Steady State Estimation for Business Process Simulations*

Mahsa Pourbafrani [1], Niels Lücking [2], Matthieu Lucke [2], and Wil M. P. van der Aalst [1,2]

[1] Chair of Process and Data Science, RWTH Aachen University, Germany
{mahsa.bafrani,wvdaalst}@pads.rwth-aachen.de
[2] Celonis, Munich, Germany, {niels.lucking, m.lucke}@celonis.com

Abstract. Simulation is a powerful tool to explore and analyze business processes and their potential improvements. Recorded event data allow for the generation of data-driven simulation models using process mining. The accuracy of existing approaches, however, remains a challenge. Various efforts are being made to improve the quality of the used data and techniques, such as extracting detailed resource performance. One of the least addressed challenges is the initial state of the simulation run. Starting from a steady state has been considered in simulation in other fields. In current process simulation approaches, the executions mostly start from an empty state. This assumption leads to initialization bias, or the startup problem, which has an impact on the early results and limits the types of analysis that can be performed. In this paper, we propose an approach to estimate a steady state of simulation models, which enables the generation of more realistic simulation results. The evaluation using real-world and synthetic event data shows the requirements for and advantages of starting from representative steady states in process simulations.

Keywords: process mining · data-driven simulation · event logs · steady-state simulation · discrete event simulation

1 Introduction

Process mining analyzes event data to provide insights into the processes, such as running process models and their conformance and performance behavior. Simulation models are used to generate future results and outcomes for processes. The insights provided by process mining can be used to simulate how the processes will continue and what the impact of the changes will be [1]. The captured events including the process instances, performed actions, and their timestamps, in event data, are the starting point for forming data-driven simulation models. Simulation models should be able to reproduce the same outputs

* Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC-2023 Internet of Production – 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. We thank Celonis for supporting the project.

as the input event data. However, due to various challenges such as data quality, models cannot fully capture the complexity of the real world, e.g., external factors and resource behaviors.

One of the open challenges is the starting point of process simulation models. Current data-driven process simulation approaches mostly start from an empty state. As a result, if they manage to reach a steady state, it takes time, i.e., a warm-up period is required while simulating. The assumption of an empty state at the start of the simulation can prevent accurate results from being obtained. In particular, simulation may never reach a steady state in the case of an unstable process with the incorrect start point. When analyzing event data of processes, one typically considers only complete cases. If the period in which event data are collected is of the same order of magnitude as the average flow time (e.g., months), then this creates a misleadingly low load. However, it is also difficult to use incomplete cases. Therefore, it is important to estimate a representative *steady state* and start simulating from this point. Short-term simulation and the accuracy of simulation results share common aspects, such as loading a representative initial state and starting simulation from a non-trivial initial state [15].

In this paper, we propose an approach to estimate steady states of processes using their event logs, which can be loaded into the simulation as the starting point. The simulation engine is considered a black box, since the focus is on the illustration of the effect and necessity of having steady states in process simulation models. We follow three main objectives while designing the approach: (1) practical components of the approach should be general, not specific to any simulation engine but applicable to common data-driven simulation approaches, (2) the time patterns in event data should be captured while discovering steady states, as there might not be a real steady state, e.g., concept drift, and (3) practical component of the approach must be efficient since simulation results might ultimately converge toward the same state as one that starts from a cold start. We evaluated our approach in estimating steady states of processes for simulation purposes using both real and synthetic logs.

Throughout the paper, a *state* of a process is an event log at a specific point in time, including the events that started before but were not finished by that time. A *start state* of a simulation is the state from which the execution will start. A *steady state* is a state of a process that shows the common behavior of the process w.r.t. different aspects, e.g., the number of cases in the process. *warm-up* period refers to the period of time that a simulation started from a cold start needs to reach a steady state of the process. The start state of a simulation can be an empty state (also referred to as a *cold start*) or a given (discovered using event data) state.

The remainder of the paper is as follows. The related work is introduced in Section 2. We elaborate on the motivation using the running example in Section 3. Preliminaries are introduced in Section 4. We explain the approach in Section 5 and evaluate it in Section 6. Section 7 concludes this work.

2 Related Work

Simulation of operational processes used to be a complex task for an expert due to the design of the process model, estimating the parameters, and providing simulation configurations [16]. The increasing availability of data and advances in process mining have led to semi-automatic approaches, where parts of the simulation model and parameters are mined and others are based on the user's input [1], e.g., [14]. This is the current setting in most academic and commercial tools, e.g., Celonis[3].

Most research efforts are currently focused on improving simulation quality, which still struggles to represent reality enough to be widely applicable. There are interactive approaches, such as [11], that propose using multiple metrics, such as stochastic conformance checking, to measure the quality of simulation results more accurately. Recent work, such as [6], focuses on the resource aspect to generate a more accurate simulation by considering multitasking and resource profiles. Another example is considering factors such as activities and external delays in generating simulation models [3]. Techniques such as those described in [2] employ hyper-parameterization to iteratively search for the set of parameters w.r.t. more accurate simulation results. In [12], a survey of data-driven simulation approaches in process mining w.r.t. user information and insights from event data has been performed. It includes a review of current data-driven simulation approaches in process mining, their challenges, and their limitations.

