To Aggregate or to Eliminate? Optimal Model Simplification for Improved Process Performance Prediction

Arik Senderovich[a,*], Alexander Shleyfman[a], Matthias Weidlich[b,*], Avigdor Gal[a], Avishai Mandelbaum[a]

[a] *Technion - Israel Institute of Technology, Haifa, Israel*
[b] *Humboldt-Universität zu Berlin, Berlin, Germany*

Abstract

Operational process models such as generalised stochastic Petri nets (GSPNs) are useful when answering performance questions about business processes (e.g. 'how long will it take for a case to finish?'). Recently, methods for process mining have been developed to discover and enrich operational models based on a log of recorded executions of processes, which enables evidence-based process analysis. To avoid a bias due to infrequent execution paths, discovery algorithms strive for a balance between over-fitting and under-fitting regarding the originating log. However, state-of-the-art discovery algorithms address this balance solely for the control-flow dimension, neglecting the impact of their design choices in terms of performance measures. In this work, we thus offer a technique for controlled performance-driven model reduction of GSPNs, using structural simplification rules, namely *foldings*. We propose a set of foldings that aggregate or eliminate performance information. We further prove the soundness of these foldings in terms of stability preservation and provide bounds on the error that they introduce with respect to the original model. Furthermore, we show how to find an optimal sequence of simplification rules, such that their application yields a minimal model under a given error budget for performance estimation. We evaluate the approach with two real-world datasets from the healthcare and telecommunication domains, showing that model simplification indeed enables a controlled reduction of model size, while preserving performance metrics with respect to the original model. Moreover, we show that aggregation dominates elimination when abstracting performance models by preventing under-fitting due to information loss.

Keywords: Generalised Stochastic Petri Nets, Model Simplification, Folding, Elimination, Aggregation, Process Mining

1. Introduction

Performance analysis is an important pillar of business process management initiatives in diverse domains, from telecommunication, through healthcare, to finance. Taking healthcare as an example, it involves the ability to answer questions such as 'how long will it take for a patient to get treatment?', and 'how many nurses do we need to staff to accommodate the incoming demand?' [1]. In call centers, analyzing performance drives the number of staffed agents, which correlates with vast operational costs [2]. Hence, answers to performance questions are key in running an organisation successfully and deliver value to its customers [3].

Operational process models such as generalised stochastic Petri nets and queueing networks are useful to answer the aforementioned performance questions [4, 5]. In particular, these models enable testing of re-design and improvement initiatives with respect to the as-is model. For instance, by changing staffing levels and altering the control-flow, the impact of operational changes on the performance characteristics of the process can be explored.

Process mining enables automatic discovery and enrichment of operational process models from logs, which record process executions [6]. Data-driven model discovery improves beyond the manual model elicitation in its ability to accurately reflect the executed process. However, automatically discovered models tend to incorporate infrequent process executions, which may result in over-fitting with respect to the originating log. Recently proposed discovery algorithms attempt to balance between over-fitting and under-fitting in the control-flow dimension [7, 8, 9, 10]. Yet, the question of how to balance over-fitting and under-fitting in terms of performance annotations of operational models has received little attention in the literature so far [11].

This work tackles the problem of balancing over- and under-fitting in performance-oriented models. Our method involves the automated simplification of generalised stochastic Petri nets (GSPNs) by starting with an initial GSPN discovered from a log, and applying simplification rules that generalise the model. To avoid under-fitting, we quantify the expected error incurred by every simplification, thereby linking model size and the total error in estimating the performance characteristics of a process.

*Corresponding author
Email addresses: sariks@tx.technion.ac.il (Arik Senderovich), alesh@ie.technion.ac.il (Alexander Shleyfman), matthias.weidlich@hu-berlin.de (Matthias Weidlich), avigal@ie.technion.ac.il (Avigdor Gal), avim@ie.technion.ac.il (Avishai Mandelbaum)

The paper builds upon our earlier work [11] and provides three main contributions:

(1) *Structural foldings*: We define a set of structural simplification rules for GSPNs, referred to as *foldings*. Unlike existing proposals for model simplification [12], these rules are local (affecting only a subnet of the GSPN), come with formal bounds regarding the introduced estimation error, and their applicability is identified automatically by structural decomposition of the GSPN. Foldings, as a form of simplification, may either use aggregation or model elimination.

(2) *Theoretical foundation*: For each folding, we prove that it is proper, i.e. preserving queueing stability, a key property of performance models. Further, for each folding, we provide an error bound on the bias that the folding introduces with respect to the original model.

(3) *Optimality*: We formulate model simplification as an optimisation problem that aims at attaining a minimal model for a given cost budget for the introduced estimation error. We prove that the optimisation problem boils down to the well-established *tree-knapsack* problem, which can be cast as an Integer Linear Programming (ILP) problem. This enables efficient computation of the optimal sequence of folding operations.

We evaluate our approach with use-cases from two relevant domains: healthcare and telecommunication services. Our experiments show that simplification of a GSPN discovered from a real-world log enables users to balance over-fitting and under-fitting from the performance perspective.

The remainder of the paper is structured as follows. The next section discusses the methods and challenges in performance-oriented process mining. Section 3 recalls the GSPN formalism. Foldings of GSPNs are introduced in Section 4, while Section 5 discusses how to identify applicable foldings for a GSPN. The model simplification problem and its encoding as an ILP program is given in Section 6. Evaluation results are presented in Section 7. Section 8 reviews related work, before Section 9 concludes the paper.

2. Background: Performance-Oriented Process Mining

In this section, we provide an overview of techniques for performance-oriented process mining. We start with a brief review of the use of process mining for operational analysis (Section 2.1). Then, we provide the intuition for our model simplification technique as a method for alleviating overfitting in performance analysis of business processes (Section 2.2).

2.1. Process Mining for Operational Analysis

We consider a setting in which a log L of recorded process executions is given and analysis questions regarding the performance of process execution shall be answered. Specifically, let Y be a performance measure, e.g., the total runtime of a process instance. Further, let $q(Y)$ be a performance query over Y, e.g., the expected value of Y, which we aim at answering based on L. In general, we distinguish two types of process mining techniques to quantify $q(Y)$.

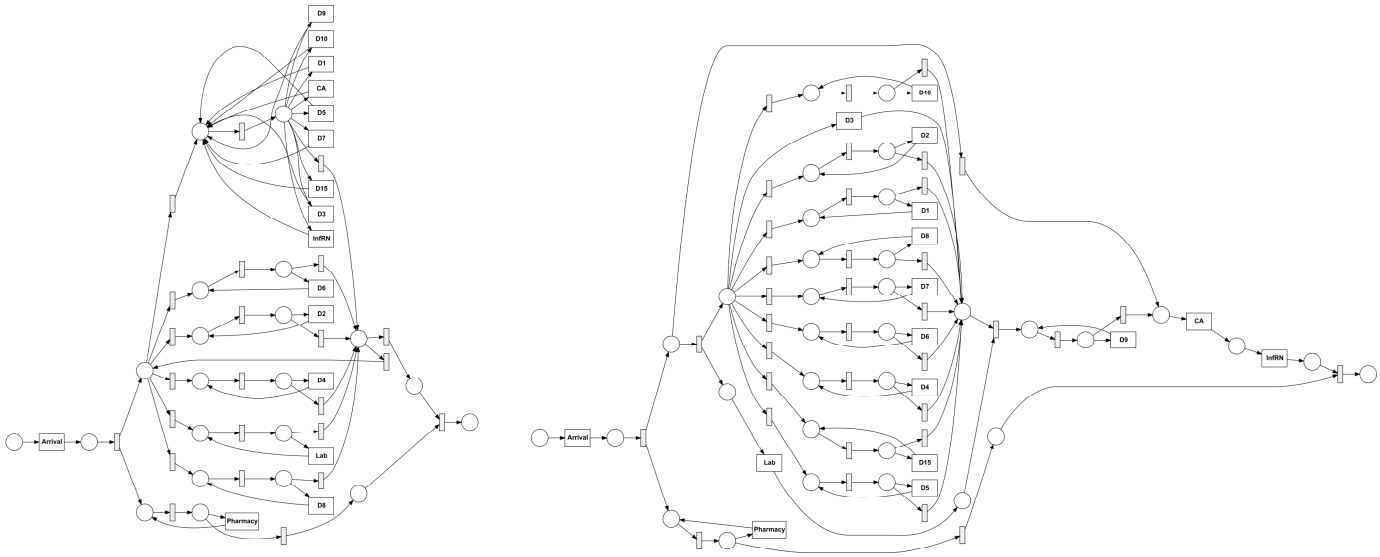
First, machine learning (ML) techniques may be exploited [13, 14, 15]. That is, process executions (including their data) are encoded as a feature vector X. Common ML methods such as regression or decision trees are used to construct an estimator $\hat{q}(Y)$ conditioned on X. While such an approach is often accurate in predicting $q(Y)$, it has two major drawbacks. Given a performance measure Z that is not directly observable in the log, one cannot quantify $q(Z)$, since ML methods require labelled observations of $q(Z)$ in the training phase. For instance, Z may be the waiting time for a specific resource. If it is not recorded in the log, $q(Z)$ must be estimated. This estimation procedure may introduce an error, which reduces the accuracy of the learning technique. In addition, exploring to-be processes and 'what-if?' analysis of current process parameters is impossible due to lack of data that describes the effect of X on Y under the new terms.

A second approach to answer performance questions is to use operational process models. Given L, operational models such as GSPNs can automatically be discovered and enriched with performance information [16, 17]. To quantify $q(Y)$, a corresponding query $q_M(Y)$ is evaluated over the model, e.g., with the help of simulation [17] or queueing theory approximations [5, 12]. A model-based approach overcomes the aforementioned limitations. It supports queries for measures that were not directly recorded in the log and enables to-be performance analysis and sensitivity analysis (e.g., by changing the control-flow and altering activity durations).

The model-based approach suffers from a major drawback, namely *over-fitting* of the estimated $q(Y)$ with respect to L [12]. ML-based methods balance over-fitting of $\hat{q}$ to L by means of model selection, which comprises two main approaches, namely regularisation, and aggregation [18]. Specifically, the former focuses on *eliminating* the least significant parts of an ML model (e.g., pruning a regression trees, subset selection, Ridge regression and Lasso), while the latter aggregate parts of the model by projecting its feature space into a smaller state-space (e.g., principal component regression, and partial least squares). An analogy to ML model selection for performance process models, does not exist.

