbafrani, M., Balyan, S., Ahmed, M., Chugh, S., van der Aalst, W.M.P.: GenCPN: automatic CPN model generation of processes. In: Proceedings of the ICPM Doctoral Consortium and Demo Track 2021 co-located with 3rd International Conference on Process Mining, ICPM Doctoral Consortium / Demo Track 2021, Eindhoven, The Netherlands, November, 2021. CEUR Workshop Proceedings, vol. 3098, pp. 23–24 (2021), http://ceur-ws.org/Vol-3098/demo_192.pdf
14. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Supporting automatic system dynamics model generation for simulation in the context of process mining. In: Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings. Lecture Notes in Business Information Processing, vol. 389, pp. 249–263. Springer (2020). https://doi.org/10.1007/978-3-030-53337-3_19
15. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. Lecture Notes in Business Information Processing, vol. 171, pp. 15–27. Springer (2013), https://doi.org/10.1007/978-3-319-06257-0_2
16. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. **54**, 1–14 (2015), https://doi.org/10.1016/j.is.2015.04.004
17. Witten, I.H., Frank, E., Hall, M.A.: Data mining: practical machine learning tools and techniques, 3rd Edition. Morgan Kaufmann, Elsevier (2011)