Aside from the current focuses and challenges in data-driven simulation approaches in process mining, steady states and starting points of simulations are still open questions. For simulations, it is known that the ideal starting state must be close to the steady state [7]. In [10], a survey of steady-state approaches for the queuing system is proposed, where the focus is on statistical analysis rather than data-driven approaches. Tail management is the default solution for the warm-up issue, as it is general and simple. Most research efforts have been focused on determining the length of the initial transient [5]. Determining the warm-up period is also one of the most common techniques in simulations for reaching a steady state of a system [9]. The general categorization of these methods is *graphical, statistical, and heuristic.* For instance, one of the most commonly used techniques to graphically specify the warm-up period is Welch's Method [8]. The presented approach in [13], which is based on time series analysis, is also a sample of statistical analysis methods for determining the warm-up period. An example of a heuristic technique is [4], where the rule is "truncate a series of measurements until the first of the series is neither the maximum nor the minimum of the remaining sets". However, these approaches do not rely on a system's historical data.

There are approaches to use steady states in specific fields (e.g., [17]), but they are also dependent on domain knowledge. The proposed techniques for estimating the steady state in general-purpose simulations are generally not based on system data and neither consider the specific attributes, e.g., case attributes

[3] `https://www.celonis.com`

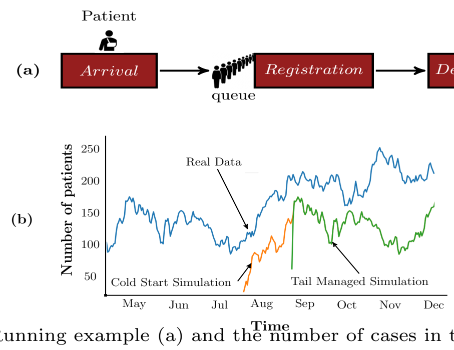

Fig. 1: Running example (a) and the number of cases in the process (b).

in the process context. The concept of starting business process simulation in its current state for short-term simulation has been proposed in [15]. In this setting, the current state is provided by the information system used. In [15], YAWL provides this state, and simulation is used to do a "fast forward" in time. In conclusion, process simulation approaches have mostly either not explicitly considered the steady-state situation or have not used event logs. Our practical components for a steady-state start can also be used for current state start and are less restrictive.

3 Running Example

Consider the simple example of the registration process in an emergency room department (Figure 1 (a)). Patients arrive and proceed to the registration desk, where they queue until a staff member is free to process their case. The nurse evaluates the severity and registers them, and indicates their next steps. In this example, a patient (case) can be in three locations, either being processed in an activity, traveling between activities or queuing if no nurse (resource) is free.

Before we start modifying the simulation to explore changes, we first simulate the as-is situation to verify that our model correctly reflects reality. We compare the simulation results by looking at the number of cases in the process, hence the number of patients in the emergency arrival room. Figure 1(b) shows the real numbers over time, compared to a cold start simulation, and a tail-managed simulation, i.e., the initial simulation results are not considered (warm-up period). A simulation starts in an empty state with no patients. As patients are generated, the simulation fills up over time and eventually might reach equilibrium. The issue is that a hospital is never empty. This initial warm-up phase biases the simulation toward wrong conclusions. The ideal solution would be to start with the correct number of patients. The state-of-the-art solution is tail management; a longer simulation where the beginning is discarded. However, the discarded part represents misspent time and simulation resources. Furthermore, tail management raises the issue of determining the length of the warm-up period and how to estimate the cut position [9].

4 Preliminaries

In this section, we introduce process mining concepts, define an event log, and continue by elaborating on the simulation concepts used in this paper.

Process Mining The executed activities for specific process instances at a specific point in time are the events that are captured in the form of event logs in the context of process mining. These events are recorded with associated information in an event log.

Table 1 is the hospital event log for the running example with attributes Case ID, Patient Name, Activity, Resource, and Activity Completion Time. Each row represents a unique event, and the columns contain the associated information. We define the used form of event log in Section 5.

Table 1: A part of the running example event log.

Event	Case	Patient	Activity	Resource	Time
871	34	John	Arrival	N/A	11:42
872	78	Alex	Registration	Nessa	12:50
873	34	John	Registration	Max	12:55
874	90	Tim	Arrival	N/A	12:50
875	78	Alex	Departure	N/A	12:58

State The state of a process is determined by variables that describe it at a precise moment in time. We use event logs with a timestamp to describe the state of a process. If there are no events in the state, it is an empty state. The current state of a process is defined as the events that have begun but have not yet been completed in the current moment of time. In the running example, the state of the process at time t=12:00 includes two events with Event IDs, 872 and 873, where these two events and their corresponding attributes started before time t and have not finished by that time. In Section 5, we define states formally in Definition 2.

Simulation The simulation engine is considered to receive a state and produce another state, i.e., given an event log, generating another event log. A simulation execution (called simulation run) can be seen as a sequence of states beginning at a starting state, where the rules of the simulation engine define the succession of states. The starting state usually is an empty state (the simulation is then called a cold start simulation).

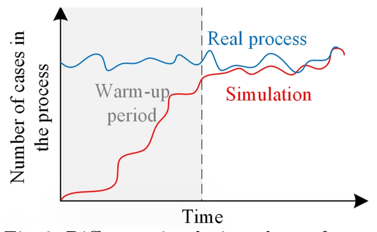

Fig. 2: Different simulation phases for an example process.