2.2. Balancing Fitness and Generalization in Performance Process Models

We illustrate the need to balance between over-fitting and under-fitting (or generalization) using two models that were discovered from a real-world case in the healthcare domain (see our evaluation results for details). Fig. 1 depicts two process models, discovered using the Inductive Miner [19] with different noise thresholds: 0% for model (a) and 20%



(a) 0% noise filter.

(b) 20% noise filter.

Figure 1: Automatically discovered model of a hospital process.

for model (b). We observe that noise filtering balances over- and under-fitting of the control-flow regarding the log, yielding less of a "spaghetti" model when filtering more events. However, the trade-off between over- and under-fitting is not addressed for the performance perspective. Moreover, the impact of filtering a specific percentage of traces from the event log on the goodness-of-fit of the resulting model is unclear.

As we later demonstrate empirically, a principled approach based on performance-driven model simplification, in turn, alleviates under-fitting in the performance dimension, thereby significantly improving the accuracy of the resulting models. This simplification can be executed by removing parts of the model or by aggregating its sub-parts into new components.

3. Performance Analysis with Generalised Stochastic Petri Nets

We start the section with syntax and semantics of GSPNs (Section 3.1), before turning to the use of GSPNs for process performance analysis (Section 3.2).

3.1. GSPN Syntax and Semantics

Generalised Stochastic Petri Nets (GSPNs) [20] are a class of Petri nets that incorporate stochastic information on time behaviour: transitions are either *immediate*, representing atomic logical actions, or *timed*, representing units of work. Note that it is straightforward to translate GSPNs into other well-known formalisms used for operational analysis, such as queueing networks [21].

Below, we recall a notion of GSPNs that includes weights of immediate transitions, and resource capacities and expected durations of timed transitions.

Definition 1 (GSPN)**.** A GSPN is a tuple $G = \langle P, T, F, \gamma, \delta, \omega \rangle$ where:

- P is the set of places,
- $T = T_i \cup T_t$ is the set of transitions, which are immediate (T_i) or timed (T_t), respectively,
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
- $\gamma : T_t \to \mathbb{R}_0^+$ assigns capacities to timed transitions (work units per time unit),
- $\delta : T_t \to \mathbb{R}_0^+$ assigns expected durations to timed transitions,
- $\omega : T_i \to [0, 1]$ assigns weights to immediate transitions.

We refer to the tuple $\langle P, T, F \rangle$ as the *structure* of the GSPN, and to $\langle \gamma, \delta, \omega \rangle$ as its *functional component*. The set $X = P \cup T$ denotes all *nodes* and the *size* of a GSPN is defined as $|X|$. For a node x, $\bullet x = \{y \in X \mid (y, x) \in F\}$ and $x\bullet = \{y \in X \mid (x, y) \in F\}$ denote its *preset* and *postset*, respectively. Further, F^* is the reflexive, transitive closure of F.

Semantics of a GSPN are defined as a 'token game': A *marking* $M : P \to \mathbb{N}_0$ assigns to each place a number of *tokens*, thereby representing a GSPN state. A transition $t \in T$ is *enabled* in M, if all places in its preset are marked, i.e., $\forall\, p \in \bullet t : M(p) > 0$.

An immediate transition that is enabled can *fire*. Firing of a timed transition t depends on its capacity and expected duration: Once it is enabled, a single exponential clock with rate $\lambda(t) = \frac{\gamma(t)}{\delta(t)}$ is started and the transition can *fire* when the clock is elapsed. That is, we assume a single-server semantics: there is one exponential clock per enabling.

Firing a transition t in a marking M yields a marking M', such that $M'(p) = M(p) - 1$ for all $p \in \bullet t \setminus t\bullet$; $M'(p) = M(p) + 1$ for all $p \in t\bullet \setminus \bullet t$; and $M'(p) = M(p)$ otherwise. Although tokens are indistinguishable, for performance analysis, we shall assume that the tokens that enable a

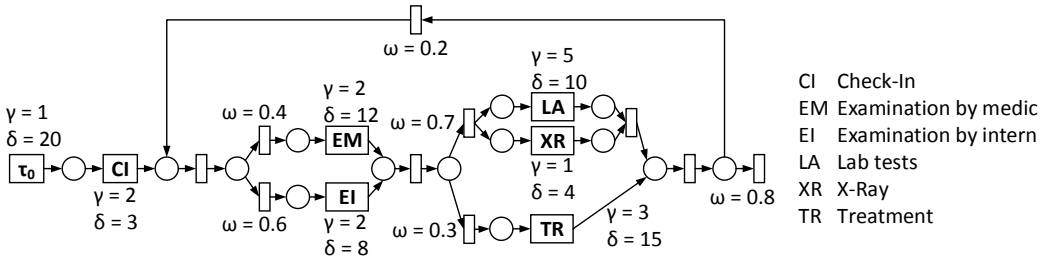


Figure 2: GSPN of a simple healthcare process. Labelled transitions are timed, unlabelled transitions are immediate (with weight $\omega = 1$, unless specified otherwise).

timed transition are selected on a First-Come First-Served (FCFS) policy. Since first-order performance measures (e.g., average waiting times and average number of tokens in a place) are indifferent to the selection policy [20], the assumed FCFS policy is indeed plausible.

Semantics of a GSPN further depends on types of transitions and their assigned rates (capacity over expected duration) and weights as follows. Let $t_1, ..., t_k \in T$ be transitions that are enabled in a marking M, i.e., they compete for firing. If transitions $t_1, ..., t_k$ are either all immediate or all timed, the assigned rates or weights determine the likelihood of each of the transitions being fired. This likelihood is defined for transition t_j, $1 \leq j \leq k$, as $\frac{\lambda(t_j)}{\sum_{i=1}^{k} \lambda(t_i)}$ (only timed transitions) or $\frac{\omega(t_j)}{\sum_{i=1}^{k} \omega(t_i)}$ (only immediate transitions), respectively. If some transitions are immediate and some are timed, the immediate transitions have priority and the likelihood model is applied only to the immediate transitions.

To exemplify the notion of a GSPN, consider a simple process from the healthcare domain, as depicted in Fig. 2. The process includes several activities. After the patient checked-in, they are examined either by a medic or an intern. Subsequently, there is a decision between diagnostics (lab tests and an x-ray are performed in parallel) or direct treatment of the patient. Afterwards, the patient may leave the hospital, or go through another loop of examination, and either diagnostics or treatment. All these activities are captured by timed transitions in the GSPN, each being assigned a capacity γ and an expected duration δ. Assuming minutes as the time scale, for instance, examination by a medic has a capacity of one and a duration of 12 minutes. This yields a rate of one over 12, meaning that we expect to be able to examine 5 patients per hour. Furthermore, we note that immediate transitions are assigned weights, which govern how often we expect patients to take a particular route through the process. Note that in our example, we normalised the weights of immediate transitions that compete for tokens from the same place. Therefore, they may be interpreted in terms of branching probabilities. For instance, when a patient is up for examination, the GSPN defines that examination will be conducted by a medic with probability 0.4, whereas an intern will take care of a patient with probability 0.6.

3.2. Process Performance Analysis

A business process is described by an *open GSPN*, which is a GSPN $G = \langle P, T, F, \gamma, \delta, \omega \rangle$ that has a dedicated timed transition $\tau_0 \in T_t$, called *arrival transition*, which represents external arrivals into the system [22]. Specifically, it holds that $\bullet\tau_0 = \emptyset$ (and for all $t \in T_t \setminus \{\tau_0\}$ it holds $\bullet t \neq \emptyset$), $\gamma(\tau_0) = 1$, and $\delta(\tau_0) = \frac{1}{\beta_0}$, so that β_0 represents the *arrival rate* of the open GSPN. In the remainder, we assume all GSPNs to be open GSPNs.

In the example in Fig. 2, we observe that there is an explicit arrival transition τ_0, with a capacity of one and an expected duration of 20 minutes. That is, the arrival rate for the process is one over 20, i.e., one patient every 20 minutes.

The arrival transition τ_0 is enabled in any marking and thus, also in the *empty marking* M_0 with $M_0(p) = 0$ for all $p \in P$, which serves as the *initial marking*. Then, the *reachability graph* of G is a graph comprising all *reachable markings*, denoted $\mathcal{R}(M_0)$, i.e., markings that can be obtained by firing of transitions of G, starting in M_0 (here, the empty marking). To perform steady-state analysis, it was shown that the reachability graph of a GSPN is isomorphic (after reduction) to a Continuous-Time Markov Chain (CTMC) [20]. The transition rates between the CTMC states correspond to the rates $\lambda(t) = \frac{\gamma(t)}{\delta(t)}$ assigned to the respective timed transitions in the GSPN. Exploiting this transformation, performance analysis of a GSPN is based on techniques of CTMC analysis: global balance equations of the CTMC are solved or, to alleviate the complexity of solving these equations, queueing theory approximations can be used.

In the analysis of a GSPN, we rely on the queueing network analyzer [23], an approximation technique that is based on a steady-state assumption. Specifically, we assume that the system is stable, i.e., the marking of a GSPN is not changing, on average, over time. When analysing time-varying systems where the number of tokens is not stable over time, one needs to fit a GSPN model per stable period. For example, when modelling a hospital process, one needs to fit a model for every time period in which the number of patients does not change 'too much' (stable). Fitting a single GSPN to the whole time period would result in an inaccurate model. An extension of our approach to time-

varying GSPNs that inherently consider time dependencies is beyond the scope of this work.

4. Simplification of GSPN

We next define a general framework for model simplification, which abstracts GSPNs by means of transformation rules (aka foldings). In the proposed framework, these foldings replace parts of a GSPN with a more compact GSPN, whereas the performance information is either *aggregated* or *eliminated*. In any case, foldings must be sound in terms of preserving performance stability. Also, we show that each folding is assigned with a cost that corresponds to the additive error due to abstraction.

In Section 4.1, we first define foldings, in general, as structural simplifications of GSPNs. We then introduce the notions of properness and cost of foldings, which enable us to characterise which foldings are reasonable for performance abstraction of GSPNs (Section 4.2). Finally, we provide the details on the functional abstraction realised by our foldings, which includes proofs of properness and the derivation of induced costs (Section 4.3).

4.1. Structural Simplification by Means of Foldings

Let $\mathcal{G}$ be the universe of GSPNs. Then, we define a folding as a structural simplification of a GSPN:

Definition 2 (Folding)**.** A folding is a function $\psi : \mathcal{G} \to \mathcal{G}$, such that for all $G \in \mathcal{G}$ it holds that $|\psi(G)| \leq |G|$.

In this work, we consider five foldings that we depict in the upper part of Fig. 3, namely a sequence folding, a race folding, a loop folding, an XOR folding, and an AND folding. These foldings relate to common behavioural structures in business process models [24].