Simulations can reach such a steady state since the model remains fixed. In some cases, simulations of real-world processes do not reach a steady state due to internal and external factors, e.g., concept drift or human involvement. However, the steady state of a simulation is a useful indicator of the process's behavior. Figure 2 depicts simulation phases over time in relation to the process's real state. In an ideal situation, the simulation will eventually reach a steady state, however, in some cases, this may never happen.

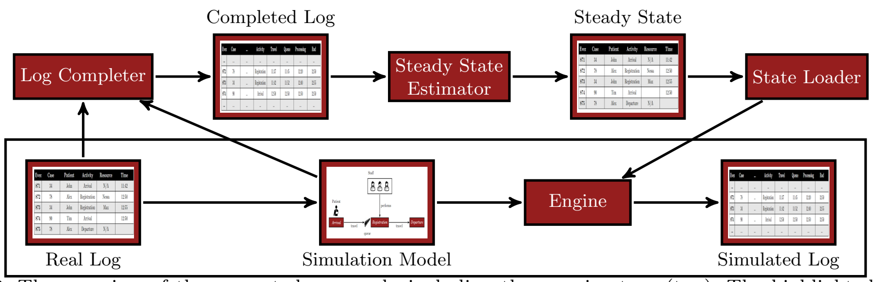

Fig. 3: The overview of the presented approach, including three main steps (top). The highlighted part (bottom) indicates the common components in data-driven simulation approaches.

5 Approach

A high-level overview of our approach is presented in Figure 3. The selected part (bottom) illustrates the common path in data-driven simulations of processes. The process model and process aspects, e.g., resource schedules, are extracted from the event log. The simulation model is then generated (designed) on the basis of the provided insights and gets executed using a simulation engine, and the generated event logs are captured. The simulated event logs are compared with the original event log to indicate the accuracy of the simulation. This can be done iteratively to find the parameters as proposed in [2]. To make our approach for estimating a steady state of simulation models generic, we consider the simulation engine and highlighted part in Figure 3 to be regarded as data-driven approaches in both academic and industrial tools.

Given an event log to create the simulation model, we first discover the missing attributes (1). Then, we use time-stratified sampling to create an event log that approximates a steady state (2). Finally, we use our state loader component (3), which allows us to load any event log into a simulation as a start state, to start the simulation from a steady state, see Figure 3 (top).

Approach Settings and Assumption Note that the simulation approach and the engine are considered *black boxes*. This allows us to adjust the approach for the general setting and make it applicable to the existing data-driven simulations in process mining. We only assume the simulation contains start and end timestamps. In our approach, we distinguish between the real queuing time, i.e., cases waiting due to a lack of resources, and the traveling time between two activities. Section 5.1 goes into more detail.

5.1 Event Log Completion

For an accurate steady-state estimation, we need a complete input event log with event attributes. Event logs, on the other hand, are limited in size and prone to errors and missing data. We use simulation models to complete an event log. The mined simulation model can be used in two ways. First, it specifies which attributes must be completed, i.e., which attributes in the simulation are

Table 2: The preprocessed sample event log of the running example, which is completed with the traveling and queuing time.

Event	...	start	Travel	Queue	Processing	End
...	...	...	...	...	...	...
872	...	11:35	11:37	11:45	12:20	12:50
873	...	11:37	11:42	11:52	12:40	12:55
874	...	12:50	12:50	12:50	12:50	12:50
...	...		...	...	...	...

relevant. Second, the simulation model can be used to quickly complete existing event logs. Every attribute in the simulation is based on either a situation-independent function (such as sampling from a probability distribution) or a situation-dependent function (e.g., queuing time depending on the number of cases queuing). For completion, we reused the situation-independent functions. The situation-dependent functions imply that the simulation model creation has completed the log, which we will also reuse. Table 2 demonstrates the event log completion provided by two methods for our running example.

For our hospital example, the given event log only contained one timestamp, complete timestamps. As our simulation distinguishes travel time, queue time, and processing time (e.g., domain knowledge and user input), those attributes need to be completed, so the position of cases can be identified for creating a steady state. The simulation provides probability distributions for travel and processing times (situation-independent), and assumes that the start time of an event is the end time of the previous case event. Queue time (situation-dependent) can be inferred as the remaining time. Through the reuse of the simulation model, we thus complete our event log, as shown in Table 2.

Definition 1. *(Completed Event Log) Let $\mathcal{E}$ be the universe of event identifiers, $\mathcal{N}$ be the universe of attribute names, and $\mathcal{V}$ be the universe of attribute values. $L = (E, N, f)$ is a completed event log, where $E \subseteq \mathcal{E}$, $N = \{cid, act, travel, queue, processing, start, end\} \subseteq \mathcal{N}$, and $f : E \times N \to \mathcal{V}$ is a function that retrieves values of event attributes. We denote $\mathcal{L}$ as the universe of event logs. For an event $e \in E$ and attribute $n \in N$, $f(e, n) = \perp$, if attribute n is undefined for event e.*

In our case, a completed event log includes travel time, processing time, queuing time, start time, and end time, Definition 1. Note that more attributes, if presented in the event log, such as age of patients, treatments, or resources, can also be included in the set of attributes.

5.2 Steady-State Estimation

In this section, we elaborate on our novel approach for estimating a steady state of processes using their event logs for simulation purposes. We first define the state notation and later apply the designed steps to estimate steady states. The input is assumed to be the completed event log which notably contains start

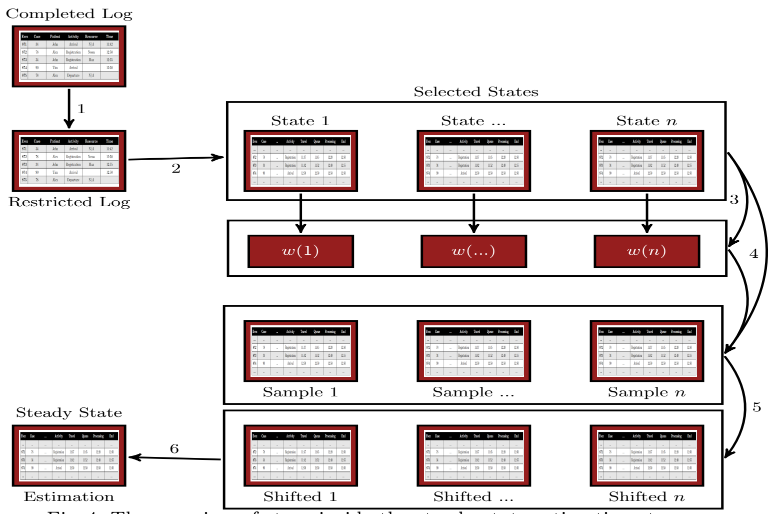

Fig. 4: The overview of steps inside the steady state estimation step.

and end timestamps. The state of a process describes the process at a precise moment in time, i.e., given an event log describing the execution over time of a process, the state at any moment in time is the slice of the event log with events currently in the process (Definition 2).

Definition 2. *(State) Let* $T \in \mathbb{R}^{+} \cup \{0\}$ *be the set of timestamps and* $L = (E, N, f)$ *be a completed event log. The* ***state*** *at time* $t \in T$ *is the sub-event-log with events active at time t, i.e.,* $L_t = (E_t, N, f_t)$*, where* $E_t = \{e \in E | f(e, start) \leq t \leq f(e, end)\}$ *and* $\forall_{e \in E_t} \forall_{n \in N} f_t(e, n) = f(e, n)$.

The state of the process at time $t = 12:00$ in Table 2 includes events 872 and 873. Six main steps are defined for estimating a steady state from an event log. The steps for estimating a steady state from a completed event log are shown in Figure 4, along with their relationships and dependencies.

Step 1 - Log Restriction The first step is for the user to limit the input event log to be used for approximation. Only relevant parts should be considered, and an unnecessarily long event log should be avoided for efficiency. This log restriction is presented using Definition 3.

Definition 3. *(Log Restriction) Let* $L = (E, N, f)$ *be a completed event log and* $[t_{start}, t_{end}] \in T$ *be a time duration. The restricted event log is* $L' = (E', N, f')$*, where* $E' = \{e \in E | f(e, start) \geq t_{start} \wedge t_{end} \geq f(e, end)\})$ *and* $\forall_{e \in E_t} \forall_{n \in N} f'(e, n) = f(e, n)$.

In the hospital example, having event logs from the last 10 years. The target steady-state approximation is a steady state of the recent hospital process. However, the hospital has evolved over time, older data might represent the hospital when it was a different process. The user thereby provides his estimate of which part of the recorded data is relevant through a time window parameter. If all the data is relevant, smaller subsets can be used for efficiency nonetheless. Here, we use the last three months, [23 May–23 July]. Note that the need to perform these steps is determined by the duration of the event log and the user domain knowledge.

Step 2 - State Selection Given that we have restricted the event log to parts relevant to the current process, we further restrict it to the states relevant to the simulation starting time, as presented in Definition 5. This allows us to account for the potential patterns in the process. Definition 4 is used to return a set of simulation timestamps in which the states are selected.

Definition 4. *(Simulation Timestamps) Let* $[t_{start}, t_{end}]$ *be a time duration, where* t_{start} *and* $t_{end} \in T$*,* $t_{sim} \in T$ *be the simulation start timestamp, and* $\delta \in T$ *be the duration of a pattern. Function* $tp \in T \times T \times T \times T \rightarrow P(T)$ *returns the set of simulation timestamps such that* $tp(t_{start}, t_{end}, t_{sim}, \delta) = \{t \in T | \exists_{m \in \mathbb{N}}\ t = t_{sim} + m.\delta \wedge t_{start} \leq t \leq t_{end}\}$

Definition 5. *(State Selection) Let* L' *be a restricted event log in time duration* $[t_{start}, t_{end}] \in T$*,* $t_{sim} \in T$ *be the simulation start timestamp,* $\delta \in T$ *be the duration of a pattern.* L'_t *is a selected state at time* $t \in TP$*, where* $TP = tp(t_{start}, t_{end}, t_{sim}, \delta)$*. We denote* $ST = \bigcup_{t \in TP} L'_t$ *to be the set of the selected states.*

For instance, in the running example, consider that the simulation starts at 12:00 on Monday, July 24, and there is a weekly pattern within our previously selected range of 3 months. The number and types of patients highly depend on the time and weekday (proportionally more emergencies at night, alcohol intoxication on the weekend, etc.). Mondays at noon are more similar to Mondays at noon in different weeks than other moments. We therefore set a $\delta = 1$ *week* parameter. If no pattern is observed, all moments are equally relevant, and any random δ can be used.