Note that the race folding and the XOR folding relate to different semantic concepts: the former folds a net that represents resources working in parallel on jobs that arrive as tokens in the respective place. The XOR folding, in turn, relates to probabilistic selection of activities, i.e., a probabilistic selection among different timed transitions. We shall denote the five folding types – XOR, AND, race, sequence, and loop – by adding subscript to the function ψ, namely $\psi_X, \psi_+, \psi_R, \psi_\rightarrow, \psi_\circlearrowleft$, respectively.

As illustrated in Fig. 3, the structural definition of the five folding rules trivially boils down to the following. Each of the foldings takes as input a GSPN $G = \langle P, T, F, \gamma, \delta, \omega \rangle$ of one of the structures visualised in Fig. 3. Applying any of the foldings yields a new GSPN $G' = \psi(G) = \langle P', T', F', \gamma', \delta', \omega' \rangle$, where the structure $\langle P', T', F' \rangle$ is the new net comprising the new τ_0' transition (with rate β_0), and two places that are connected via a timed transition, denoted t'.

For the resulting GSPN, we also need to define the functional component $\langle \gamma', \delta', \omega' \rangle$. Since the resulting GSPN contains only a single timed transition, ω' is trivially empty. The definition of the capacity $\gamma'(t')$ and expected duration $\delta'(t')$ assigned to the timed transition t' that replaces the original GSPN, in turn, depends on the semantics of the abstracted GSPN and whether performance information is aggregated or eliminated.

4.2. Properness and Cost of Foldings

Before turning to the definition of the capacities and expected durations for different types of foldings, we review requirements on such a definition. By Definition 2, a folding operation is a contraction of a GSPN, which yields a GSPN that is equal or smaller in size. Yet, not all contractions are reasonable when aiming at improved accuracy of performance prediction. It is important that foldings preserve *stability* to ensure that the resulting model has a finite expected waiting time value. In GSPN terminology, a timed transition $t \in T_t$ of a GSPN $G = \langle P, T, F, \gamma, \delta, \omega \rangle$ is *stable*, if the marking $M_h(p) = 0$,$\forall\ p \in {\bullet t}$ is a home marking for M_0 in G, i.e., $\forall\ M' \in \mathcal{R}(M_0) : M_h \in \mathcal{R}(M')$. We call G *stable*, if all its timed transitions T_t are stable.

Referring again to our running example in Fig. 2, we consider the transition representing the check-in of a patient. Here, the arrival rate into this transition is one over 20, i.e., the rate of τ_0. Since the rate of the check-in transition, two over three, is larger than the arrival rate, the transition is stable. Intuitively, this means that tokens are consumed by the check-in transition faster than they are produced by the arrival transition. Hence, there is a finite expected waiting time for tokens in the place preceding the check-in transition. Using the aforementioned queueing network analyzer [23], we compute the local arrival rates into timed transitions to verify their stability. For the values illustrated in Fig. 2, the net indeed turns out to be stable.

The preservation of stability, termed *properness*, can be seen as a correctness criterion for foldings.

Definition 3 (Folding Properness)**.** A folding ψ is called *proper*, if for all $G \in \mathcal{G}$ it holds that G being stable implies that $\psi(G)$ is stable.

Furthermore, foldings are associated with costs that correspond to the error of $\psi(G)$ with respect to G. Clearly, this cost is specific to a particular performance measure and, thus, the type of performance analysis that shall be conducted with the folded model. As a prominent example measure, which we shall use throughout the paper, we consider the expected sojourn time of an arriving case, which is the mean time that it takes for a case from start to finish. In terms of the GSPN formalism the expected sojourn time can be interpreted as follows. Assuming that the arrival transition τ_0 fires only a single time to create a token. Then, the individual sojourn time is the time it takes until the empty marking is reached. However, we are interested in the mean among multiple arrivals. Hence, the expected sojourn time is the average time that it takes until a single arrival into the system disappears. In the GSPN in Fig. 2, the expected sojourn time, 25.5 minutes, is computed using the queueing network analyzer [23].

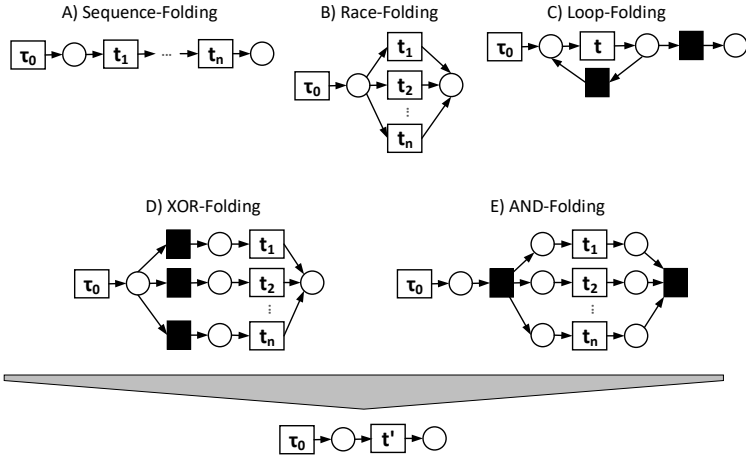


Figure 3: Overview of foldings; the filled (white) rectangles denote immediate (timed) transitions.

We define the *sojourn time cost*, or simply *cost*, of a folding as follows:

Definition 4 (Folding Cost)**.** Let S and S' be random variables for the sojourn times of G, and $G' = \psi(G)$, respectively. The *cost* of applying folding ψ to G is the absolute deviation in expectation between the sojourn times: $c(G, \psi) = |\mathbb{E}S' - \mathbb{E}S|$.

We note that firing delays are given in the GSPN, and hence the main challenge in evaluating sojourn times is obtaining good estimates for waiting times.

4.3. The Functional Component of Foldings

From a functional point of view, each of the foldings illustrated in Fig. 3 exists in two variants: performance information may either be aggregated or eliminated. We distinguish these variants by annotating the folding functions with subscripts. For example, $\psi_{X,A}$ corresponds to an aggregation XOR folding, while $\psi_{X,E}$ is an elimination XOR folding.

The capacity $\gamma'(t')$ and expected duration $\delta'(t')$ assigned to the timed transition t' introduced as part of an aggregation folding depend on the semantics of the abstracted GSPN. In case of an elimination folding, however, these functional aspects are trivially given by $\gamma'(t') = 1$ and $\delta'(t') = 0$. Setting the expected duration to 0 means that, from a performance point of view, the folded part of the GSPN is removed from the analysis.

The definition of elimination foldings has immediate consequences with respect to properness and cost. Elimination foldings are always proper, as a GSPN with duration 0 is stable by definition. In addition, it holds that the sojourn time S' of the resulting GSPN is 0, and therefore the cost $c(G, \psi)$ of applying an elimination folding ψ to G is $|\mathbb{E}S| = \mathbb{E}S$, as sojourn times are always nonnegative.

4.3.1. The Sequence Folding

Aggregation. For an aggregation sequence folding, $G' = \psi_{\rightarrow,A}(G)$, the functional part, i.e., the capacity (γ') and expected duration (δ') of the timed transition t' of G' are set as:

- $\gamma'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)$, i.e., the new transition is allocated the total capacity of the internal timed transitions in G;
- $\delta'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)$, i.e., the new transition is assigned an expected duration, which is the sum of timed transition durations of G.

The new firing rate of transition t' is given by:

$$\lambda'(t') = \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}. \tag{1}$$

The intuition behind the aggregation sequence folding is that G can be approximated by a simpler model that presents a single resource that has the capacity of all resources in G, while having a duration that is the sum of all durations in G. Without resource queues, the two systems are equivalent in expectation. However, due to queueing, the new system is 'faster' on average in terms of waiting time. This explains why G' is stable: it processes more tokens work per time unit, given a steady arrival rate. We formalize this statement by showing that the aggregation sequence folding is proper by proving stability for the resulting GSPN.

Theorem 1. If G is stable, then $\psi_{\rightarrow,A}(G)$ is stable.

Proof. Given Poisson external arrivals into the sequence of transitions and exponential firing durations, it has been shown that the arrival to every transition in the sequence is Poisson with rate β_0 as well [25]. According to [26], the stability condition for G is that for all $t_t \in T_t \setminus \{\tau_0\}$ it holds

that $\beta_0 < {}^{\gamma(t_t)}/_{\delta(t_t)}$. Hence, the stability condition for G is

$$\beta_0 < \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\delta(t_t)}. \tag{2}$$

Due to

$$\min_{1 \le i \le n} \frac{a_i}{b_i} \le \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}, \tag{3}$$

for $a_i, b_i > 0$ [27], we arrive at

$$\begin{aligned} \beta_0 \;&<\; \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\delta(t_t)} \le \\ &\le\; \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)} = \frac{\sum_{t \in T'_t \setminus \{\tau_0\}} \gamma'(t)}{\sum_{t \in T'_t \setminus \{\tau_0\}} \delta'(t)}, \end{aligned} \tag{4}$$

which is a sufficient condition for the stability of G'. □

What remains is to show the cost calculation for the aggregation sequence folding. We do so by calculating the two expected sojourn times: one for the originating system, G, and one for $\psi(G)$.

The firing delays for each of the timed transitions in the two nets are assumed to be independent of the arrival process, and of each other. Further, we assume that tokens fire in a First-Come First-Served order (see the discussion in Section 3.1).

Given the aforementioned assumptions and Definition 1, each timed transition in G can be approximated with an $M/M/1$ queue. Therefore, expected sojourn time through G can be written as [22]:

$$\mathbb{E}S_{\rightarrow} = \sum_{t_t \in T_t \setminus \{\tau_0\}} \frac{\delta(t_t)}{\gamma(t_t) - \beta_0 \delta(t_t)}. \tag{5}$$

We now turn to the calculation of the expected sojourn time for the folded GSPN, namely $G' = \psi_{\rightarrow,A}(G)$. Since the new system is also an $M/M/1$ queue with the new durations and capacities, we may write [22]:

$$\mathbb{E}S'_{\rightarrow} = \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t) - \beta_0 \delta(t_t)}. \tag{6}$$

The resulting cost for the sequence folding is: $c(G, \psi_{\rightarrow,A}) = |\mathbb{E}S'_{\rightarrow} - \mathbb{E}S_{\rightarrow}|$, which is easy to compute as it comprises only of primitives of the originating GSPN, G.

Elimination. As stated above, in all elimination foldings and, thus, also the elimination sequence folding $\psi_{\rightarrow,E}$, it holds that $\gamma'(t') = 1$ and $\delta'(t') = 0$. Hence, $\lambda'(t') \to \infty$ and the sojourn time through G' becomes 0. The cost of the elimination sequence folding is thus given by Eq. 5, as $\mathbb{E}S_{\rightarrow} = 0$.

4.3.2. The Race Folding

The race construct G represents a many-server scenario in queueing theory. Specifically, n servers compete over the queueing customers. We assume that once a server becomes available, it selects the head-of-line out of the waiting tokens. When a token arrives and more than a single resource is available the token is routed randomly to one of the idling servers.

Aggregation. For the aggregation race folding, the capacity (γ) and expected duration (δ) of the timed transition t' of G' are set to be:

- $\gamma'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)$;
- $\delta'(t') = \min_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)$,

i.e., the new construct becomes a 'super-server' with the sum of capacities of all resources in G, and a reduced expected duration (being the minimum between all previous expected durations).

The aggregated system approximates G as follows. The original system G represents a situation where n resources have different skills that result in different capacities and durations per resource (time transition). When we aggregate G, and define the functional part as above, we assume that the new resource has all relevant skills, with the ability to process tokens at the rate of the fastest resource in G. Essentially, the system $\psi_{R,A}(G)$ is an accelerated single-server version of G.

Next, we show that the aggregation race folding is proper by proving stability for the resulting GSPN.

Theorem 2. If G is stable, then $\psi_{R,A}(G)$ is stable.

Proof. Consider a GSPN, G^-, which is structurally equal to G. As for the functional part, each timed transition in G^- has resource capacity of:

$$\gamma^-(t_t) = \sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t). \tag{7}$$

The durations $\delta^-(t_t)$ for the timed transitions in G^- are set to:

$$\delta^-(t_t) = \min_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t). \tag{8}$$

By definition, G^- is a many-server system with $|T_t|$ 'fastest' resources, as $\forall t_t \in T_t \setminus \{\tau_0\}$, the processing rate of G^- is strictly greater than that of G:

$$\lambda^-(t_t) = \frac{\gamma^-(t_t)}{\delta^-(t_t)} = \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\min_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)} \ge \frac{\gamma(t_t)}{\delta(t_t)} = \lambda(t_t). \tag{9}$$

Hence, if given the arrival rate β_0, G is stable, G^- must be stable. The system G^- is an $M/M/n$ queue with homogeneous i.id. servers working with rate $\lambda^-(t_t), t_t \in T_t \setminus \{\tau_0\}$. The stability condition for G^- is [22]:

$$\beta_0 < \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\min_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}, \tag{10}$$

and since G^- is stable, this condition holds. Since $\psi_{R,A}(G)$ is an $M/M/1$ system with arrival rate of β_0 and service rate of $\frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\min_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}$, the stability condition of $\psi_{R,A}(G)$ coincides with Eq. 10 [26]. Therefore, since Eq. 10 holds, the race folding preserves stability.

□

To calculate the cost of the race folding, one needs to compute the expected sojourn times, $\mathbb{E}S_R$ for G, and $\mathbb{E}S'_R$ for $G' = \psi_R(G)$. The system after folding is an $M/M/1$ queue (as we mentioned in the proof of Theorem 2) for which the sojourn time, S'_R, has a well-established closed form (see proof of Theorem 1 for the closed-form expression). The original net is a many-server queueing system with heterogeneous servers, as capacities and expected durations have different values for each timed transition. A closed expression for S_R exists only in the form of approximations. We follow the work of [28] to approximate $\mathbb{E}S_R$. Below we provide a brief intuition to their approach.

The original GSPN, G, is approximated by interpolation between the expected sojourn time of a GSPN G^- (see Proof 4.3.2), which assumes that all transitions work at the rate of the slowest transition, and G^+, which makes a similar assumption to G^- assuming that all transitions have the rate of the fastest transition. The two systems, G^- and G^+ correspond to the well-studied $M/M/n$ queue for which the expected sojourn time is given as a closed expression in [22]. In the special case of ample capacity, the expected sojourn time can be computed precisely (without the need to approximate), as the expected wait time is a race between exponentials (which is the sum of all transition rates).

Elimination. For the elimination race folding $\psi_{R,E}$, again, it holds that $\lambda'(t') \to \infty$ and the sojourn time through G' becomes 0, so that the cost of the elimination race folding is given as $\mathbb{E}S_R$. This cost is computed as detailed above for the aggregation race folding.

4.3.3. The Loop folding

Aggregation. For the folding $\psi_{\circlearrowleft,A}(G)$, the capacity γ' and duration δ' are identical to γ and δ, respectively. However, the arrival rate into the new construct is defined to be $\beta'_0 = \frac{\beta_0\omega}{1-\omega}$ with ω being the weight of the returning immediate transition in the loop construct (Fig. 3, construct (C)). The weight of the outgoing immediate transition is therefore $1-\omega$. The two systems are equivalent as $\frac{\beta_0\omega}{1-\omega}$ is the effective arrival rate into the timed transition of G. Therefore the resulting system is stable, and the cost of the loop folding is 0.

Essentially, the loop folding is the only aggregation folding that preserves the expected sojourn time. The aggregation loop folding reduces the GSPN, without incurring any costs or biases with respect to G. The 'loss' due to this folding comes from the behavioural perspective, as one cannot represent the control-flow properly, by not allowing to repeat the task that corresponds to t'.

Elimination. For the elimination loop folding, $\psi_{\circlearrowleft,E}(G)$, it holds that $\delta'(t') = 0$ and thus $\lambda'(t') \to \infty$. Hence, the system $\psi_{\circlearrowleft,E}(G)$ is stable. However its cost is not 0, and can be calculated as follows. Let t be the timed transition in the loop construct, and ω the return probability. The expected sojourn time through the original system, $\mathbb{E}S_{\circlearrowleft}$ is given by

$$\begin{aligned}\mathbb{E}S_{\circlearrowleft} &= \frac{1}{\frac{\gamma(t)}{\delta(t)} - \frac{\beta_0\omega}{1-\omega}} = \\ &= \frac{\delta(t)(1-\omega)}{\gamma(t)(1-\omega) - \omega\beta_0\delta(t)}. \end{aligned} \tag{11}$$

The system G' has a sojourn time of 0 and hence the expression in Eq. 11 is the loop folding cost.

4.3.4. The XOR folding

Aggregation. The functional part of $\psi_{X,A}(G) = G'$, that is $\langle\gamma', \delta', \omega'\rangle$, is constructed as follows. The capacity (γ) and expected duration (δ) of the timed transition t' of G' are set as:

- $\gamma'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)$, i.e., the new transition is allocated the total capacity of the internal timed transitions in G;
- $\delta'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \omega(t_i)\delta(t_t)$, i.e., the new transition is assigned an expected duration that is the weighted average of the durations of the timed transitions in G, where the weights stem from the respective immediate transitions.

Theorem 3 ascertains the aggregation XOR folding properness.

Theorem 3. If G is stable, then $\psi_{X,A}(G)$ is stable.

Proof. By [29], the stability condition for G is that for all $t_t \in T_t \setminus \{\tau_0\}$ it holds that $\beta_0\omega(t_i) < \frac{\gamma(t_t)}{\delta(t_t)}$ with $t_i \in T_i$ such that $(t_i, t_t) \in F^*$. Hence, for stability to hold we require that $\beta_0 < \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\omega(t_i)\delta(t_t)}$. Due to

$$\min_{1 \le i \le n} \frac{a_i}{b_i} \le \frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i},$$

for $a_i, b_i > 0$, we get that

$$\begin{aligned}\beta &< \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\omega(t_i)\delta(t_t)} \le \\ &\le \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \omega(t_i)\delta(t_t)},\end{aligned}$$

which proves stability of G'. □

To calculate the cost of the XOR folding, we compute the expected sojourn times, S_X in G, and S'_X in $G' = \psi_X(G)$. Since arrivals into the systems (by firing the arrival transition τ_0) are Poisson arrivals, the arrival of timed transitions $t_t \in T_t \setminus \{\tau_0\}$ in G are also Poisson (due to the 'Poisson splitting' property [30]). The arrival rate for $t_t \in T_t \setminus \{\tau_0\}$ is given as $\omega(t_i)\beta_0$ with $t_i \in T_i$ such that $(t_i, t_t) \in F^*$. Note that for GSPNs showing concurrency, the 'Poisson splitting' property does not hold true and a refinement of the above approximation can be made by using Eq. (24) in [23].

The firing delays for each of the timed transitions, $t_t \in T_t \setminus \{\tau_0\}$ are assumed to be independent of the arrival

process, and have exponential durations. These assumptions enable the use of the $M/M/1$ formula for each timed transition to calculate the sojourn times [22]. We write the expected value of S_X as:

$$\mathbb{E}S_X = \sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \omega(t_i)\mathbb{E}S_X^{t_t}, \quad (12)$$

Since it is known that

$$\mathbb{E}S_X^{t_t} = \frac{1}{\lambda(t_t) - \omega(t_i)\beta_0} = \frac{\delta(t_t)}{\gamma(t_t) - \beta_0\omega(t_i)\delta(t_t)}, \quad (13)$$

for $t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i, t_t) \in F^*$, see [22], the sojourn time is given by:

$$\mathbb{E}S_X = \sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \frac{\omega(t_i)\delta(t_t)}{\gamma(t_t) - \beta_0\omega(t_i)\delta(t_t)}. \quad (14)$$

We now turn to the calculation of the sojourn time S'_X for $G' = \psi_X(G)$:

$$\mathbb{E}S'_X = \frac{1}{\lambda'(t) - \beta_0}, \quad (15)$$

with $\lambda'(t) = \frac{\gamma'(t)}{\delta'(t)}$. In primitives of G, the expected sojourn time is given as:

$$\mathbb{E}S'_X = \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \omega(t_i)\delta(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}, t_i \in T_i, (t_i,t_t) \in F^*} \gamma(t_t) - \beta_0\omega(t_i)\delta(t_t)}. \quad (16)$$

The resulting cost for the XOR folding is: $c(G, \psi) = |\mathbb{E}S'_X - \mathbb{E}S_X|$, which is easy to compute as it comprises only of primitives of G, the originating GSPN.

Elimination. Considering the cost of $\psi_{X,E}$, we observe the following. Again, the sojourn time through G' is 0, so that the cost of the elimination XOR folding is given by $\mathbb{E}S_X$, which is computed according to Eq. 14.