Step 3 - Recency Weight The relevance of the selected relevant states to the process's steady state is not equal. As the process evolves over time, older states may be less informative than current ones. A recency weight function that assigns a timestamp a relevance score based on how recently it occurred in relation to the simulation start time is provided for the user as an estimate. This step is considered to give the user the option to increase the influence of the process' more recent states on the simulation.

An estimate in the form of a recency weight function assigns an importance score to a timestamp based on its recency relative to the simulation start time. Based on that, we derive a weight function in Definition 6 for states (normalized to sum to 1), which determines their impact on the average.

Definition 6. *(Recency Weight) Let $r \in T \to [0,1]$ be the time recency weight function that assigns a weight score depending on the distance to simulation start time t_{sim}, i.e., $r(t_{sim} - t) \in [0,1]$, and TP be the set of simulation timestamps. For any $t \in TP$, $w(L,t) = r(t_{sim} - t)/\sum_{t' \in TP} r(t_{sim} - t')$ is the normalized state weight.*

In the running example, we observe a slow and steady increase in the number of patients. While the data from previous months is still relevant, the data from this month should have more impact. For the state L_t, function $r((t_{sim} - t)) = 1/((t_{sim} - t).weeks + (t_{end} - t_{start}).weeks)$, t_{sim} is the start of the simulation, $\delta.weeks$ returns the number of weeks in a duration, and t_{start}, t_{end} are the start and end of the restricted event log. As a result, Monday, July 17, has a normalized weight of 0.11; Monday, July 10, has a normalized weight of 0.10; and so on until Monday, May 23, has a normalized weight of 0.06.

Step 4 - Weighted Sampled State With the selected states and their importance, we can now build a representative average by sampling the states. We bias the sampling by assigning a weight to each event in a state as the selection probability using Definition 7.

Definition 7. *(Weighted Sampled State) Let $L_t = (E_t, N, f_t)$ be a selected state, w be the state weight recency function, and $\rho : 2^E \to 2^E$ be a function that randomly selects a subset of events, i.e., given $E_1 \subseteq E_t$, $\rho(E_1) \subseteq E_1$, where each event has probability $w(L_t)$ of getting selected. The weighted sample of L_t is $L^s_t = (\rho(E_t), N, f_s)$, where $\forall_{e \in \rho(E_t)} \forall_{n \in N} f_s(e,n) = f(e,n)$.*

In the example hospital, with the previously defined weight, around 11% of events from Monday, July 17, 12:00 will be selected, and 6% of events from Monday, May 23, 12:00.

Step 5 - Sample Shifting The selected events from the different states do not form a state together, since they originate from different times. To build one consistent state, we first shift their positions at the simulation start time to be the same w.r.t. the sampled moment. Definition 8 is designed to shift the samples.

Definition 8. *(Sample Shifting) Let $t_{sim} \in T$ be the simulation start time and $L^s_t = (E_s, N, f_s)$ be the sampled state. The shifted sampled state to the start of simulation is $L^s_{t \to t_{sim}} = (E_s, N, f^{t_{sim}}_s)$ such that for the two attributes $start, end \in N$, $\forall_{e \in E_s} f^{t_{sim}}_s(e, start) = (t_{sim} - f_s(e, start)) + f_s(e, start)$ and $f^{t_{sim}}_s(e, end) = (t_{sim} - f_s(e, end)) + f_s(e, end)$, for $n \notin \{start, end\}$, $f^{t_{sim}}_s(e,n) = f(e,n)$.*

In our example, Table 2, assume event 872 was selected within the state on July 17 at 12:00 w.r.t. this state, and the patient was queuing for another 20 minutes. We shift the timestamps so that the event has now started queuing on July 24 at 11:45 and will do so until July 24 at 12:20.

Step 6 - Sample Merging We finally merge the samples into a new state using Definition 9.

Definition 9. *(Sample Merging) Let TP be the set of simulation timestamps. $L_S = \bigcup_{t \in TP} L^s_{t \to t_{sim}}$ is the set of merged states.*

In Definition 9, by merging the states, the tuple of merged events, attributes, and functions of the states is created. For our running example, we created an event log that is also a state, since all events are active at simulation time. This event log is an average representation of Monday 12:00 from the last three months, with a stronger influence from recent data.

5.3 State Loader

The final step loads a state as the starting state of the simulation. We omit details, as this step depends strongly on the simulator. The intuition is that since our events are all complete, we know their position w.r.t. the start timestamp at each moment. For example, event 872 from Table 2 for a simulation starting at 12:00 should be loaded into its activity with 20 minutes remaining. This can be done by reusing the functions the simulation engine uses to advance cases through the system.

6 Evaluation

To evaluate our approach, we designed a framework to compare the simulation results in three different settings. The goal of the evaluation is to assess the ability of our approach to capture the accurate steady states of processes using both real and synthetic event logs, as well as the use of such an estimation as a simulation starting point. We simulate the three scenarios for each event log, starting from a cold start, using tail management, and starting from the estimated steady state. The results enable us to compare and demonstrate the impact of our approach in practice. Furthermore, we discuss and illustrate the various situations in which steady-state estimation is applicable and should be considered. As discussed in Section 5, the simulation engine and the quality of the simulation results are not the focus of the evaluation. Owing to privacy considerations, the public sharing of both the data and the implemented codes within commercial tools are limited. Nonetheless, the evaluation details of the presented datasets, including performance metrics, are reported in the evaluation section.