4.3.5. The AND folding

Aggregation. For the folding $\psi_{+,A}(G) = G'$, the capacity (γ) and expected duration (δ) of the timed transition t' of G' are set to be:

- $\gamma'(t') = \sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)$, i.e., the new transition is allocated the total capacity of the internal timed transitions in G;
- $\delta'(t') = \max_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)$, i.e., the new transition is assigned with the maximal expected duration of all timed transitions of G.

The logic behind the aggregation of an AND construct is to produce a construct that preserves the sum of capacity, while having the expected processing time of the slowest server. Theorem 4 ascertains the aggregation AND folding properness by proving stability for the resulting GSPN.

Theorem 4. If G is stable, then $\psi_{+,A}(G)$ is stable.

Proof. By [31], the stability condition for G is

$$\beta_0 < \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\delta(t_t)}. \quad (17)$$

Since $\psi_{+,A}(G)$ is an $M/M/1$ queue with arrival rate β_0, it will be stable if and only if:

$$\beta_0 < \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\max_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}. \quad (18)$$

From the stability condition of G in Eq.(17), and since

$$\min_{1 \le i \le n} \frac{a_i}{b_i} \le \frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i},$$

for positive a_i, b_i, we get that:

$$\begin{aligned} \beta &< \min_{t_t \in T_t \setminus \{\tau_0\}} \frac{\gamma(t_t)}{\delta(t_t)} \le \\ &\le \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)} \le \\ &\le \frac{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t)}{\max_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}. \end{aligned}$$

with the last expression being true since the maximum over positive numbers is smaller than their sum. This proves by Eq. 18, the stability condition for $\psi_{+,A}(G)$. □

To calculate the cost of the AND folding, we first present the sojourn time of $G' = \psi_{+,A}(G)$. As G' collapses into an $M/M/1$ queue, we have shown before that the expected sojourn time $\mathbb{E}S'_+$ (AND folding, aggregation) is given by the primitives of G as,

$$\mathbb{E}S'_+ = \frac{\max_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}{\sum_{t_t \in T_t \setminus \{\tau_0\}} \gamma(t_t) - \beta_0 \max_{t_t \in T_t \setminus \{\tau_0\}} \delta(t_t)}. \quad (19)$$

The sojourn time of the original system can be only approximated, as there are no closed-form expressions for the sojourn time of G [22]. We use the approximation proposed in [32] to calculate the expected sojourn time $\mathbb{E}S_+$. Below, we give intuition to their approach.

Prior to calculation of the expected values, the authors of [32] propose an approximate construction of the cumulative distribution function, $F_S(t)$ for the sojourn time as function of time (Eq.(12) in [32]). The expression $F_S(t)$ depends only on the parameters of the original GSPN, G.

Next, to calculate $\mathbb{E}S_+$ the authors propose using the tail formula from basic probability [33]:

$$\mathbb{E}S_+ = \int_0^\infty (1 - F_S(t))dt. \quad (20)$$

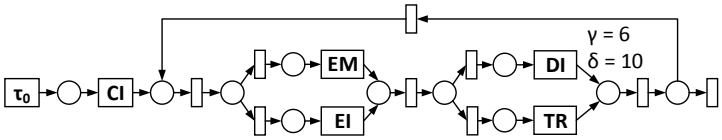


Figure 4: Applying an aggregation AND-folding to the GSPN of the process in Fig. 2 yields a new timed transition for diagnostics (DI).

The integration is straightforward to compute once $F_S(t)$ is obtained.

Elimniation. The cost of $\psi_{+,E}$ is again given by the sojourn time in the original system, $\mathbb{E}S_+$. It is approximated as outlined above.

We illustrate a folding for our example process in Fig. 4. Here, an aggregation AND-folding has been appiled to the GSPN of Fig. 2. The resulting net contains a new timed transition for diagnostics (DI), which represents the transitions of lab tests (LA) and x-rays (XR) in the original model. The new transition is assigned the accumulated capacities and the maximal duration of the timed transitions that it replaces. If the folding would be an elimination folding, the timed transition for diagnostics would be assigned a capacity of one and a duration of zero.

5. Folding Identification by GSPN Decomposition

So far, we discussed the foldings shown in Fig. 3 as a transformation of a complete GSPN of the according structure. However, we employ foldings also to transform parts (aka subnets) of a GSPN, which may enable iterative application of foldings. This holds in particular, as the foldings can be applied to any part of a GSPN that has one of the structures shown in Fig. 3, when removing the arrival transitions τ_0. The reason is that the rate of token arrival into the structures, as encoded by the arrival transitions, can be precomputed by solving the (linear) 'traffic equations' [23, 34], which tie the external arrival rate of the entire GSPN to the internal arrival rates of places of the GSPN.

Observing that the structures in Fig. 3, once the arrival transitions have been removed, correspond to common *single-entry/single-exit (SESE)* control-flow structures, we employ structural decomposition of the GSPN to identify applicable foldings. Specifically, the Refined Process Structure Tree (RPST) [35, 36] is used to parse a GSPN into a hierarchy of SESE *fragments*. Then, the RPST is a containment hierarchy of *canonical* fragments of the graph, which is unique and can be computed in linear time [36]. Fragments can be classified to one out of four structural classes: trivial fragments consists of a single edge; polygons (P) that are sequences of fragments; bonds (B) that are collections of fragments that share entry and exit nodes; and rigids representing any other structure.

To identify which of the foldings outlined in Fig. 3 can be applied to a given GSPN, we rely on the RPST of the GSPN as follows:

Sequence-Folding can be applied to any polygon fragment that has only trivial fragments as children and comprises at least two timed transitions. Assuming that the GSPN has been normalised (immediate transitions may occur only as the first child, as they are redundant at any other position in a polygon), the folding applies to the maximal sequence of timed transitions.

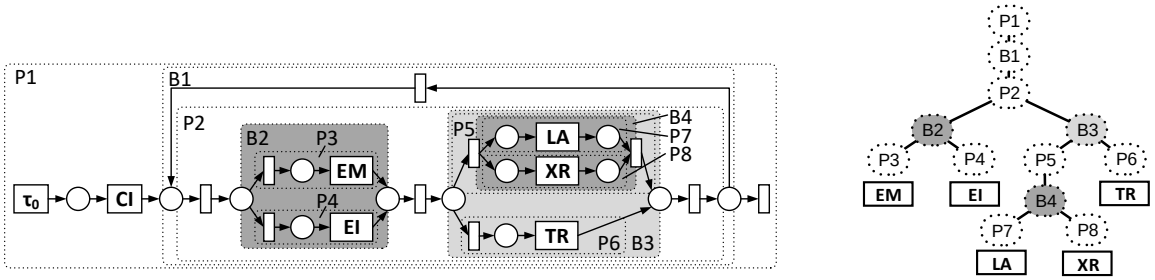
Race-Folding/XOR-Folding/AND-Folding can be applied to place-bordered (race, XOR) or transition-bordered (AND) bond fragments for which one of the following two conditions hold: (i) the fragment contains only polygons of single timed transitions (race, AND); or (ii) the fragment contains polygons with two children, a timed transition that is preceded by an immediate transition (XOR).

Loop-Folding can be applied to place-bordered bonds that are cyclic and part of a polygon. The bonds needs to be followed by an immediate transition in the parent polygon and have the structure visualised in Fig. 3. That is, the children of the bond are polygons of single transitions that are either immediate (if flows in the child lead from the bond entry to the bond exit) or timed (otherwise).

All foldings are identified by polygon and bond fragments of the RPST. As a consequence, foldings may be applied to polygon and bond fragments inside a rigid fragment, but not to the rigid fragment itself. However, certain types of rigids may be restructured into a behaviourally equivalent fragment comprising solely polygons and bonds, see [37, 38, 39]. Restructuring techniques can thereby increase the applicability of the introduced foldings.

The above rules identify foldings iteratively: whenever an applicable folding was found, the respective part of the GSPN is replaced by a timed transition and the rules are checked again. This way, given a GSPN, we obtain a set of *folding instantiations* $\Phi = \{\phi_1, \ldots, \phi_n\}$, each being defined by a folding and the GSPN that is folded. Further, a precedence function $\nu : \Phi \to \wp(\Phi)$ defines, given $\phi \in \Phi$, the set of all folding instantiations that must be applied before f, to generate the GSPN on which f is applied.

We illustrate this approach with our running example, for which the GSPN was shown in Fig. 2. The RPST of the GSPN is illustrated in Fig. 5. Here, the canonical fragments of the GSPN are highlighted in Fig. 5a, while the actual RPST is illustrated as a containment hierarchy of these fragments in Fig. 5b. For the leaf nodes of this tree, we also indicate which timed transitions they contain.



(a) SESE fragments.

(b) The RPST.

Figure 5: RPST of the GSPN of Fig. 2.

Here, an XOR-folding can be applied to the bond fragment $B2$, whereas an AND-folding is applicable for the bond fragment $B4$. If the latter is implemented, the GSPN shown already in Fig. 4 is obtained. Subsequently, an additional XOR-folding becomes applicable for the bond fragment $B3$.

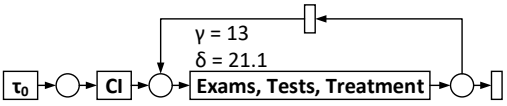


Figure 6: GSPN after applying one AND-folding, two XOR-foldings, and a sequence-folding to the GSPN given in Fig. 2.

6. Optimal Folding of GSPN

Using the foldings proposed above, this section shows how our approach identifies the sequence of foldings that maximises GSPN simplification under the budget constraint. To this end, we define the problem of optimal folding simplification, show how it is encoded as an Integer Linear Program, and elaborate on a method to select an appropriate cost budget.

6.1. Optimal Folding Simplification

Let G be a GSPN, Φ be a set of folding instantiations, and ν the respective precedence function. The cost of every folding instantiation $\phi_i \in \Phi$ is denoted $c_{i,t}$, and calculated as described in Section 4, with $t \in \{E, A\}$ corresponding to either elimination or aggregation, respectively. Note that the only difference between the two folding types is in the definition of their costs.

We use a real-valued budget, $B \in \mathbb{R}^+$, which corresponds to the cumulative error (sum of all costs) that is the result of the foldings with respect to some performance query $q(Y)$, e.g., the total sojourn time. The utility of a folding instantiation $\phi_i \in \Phi$, denoted u_i, is defined as the difference in the number of transitions before and after folding. Here, we assume that after both aggregation and elimination, a new transition is created: for aggregation it has the new functional part in correspondence with the definitions in Section 4.2 , and for elimination it has the duration of 0.

The *Optimal Folding Simplification* (OFS) problem involves finding a sequence of folding instantiations of Φ that respects ν, such that the utility is maximised (the GSPN size is minimised), every folding that corresponds to $\phi_i \in \Phi$ is either elimination or aggregation folding, and the total cost of these foldings does not exceed B.