Evaluated Event Logs The simulation models of the two real-world event logs were created and validated jointly by process analysts and the companies. We reuse them to show that our steady-state start presents a significant improvement due to the slow convergence of the process in one situation, and that the simulation with a steady state is not necessary given the process characteristics in the other one. Our synthetic hospital model is designed to show a different

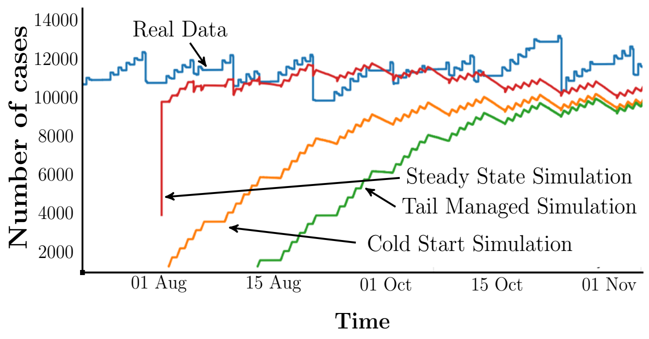

Fig. 5: The comparisons of the simulated number of cases in the process for the medical company.

use case, instability, i.e., the incoming cases exceed the capacities of the process. In this case, a steady-state start is an improvement and the only viable solution.

The number of cases in the process is the primary metric used to represent simulations. We detail whether other metrics, such as case throughput time, are affected in the various data sets. The longer the case throughput time, the more effective the use of steady-state estimation is expected to be. If the throughput time is short, the simulation with a cold start also reaches the steady state quickly, and the tail-management strategies appear plausible.

6.1 Real-world Event Log

We tested our method on two real-world event logs with different characteristics, such as different throughput time of cases within the process. Because of privacy concerns, the event logs have been anonymized.

Medical Company Event Log The throughput time for a single case in the medical company process is around 24 days. The process has a high case load (10000–12000 cases at all times). Most of the time is spent traveling in between activities, time that can be attributed to waiting for external processes (e.g., waiting for customer payment). Furthermore, the process is very stable, with a consistent case load throughout the years. Figure 5 shows the results for the real medical invoice process.

Cold start simulation performs poorly under these conditions. Due to the high case load and long throughput times, the simulation converges slowly toward a stable result. It takes 60 simulated days for the cold start simulation to stabilize. The tail-managed simulation must thus simulate 60 additional days to eliminate the warm-up phase. Also, the required warm-up time is only obvious in hindsight. The common tail-managed simulation uses a warm-up time of 20%, which proves insufficient. Our approach shows a striking improvement. Due to the stability of the process, almost no warm-up is required. The simulation is immediately in a state that is representative of the real process. Since a majority of the throughput time of cases is spent traveling between activities in the process, results of the simulation other than process load are not impacted (for example, throughput time), all simulations deliver the same results.

Sensitivity Analysis with 95 Confidence Interval and 50 Repetitions Each simulation was performed 50 times. We compute the 95 confidence interval for each moment by taking the 50 computed values for cases in the system. For each simulation, we plot the lower and upper bounds as shown in Figure 6. Because of the small size of the confidence interval in relation to the scale, the lower and upper bounds are mostly indistinguishable. The simulations remain consistent, and the current state start and steady-state start outperform the cold and tail simulations significantly. Even the worst steady or current simulation outperforms cold and tail simulations. Furthermore, the nontrivial state start adds 17.37 seconds of overhead. Event log completion, on the other hand, only needs to be done once and can be reused for all simulations. 3.14 seconds is the worst overhead for a nontrivial state start after completion.

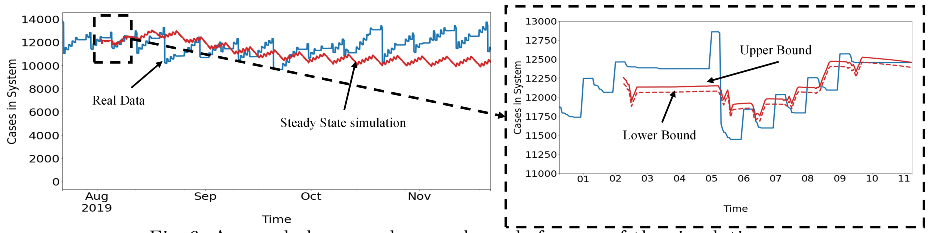

Fig. 6: A sample lower and upper bounds for one of the simulations.

Car Production Event Log Figure 7 shows the results for a car production company. The event log has a short duration (three months), and the process has a throughput time of 2 hours. As such, the initial state loaded into the simulation is processed after 2 hours. In such cases, a steady-state start is not required, but the improvement can be seen by starting from a more realistic state compared to the real process.

6.2 Running Example Event Log

We deliberately designed our running example process to have different characteristics as a showcase. The process has a lower case load and short throughput times (a few hours). However, the process is unstable. The amount of patients slightly exceeds the capacities of the hospital's registration process, resulting in a slowly increasing queue. Patients spend the majority of their time queuing.

Figure 8 depicts the results for our hospital example. The cold start simulation starts in an unrealistically empty state. It never reaches the real data because of the existing concept drift. Because the simulation starting from the

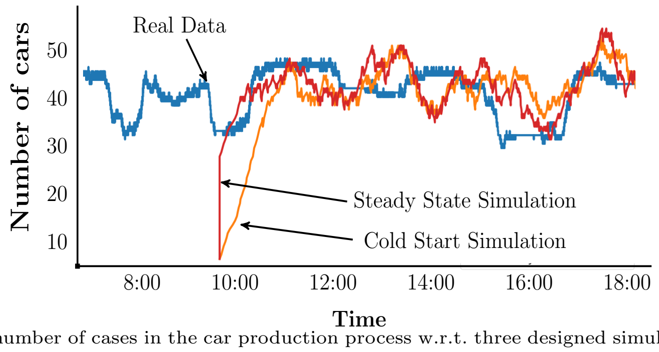

Fig. 7: The number of cases in the car production process w.r.t. three designed simulation strategies.

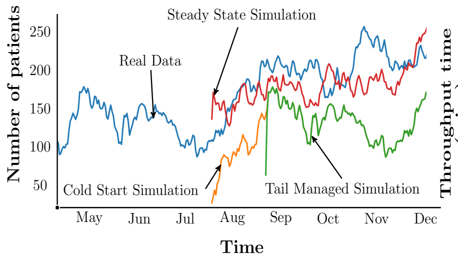

Fig. 8: The number of patients (cases) in the running example (hospital example) and three designed simulation strategies.

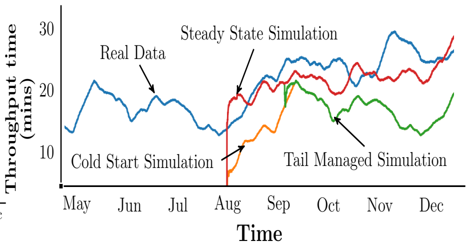

Fig. 9: The throughout time of cases in the running example (hospital example) process w.r.t. three designed simulation strategies.

cold start will be inaccurate, the tail-managed simulation will be inefficient in terms of providing more accurate simulation results. As such, it also never reaches the real process load. Our steady-state estimation is a significant improvement. Our simulation starts close to the real process and remains an improvement at all times. Figure 9 graphs the throughput time, i.e., how long it takes patients to complete the process. As this value depends on the queue time, and hence the process load, the same problem with cold start simulation can be observed, and the same improvement can be observed with steady-state start. All metrics that do not depend on process load remained identical for all simulations.

In such scenarios, the cold start simulation is the worst case. By extension, tail-managed simulation cannot solve this. Since the cold start simulation never stops warming up, no amount of cut-off will improve the result. This problem is solved by using a steady-state start. However, in our example, due to the process's instability, the approximated starting point is not as good as in the previous examples. Yet, it remains vastly superior to an empty start. Our steady-state start simulation is an accurate representation of the process. These results extend to other metrics. Because patients spend the majority of their time queuing, the number of patients correlates with the throughput time, resulting in the same issues for cold start and tail-managed simulation.

In order to assess the performance of our approach in practice, we ran the workflow several times. All simulations took approximately 30 seconds. The

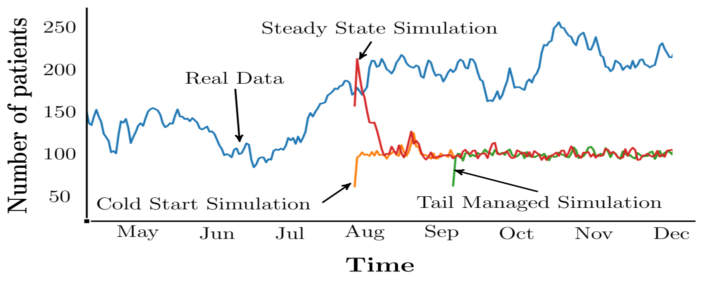

Fig. 10: The comparison of simulation results for three starting points in what-if analysis scenarios for the running example (hospital).

steady-state estimation module takes about 2 seconds on average, and the state loading module takes about 0.1 seconds.

What-if Simulation So far, we replicated the real data, and steady-state start reduced warm-up time. For what-if simulations, our approach does not aim to minimize warm-up, but make it a valuable part of the simulation instead. In a cold start simulation, the warm-up indicates the states the system goes through from an empty state until stabilization. Unless the system is actually regularly empty, this is irrelevant. In a what-if simulation (the simulation is modified to represent a change in the system), starting from an estimation of the new steady state would optimize convergence. However, when starting from the steady-state approximation of the current process, warm-up represents the transitory period that changes create until a new equilibrium is found.

In our example, the hospital aims to stop the increase in queues. They assign one additional nurse to registration, who assists patients in between activities. The simulations for this situation are shown in Figure 10. Although cold start simulation happens to converge quicker than steady-state start simulation, the latter indicates that the changes would need two weeks to break down backlog and reach the new equilibrium.