Assuming that for our example GSPN, see Fig. 2, the budget would allow for the application of one AND-folding (fragment $B4$ in Fig. 5), two XOR-foldings (fragments $B2$ and $B3$), and a sequence-folding (fragment $P2$), the model obtained after folding is shown in Fig. 6. The accumulated utility of this sequence of foldings would be 12, since the net size is reduced from 17 to five transitions.

6.2. ILP Encoding

In what follows, we show a an encoding of the OFS problem to an Integer Linear Program (ILP). The ILP problem is well-studied and many tools exist for its solution. We instantiate the ILP as follows. Let $x_{i,t}$ be a decision variable that receives 1 if the folding instantiation ϕ_i is applied to G with $t \in \{E, A\}$ to represent either aggregation or elimination, and 0 otherwise. Using vector notation for the decision variables, we denote by $\mathbf{x}$ a $2n \times 1$ vector corresponding to $x_{i,t}$. As an auxiliary notation, by $[n]$ we denote the set of integers $\{1, \ldots, n\}$. Next, we define the ILP formulation of the OFS.

Definition 5 (ILP representation of the OFS)**.** The ILP representation of the OFS is given by,

$$\max_{x_{i,t}} \sum_{i=1}^{n} u_i(x_{i,A} + x_{i,E}), \tag{21}$$

subject to:

$$\sum_{i=1}^{n} \sum_{t \in \{E,A\}} c_{i,t} x_{i,t} \leq B, \tag{22}$$

$$\forall\ i, j \in [n], \phi_j \in \nu(\phi_i) :$$

$$x_{j,A} + x_{j,E} \geq x_{i,A} + x_{i,E}, \tag{23}$$

$$\forall\ i \in [n] : x_{i,A} + x_{i,E} \leq 1, \tag{24}$$

$$\forall\ i \in [n], t \in \{E, A\} : x_{i,t} \in \{0, 1\}. \tag{25}$$

The score function ensures that total utility is maximised, while the constraints ensure that folding errors do not exceed budget B and that the precedence ν is respected, and for every folding construct we may apply either an elimination or aggregation. Note that there is always an optimal solution to the OFS problem, since $x_{i,t} = 0, \forall i \in [n], t \in \{E, A\}$ is always feasible solution.

Below, we show that when we consider both elimination and aggregation in Definition 5, if there exists a solution to the ILP, one can find a utility equivalent solution that will only comprise aggregation foldings.

Theorem 5. For any feasible solution $\mathbf{x}^*$ to the ILP in Definition 5 there exists a feasible solution $\mathbf{x}'$ (possibly $\mathbf{x}^*$ itself) in which $\forall i \in [n] : x'_{i,E} = 0$, s. t.

$$\sum_{i=1}^{n} u_i x'_{i,A} = \sum_{i=1}^{n} u_i (x^*_{i,A} + x^*_{i,E})$$

and

$$\sum_{i=1}^{n} c_{i,A} x'_{i,A} \leq \sum_{i=1}^{n} \sum_{t \in \{E,A\}} c_{i,t} x^*_{i,t}.$$

Proof. We prove by constructing solution $\mathbf{x}'$ from a feasible solution $\mathbf{x}^*$.

Let $\mathcal{I} = \{i \in [n] | x^*_{i,E} = 1\}$ be the set of elements of $\mathbf{x}^*$ where elimination is chosen over aggregation. Due to Eq.(24) we get that $\forall i \in \mathcal{I} : x^*_{i,A} = 0$. Let $\mathbf{x}'$ be an alternative vector in which $x'_{j,A} = x^*_{j,A}$ and $x'_{j,E} = x^*_{j,E}$ for $j \notin \mathcal{I}$. Further, $\forall i \in \mathcal{I}$, we set $x'_{i,A} = 1$ and $x'_{i,E} = 0$. Since $c_{i,A} < c_{i,E}, \forall i \in [n]$, the solution $\mathbf{x}'$ is also feasible as it satisfies all constraints in Definition 5, yields the same sum of utilities as $\mathbf{x}^*$, and

$$\sum_{i=1}^{n} c_{i,A} x'_{i,A} = \sum_{i=1}^{n} \sum_{t \in \{E,A\}} c_{i,t} x'_{i,t} \leq \sum_{i=1}^{n} \sum_{t \in \{E,A\}} c_{i,t} x^*_{i,t}.$$

□

It follows that, for every optimal solution that applies elimination foldings there is a solution that achieves the same utility when considering only aggregation foldings at a less or equal cost. In the special case where no elimination foldings appear in $\mathbf{x}^*$, $\mathbf{x}'$ coincides with $\mathbf{x}^*$. Based on the following corollary we may actually apply the aggregation folding separately from elimination, as the aggregation dominates elimination in terms of cost, while preserving the utility score.

Corollary 1. There is an optimal solution $\mathbf{x}'$ to the OFS problem that does not include eliminations: $x'_{i,E} = 0, \forall\ i \in [n]$.

Note that the problem in Definition 5, with $x_{i,E} = 0$ is a tree-knapsack problem, a generalised 0-1 knapsack problem, that is known to be $\mathcal{NP}$-complete. In this problem all items are subject to a partial ordering represented by a rooted tree [40]. In our case, this partial ordering is induced by the precedence function defined over the folding instantiations.

With the assignment $x_{i,E} = 0$ for each $i \in [n]$, we may ignore elimination foldings. We reformulate our ILP as follows. Denote $x_i = x_{i,A}$, the decision variables and $c_i = c_{i,A}$ the folding costs. The resulting ILP is,

$$\max_{x_i} \sum_{i=1}^{n} u_i x_i,$$

subject to:

$$\sum_{i=1}^{n} c_i x_i \leq B,$$

$$\forall\ i, j \in [n], \phi_j \in \nu(\phi_i) : x_j \geq x_i,$$

$$\forall i \in [n] : x_i \in \{0, 1\}.$$

By Corollary 1, this ILP has the same optimal solution as the one formulated as OFS. Keeping in mind that the partial order is induced by a rooted tree we get an identical ILP formulation to the ILP formulation of the tree-knapsack problem as it is presented in Section 2 of [40].

6.3. Budget Selection

The only input of the OFS problem that is not based on the originating model, G, is the budget B. The budget can be interpreted as the amount of trust in G: B should be small if trust is high, and vice versa.

When applying our approach to a model G that was constructed based on some event log L, the budget can be set in the spirit of model selection techniques that are often used in machine learning [18]. Specifically, one may elicit the 'best' budget for a given log via K-fold cross-validation [18, Ch. 7]: The event log is partitioned into K parts, and the budget is determined based on random $\frac{K-1}{K}$ parts that are treated as training logs, and tested on the remaining part. All budgets between 0 (no folding) and $\sum_{i=1}^{n} c_i$ (unlimited folding) are considered and the budget that yields the most accurate answer to the performance query $q(Y)$ under a certain criteria (e.g., sampled root-mean squared error) is selected for the OFS problem.

Lastly, the training set was used to elicit the best budget, B^*, which yielded the most accurate $\hat{q}(Y)$ with respect to the training set. Lastly, the best (training based) budget B^* corresponds to G^* which is the best model based on the training set (folded under budget B^*). This model was then used to predict the total sojourn time for patients from the test set (five different months of hospital data). If found more accurate than G_0 it would indicate that it has higher generality (less overfitting the event log).

7. Evaluation

We evaluated our simplification approach with two real-world cases, using an implementation of our approach in Python[1] that is based on the Gurobi [41] ILP solver. The input is a process model (GSPN), and an event log; the method produces a folded GSPN model based on either aggregation or elimination of the five control-flow constructs, namely sequence, race, XOR, AND, and loop.

Our results indicate that the proposed simplification technique is able to provide accurate theoretical bounds for the impact of model simplification on performance measures. Furthermore, we show that aggregation-based folding balances between model generality and accuracy: It provides simple models, largely without comprising the ability to predict the process' performance.

Further, we compare our approach to heuristic model simplification techniques. Foldings turn out to give users improved control of the simplification process from the performance perspective, compared to heuristic noise filtering.

Lastly, we show the impact of the initial model on folding in terms of the balance between fitness and precision. More precise and less general models benefit from the generalization power that the folding framework provides. Again, the guaranteed error bounds due to simplification are shown to be accurate in practice. Moreover, using the aggregation folding, we prevent under-fitting that may occur due to an increase in generalization.

Below, we introduce our datasets and experimental setup (Section 7.1), before presenting the main results for aggregation and elimination foldings (Section 7.2). We conclude the section with a discussion of our results (Section 7.3).

7.1. Datasets and Setup

Our experiments were based on two real-world datasets coming from different domains, namely healthcare and call centres. The first dataset comprises five months (April-August, 2014) of operational data stemming from the treatment process of DayHospital, a large outpatient hospital in the United States. The hospital treats approximately 1000 patients a day, with patients arriving and leaving on the same day. The average length-of-stay per visit is 4.4 hours (standard deviation of 2 hours) with the highest number of patients arriving between 8:00 and 11:00 in the morning. The dataset includes the following attributes: case identifier, activity start time, activity end time, and resource performing the activity.

We selected April as our training set for discovering a GSPN and enriching it with data, as well as for the error budget selection (outlined in Section 6). The other four months were used as separate test sets, to validate the results.

The second dataset comes from an Israeli telecommunication company, ILTelecom and is gathered and stored in the Technion laboratory for Service Enterprise Engineering (SEELab)[2]. Specifically, ILTelecom operates a call centre, which processes up to 50,000 service requests a day on various topics. The business process consists of several steps, where a customer chooses whether to receive service from an interactive voice response unit (IVR), or to be served by a human agent. The centre is operated with around 600-800 agent positions on weekdays and 200-400 agent positions on weekends. Further, several types of services are provided; the most common are Private, Business, Technical and Content Internet.

For our empirical evaluation, we selected three months of data, from January 1, 2008 to March 31, 2008. The data features the events of a customer service log as well as those of a resource service log. As for the hospital use-case, we divided the data into two subsets: a training set (75% of the data set) and a test set (25%).

For either dataset, we assess the total sojourn time per patient or customer, respectively. That is, the performance query $q(Y)$ is defined as the sojourn time, while its estimate $\hat{q}(Y)$ is derived for a GSPN by means of the queueing network analyzer [23].