6.3 Discussion

Our goal was to demonstrate the significance of the starting point in process simulations and propose a method to estimate it using event logs. This step in the process simulation is either ignored, i.e., assumed to begin from an empty state, or in practice employs a tail management strategy, i.e., simulating from a cold start and disregarding the considered duration as a warm-up period. Patterns are one of the crucial pieces of information that should be represented in the simulation models and be reproducible, as they affect the extraction of steady states. In the case of our running example, a few example patterns are as follows. On the weekend, there are more cases of alcohol intoxication. In the winter, there are more flu cases. During the night, the proportion of severe cases is higher. We considered the existence of patterns in our approach by state selection to sample states. This allows us to include patterns in our steady states.

Our results show that simulating from steady states is effective, especially when the process has a long throughput time as well as scenarios with a long convergence time (high case load). We showed that the warm-up period for one of the event logs would be 60 days of simulation to reach the steady state. For unstable simulations (with no convergence), we are not aware of other general purpose solutions. Our work is most similar to [15], in which the simulation begins from the current state. Steady state includes more general process behavior because it is sampled across the process rather than just the current moment, e.g., the current state can be an anomaly in the process.

Estimating process states as a part of our approach is practical in different use cases. Instead of having a lot of time-dependent parameters in the simulation, chaining simulation models by using the end state of one simulation as the start state of the next is one of the effective scenarios. The difference is that only one method is required, and then any time-related change can easily be expressed. This method also scales beautifully; new elements in the simulation will be directly expressible in time.

Example of Mixed Models Using Steady States There are other scenarios in which the extracted steady states are effective. Consider an x-ray machine breakdown, creating an increasing queue. There are two potential replacements, one available quickly but slowly, and one available in two weeks but efficiently. The current simulation cannot compare these two solutions, as it can only simulate them as if they have always existed. Our steady state can then be used to simulate what would happen if we started from there. However, by chaining models, we can start from the steady state, simulate the problem scenario, and then explore the different solution scenarios. As an alternative, suppose we discover through process mining that the first and second Mondays of each month are less efficient. Adding parameters to every resource and arrival rate is cumbersome, but we can very easily chain together simulations to express this.

7 Conclusion

In this paper, we proposed an approach for estimating the steady state start point for the simulation of processes. The steady state estimator is innovative and efficient in the context of data-driven process simulation. The approach is designed to be independent of the simulator. The evaluation shows that it is efficient, as it scales only with the number of sampled moments, and the number of cases in the system. We extended it to include weights, for concept drift, as well as a curated selection of existing patterns. Since the event log is directly sampled, anything represented in the log is already included in the steady state. The idea of using non-trivial starting states still holds much untapped potential. As data collection, process mining and simulation models evolves and improve, starting from the steady or current state will open more efficient and diverse analysis potential.

References

1. van der Aalst, W.M.P.: Process Mining and Simulation: A Match Made in Heaven! In: Computer Simulation Conference . pp. 1–12. ACM Press (2018)
2. Camargo, M., Dumas, M., González, O.: Automated discovery of business process simulation models from event logs. Decis. Support Syst. **134**, 113284 (2020)
3. Chapela-Campa, D., Dumas, M.: Modeling extraneous activity delays in business process simulation. In: 2022 4th International Conference on Process Mining (ICPM). pp. 72–79 (2022)
4. Conway, R.W.: Some tactical problems in digital simulation. Management science **10**(1), 47–61 (1963)
5. Eickhoff, M., McNickle, D., Pawlikowski, K.: Detecting the duration of initial transient in steady state simulation of arbitrary performance measures. In: Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools. ValueTools '07 (2007)
6. Estrada-Torres, B., Camargo, M., Dumas, M., Yerokhin, M.: Discovering business process simulation models in the presence of multitasking. In: International Conference on Research Challenges in Information Science. pp. 381–397. Springer (2020)
7. Law, A.M., Kelton, W.D., Kelton, W.D.: Simulation modeling and analysis. In: volume 3. Mcgraw-hill New York (2007)
8. Law, A.M., Kelton, W.D., Kelton, W.D.: Simulation modeling and analysis, vol. 3. Mcgraw-hill New York (2007)
9. Mahajan, P., Ingalls, R.: Evaluation of methods used to detect warm-up period in steady state simulation. In: Proceedings of the 2004 Winter Simulation Conference, 2004. vol. 1, p. 671 (2004)
10. Pawlikowski, K.: Steady-state simulation of queueing processes: Survey of problems and solutions. ACM Comput. Surv. **22**(2), 123–170 (jun 1990)
11. 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)
12. Pourbafrani, M., van der Aalst, W.M.P.: Data-driven simulation in process mining: Introducing a reference model. In: Communications of the ECMS, Volume 37, Issue 1, Proceedings. pp. 411–420 (2023)
13. Robinson, S.: A statistical process control approach for estimating the warm-up period. In: Proceedings of the Winter Simulation Conference. vol. 1, pp. 439–446. IEEE (2002)
14. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering simulation models. Information System **34**(3), 305–327 (2009)
15. Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J.: Workflow simulation for operational decision support. Data Knowl. Eng. **68**(9), 834–850 (2009)
16. Tumay, K.: Business process simulation. In: Proceedings of the 28th conference on Winter simulation. pp. 93–98 (1996)
17. William, N., Tracy, C., Nick, B.: Improving the accuracy of random waypoint simulations through steady-state initialization. In: Proceedings of the 15th International Conference on Modeling and Simulation. pp. 319—-326 (2004)