We focus on three evaluation scenarios:

(1) *Aggregation vs. Elimination.* We explored the impact of aggregation and elimination simplifications given different values of error budget. To test aggregation foldings, we apply the framework with the reduced ILP from Section 6, where elimination foldings are not allowed. For the elimination folding experiments, we do not allow aggregation. By separating the two, we are able to observe their effects in a controlled manner, since, according to our theory, aggregation foldings dominate elimination foldings, see Corollary 1.

(2) *Noise Filtering vs. Elimination Folding.* We considered the interplay of methods to avoid over-fitting in the control-flow dimension by using noise filtering in the initial GSPN discovery and simplification. We altered the aggressiveness of noise filtering in the discovery algorithm and estimated the prediction accuracy of an unfolded model. We compared the result to elimination folding using a cross-validated budget.

(3) *Balancing Fitness and Precision.* We test whether our approach is influenced by the underlying control-flow discovery algorithm, and its ability to balance fitness and precision [42]. Specifically, we repeated the experiment in Scenario 1 when varying the noise threshold applied to filter infrequent execution paths. The increase in noise threshold produces an increase in precision, and a decrease in fitness.

To discover an initial Petri net in Scenario 1, we applied the Inductive Miner (IM) [19] on the training set, with resources being treated as activities and a 20% noise threshold (see Fig. 1b). Our selection of this noise threshold is rationalized by the unbiasedness of the 20% model with

[1] `https://github.com/ArikSenderovich/P3Folding/`

[2] `http://ie.technion.ac.il/Labs/Serveng`

respect to the sojourn time query (see Section 7.2, namely Fig. 8b and Fig. 8d). Next, we enriched the model based on the training set using the techniques described in [16], thus turning it into the initial GSPN. For Scenario 2, we used the IM with various noise thresholds, without folding the models. Lastly, in Scenario 3 we compared the base case model that stems from the IM with 20% noise threshold to other models that result from the IM with various levels of noise (15%, 25%, and 30%).

To quantify the accuracy of models, we use the (sampled) root-mean squared error (RMSE) and bias, which are standard statistical accuracy measures, defined as follows. Let $\{y_k\}_{k=1}^{K}$ be the sample of K total sojourn times as observed in the log (training or test). Then, the sampled RMSE and bias are defined as:

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}[y_k - \hat{q}(Y)]^2}, \tag{26}$$

$$\text{Bias} = \frac{1}{K}\sum_{k=1}^{K}[y_k - \hat{q}(Y)]. \tag{27}$$

Note that our theoretical guarantees when solving the Optimal Folding Simplification (OFS) problem are provided only for the bias, and not for the RMSE.

To summarize, controlled variables in the experiments are the folding method (aggregation vs. elimination), the budget, and the noise filtering thresholds for the Inductive Miner, while the RMSE and bias serve as the dependent variables.

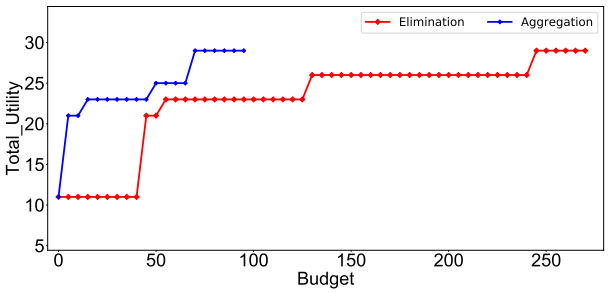
7.2. Results

We start with the observation that the analysis is feasible for the used real-world datasets. While the tree-knapsack problem is known to be $\mathcal{NP}$-complete [40], modern ILP solvers enable efficient reasoning on this problem. The run-time of our method when considering all training data and all budget configurations was 152 seconds, on an Intel $i7-4510U$ running at 2.00GHz with 4GB RAM. This run-time is in the same range as model discovery via the Inductive Miner on the same PC.

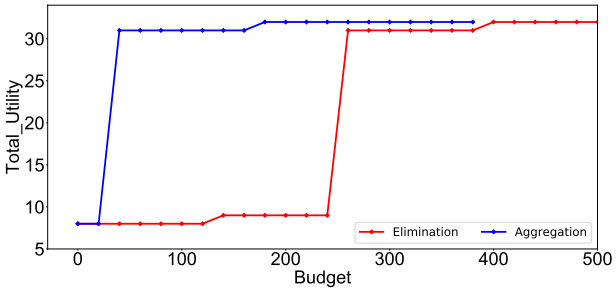
In the remainder of this section, we outline the main results given the three aforementioned scenarios, namely *aggregation vs. elimination* (Section 7.2.1), *noise filtering vs. folding* (Fig. 7.2.1), and *balancing fitness and precision* (Section 7.2.3).

7.2.1. Scenario I: Aggregation vs. Elimination.

Fig. 7 presents the total utility as in Definition 5, as function of various budgets; Fig. 7a presents the results for the hospital use case, while Fig. 7b corresponds to the call centre dataset. Specifically, the vertical axis is the total number of nodes reduced with respect to the unfolded model, and the horizontal axis depicts different values of the user-defined budget. The total utility in the OFS problem corresponds to model simplicity, *i.e.*, it quantifies the total



(a) Total utility as function of budget (hospital).



(b) Total utility as function of budget (call centre).

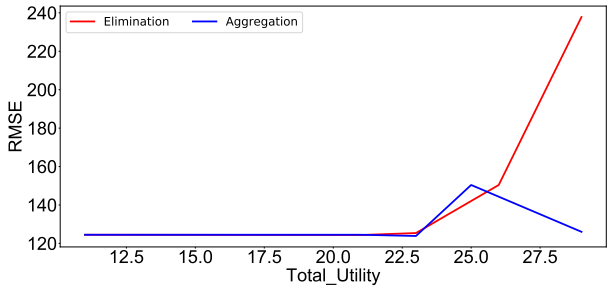
Figure 7: Total utility (i.e., size of the model) in relation to the error budget used for folding.

number of transitions that are reduced with respect to the original model. Elimination foldings are more expensive, and therefore it is expected that higher budget is required to fully fold the model.

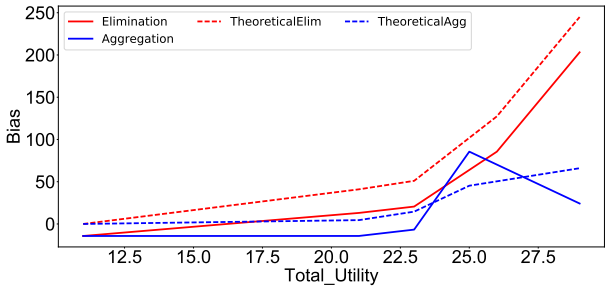
Next, we present the two accuracy measures against total utility (model size), namely RMSE (Fig. 8a for the hospital data and Fig. 8c for call centre data) and bias (Fig. 8b for the hospital use-case and Fig. 8d for the call centre use-case). An increase in utility corresponds to model-size reduction. Further, note that the empirical bias can be compared to the theoretical errors due to aggregation and elimination folding, as the latter correspond to the total cost (left-hand side of Eq. (22) in Definition 5). The charts enable us to test whether the accuracy of a model decreases when it is simplified.

The aggregation folding results are presented using the blue solid lines, while the elimination folding is illustrated using red solid lines. The vertical axes, namely RMSE and bias, are presented in minute time units. Theoretical lines are presented with dashed lines. A positive (negative) bias represents an under-(over-)estimation of the total length-of-stay for arriving cases. The RMSE of unbiased models corresponds to the variance when estimating $q(Y)$.

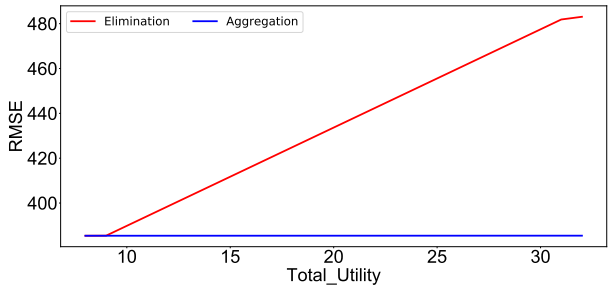
For the hospital scenario, the RMSE results are presented in Fig. 8a. There, we observe that the RMSE that corresponds to the aggregation folding increases at first, as model size is reduced, before showing an improvement, indicating that over-fitting is avoided. This non-monotonic behaviour of the aggregation folding supports the need for cross-validating the best budget from data (see Section 6.3).



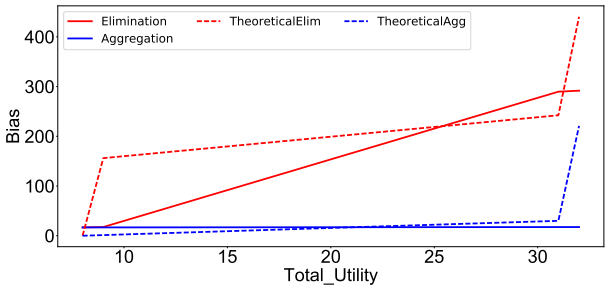
(a) RMSE in hospital data.



(b) Bias in hospital data.

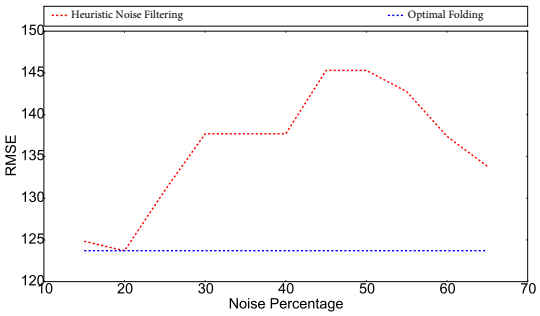


(c) RMSE in call centre data.

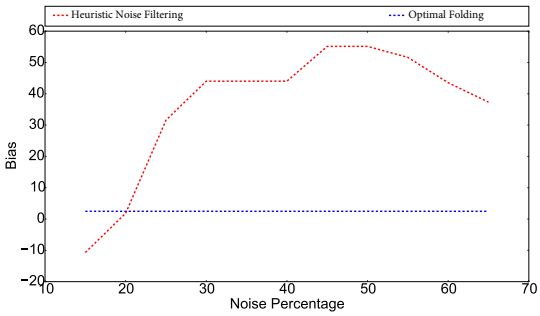


(d) Bias in call centre data.

Figure 8: RMSE and bias in relation to model size reduction. Theoretical error (bias) is presented.



(a) RMSE for Control-Flow Noise Filtering.



(b) Bias for Control-Flow Noise Filtering.

Figure 9: RMSE and bias as function of control-flow noise filtering.

For the call centre scenario, all models in Figure Fig. 8c present a similar RMSE, as the aggregation folding results in 'lossless' model simplification. This result can be explained as follows. The RMSE with respect to the sojourn time query corresponds to the variance when estimating $q(Y)$. If the folding (empirically) preserves the variance, which is not guaranteed for any dataset, but happens to be the case for the call centre data, then the RMSE will not change.

Moving to the elimination folding, we observe a monotonic deterioration in the RMSE for both scenarios, as model-size is decreased. By using elimination, the process is shortened without compensations, which leads to the expected under-fitting, and to lower prediction accuracy.

When shifting the focus to the bias measure (Fig. 8b and Fig. 8d), we observe (similarly for both datasets) that the two types of foldings first un-bias the model. Then, for the elimination folding the empirical bias increases following (in trend) its theoretical counterpart. However, for aggregation folding in the hospital scenario, the expected bias rises as model size reduces, while the empirical one is reduced. This points at the beneficial effect of aggregation folding on datasets that include noise. Further, the empirical bias in the call centre scenario slightly decreases as utility increases. This result stems from the fact that the smaller model appears to be a better fit for the call centre reality.

7.2.2. Scenario II: Noise Filtering vs. Elimination Folding. In this scenario we compare model simplification via elimination folding to *heuristic* noise filtering, which is often used in discovery algorithms. For example, a heuristic approach for the Inductive Miner involves filtering the event

log using a percentage threshold [19]. In the experiments, we set these heuristic values to be between 15% and 30%. For the elimination folding we set the budget to a value that yields the best experimental RMSE from Fig. 8a. The budget set corresponds to a total utility of 23, i.e., the model is a simplified model with 23 eliminated transitions.

To test the predictive accuracy of the discovered models, we apply the enrichment and approximation techniques that we used in previous scenarios to both datasets. call centre results show a similar behaviour and are thus omitted. The results in terms of the two performance measures, namely RMSE and bias, are presented in Fig. 9. Note that models that were constructed with noise filtering were not folded.

We observe that optimal RMSE and lowest bias are achieved at 20% heuristic noise threshold (Fig. 9a, Fig. 9b), with both the RMSE and the bias of the heuristic approach having a non-monotonic shape that deteriorates for moderate filtering percentage and improves when a high noise filtering threshold is set. Further, we observe that the best performance in terms of RMSE when using noise filtering is equivalent to the best performance of the elimination folding. However, the elimination folding provides theoretical bounds that are able to predict the bias for a selected budget, while the heuristic approach disregards the impact of filtered traces on performance.

7.2.3. Scenario III: Balance of Fitness and Precision

In Section 7.2.1, we presented the results for an initial model that was discovered using the Inductive Miner, with noise threshold of 20%, i.e., 20% of the most infrequent traces were filtered from the event log. This model is an appropriate starting point as it is empirically unbiased (see Fig. 8). However, selecting a single noise threshold for discovering the initial unfolded model fixes a specific level of fitness and precision as defined in [42].

In this last scenario, we alter the noise to yield different levels of fitness and precision. When the threshold is increased, the models become more precise, while losing fitness [19].

Fig. 10 presents the RMSE and bias (similarly to Fig. 8) for the hospital dataset. The noise threshold for the initial discovery varies between 15% and 30% with 20% being the original noise threshold presented in Fig. 8.

Discovery with 15% threshold yields a more fitting model than the one in Fig. 8. Bias increases monotonically for both folding types. For the aggregation folding, this is in contrast to the reduction in bias observed in Fig. 8. Yet, the RMSE shows a similar reduction as in Fig. 8, which actually indicates that the variance of the estimate is reduced when quantifying $q(Y)$. This result implies that over-fitting is prevented at the expense of introducing a bias.

As we increase the noise threshold to 25% and 30%, the unfolded models become biased, hence the benefit of aggregation folding increases. For both cases demonstrated in Fig. 10c to Fig. 10f, we observe a behaviour where initially, as model size decreases, the RMSE and bias grow, followed by a decline of both measures. This pattern demonstrates that smaller models yield more accurate results than the unfolded ones. Interestingly, for both higher noise thresholds, the empirical results for the smallest attainable models, coincide with the theoretical guarantee provided by the OFS solution. To conclude, the empirical results showed that aggregation foldings become more beneficial when model precision increases, and fitness decreases.

7.3. Discussion

The main finding of our experiments is that folding, by means of aggregation, can improve prediction accuracy, while providing model simplification. In analogy to model selection techniques in machine learning, the model simplification framework aggregates the less significant parts of the process model. Here, significance is measured by a theoretical bound that each folding introduces.

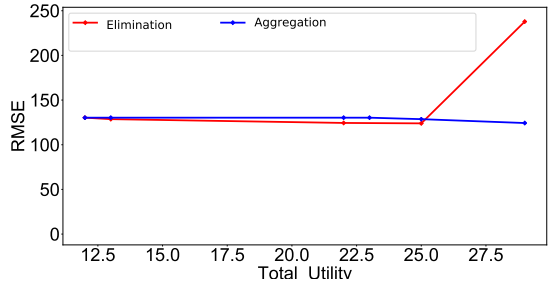
In Section 6 (Corollary 1), we showed that elimination foldings are dominated by aggregation foldings. Our evaluation supports this claim. Both RMSE and bias obtained with aggregation folding improve over those obtained by elimination folding in virtually all cases.

We also compared an event log filtering based approach for model simplification and elimination folding of the model. The main drawback of log filtering is that it removes infrequent traces from the event log without considering their durations, leading to decoupling of the filtering approach from the performance perspective. Hence, one may filter infrequent paths that have long durations and thus contribute much to the workload, leading to performance prediction inaccuracy. In contrast to log filtering, the elimination folding prioritises the removal of control-flow structures that would introduce the smallest changes to the model-based performance prediction.

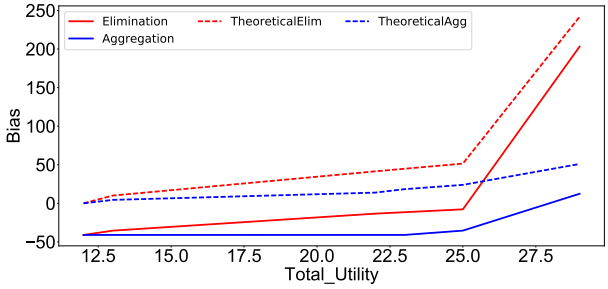
Lastly, in Section 7.2.3 we demonstrate the impact of the discovery algorithm and its parameters on folding. Specifically, we examined different noise thresholds when applying the Inductive Miner for the initial discovery. This experiment was used to test the sensitivity of our approach to the balance between fitness and precision at the control-flow stage. We have shown that aggregation folding yields larger improvements when the initial model is more precise and less fitting. This implies that simplifying overly general models, such as discovered by the Inductive Miner with low noise thresholds, is inadvisable. However, when models are precise (i.e., they do not allow behaviour outside of the event log) would benefit from the generalization that our folding technique provides.

8. Related Work

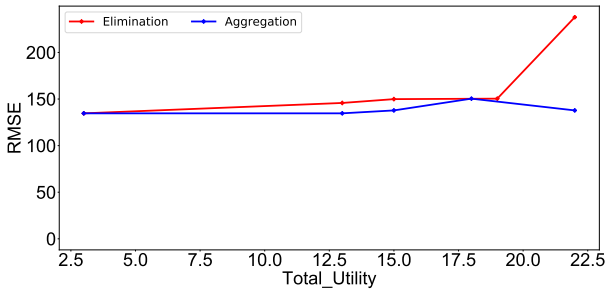
Our work relates to the broader field of process model simplification, or synonymously process model abstraction. In general, such simplification considers ad-hoc rules that transform the model into a smaller one, while preserving



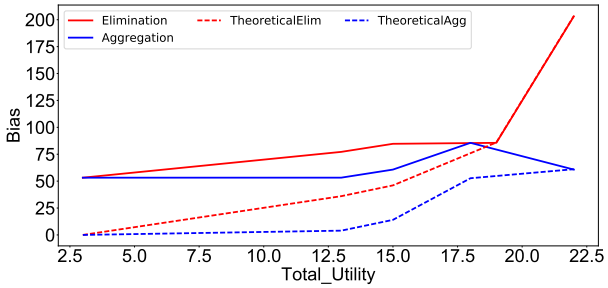
(a) RMSE, 15% Noise Threshold.



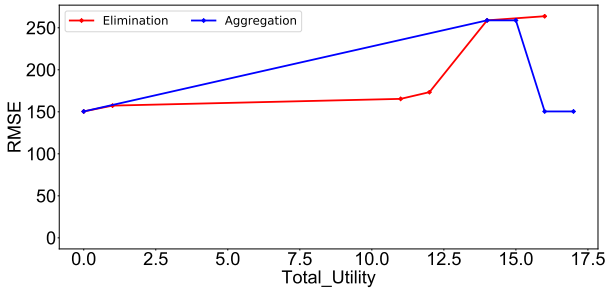
(b) Bias, 15% Noise Threshold.



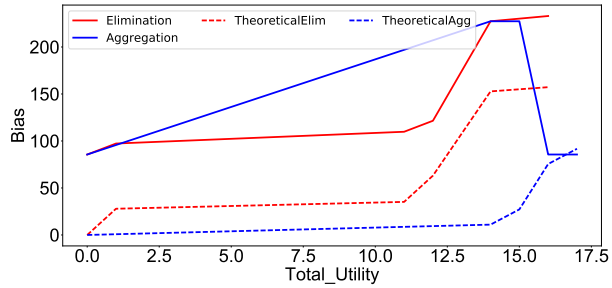
(c) RMSE, 25% Noise Threshold.



(d) Bias, 25% Noise Threshold.



(e) RMSE, 30% Noise Threshold.



(f) Bias, 30% Noise Threshold.

Figure 10: RMSE and Bias as function of Noise Filter percentage. Precision (Fitness) increases (decreases) with noise threshold.

similarity to the originating model [43]. Also, model simplification proposed in the context of automated model discovery argued that a combination of aggregation and elimination of model components is needed [44]. Yet, these approaches are primarily concerned with obtaining a simpler visualisation of the process, while preserving behavioural relations between its various components. Approaches for process model simplification employ a diverse set of techniques, such as semantic abstraction goals expressed as constraints [45]. When process models are discovered from event data, model simplification may be achieved by filtering the event data before constructing a model [9] or by exploiting to the event log to simplify the model after a first synthesis step [9]. However, none of these works considered model simplifications that preserve performance information, which is the focus of our work.

Existing performance-oriented model simplification techniques approximate typical process patterns (e.g., sequence, choice) via concepts from queueing theory, and guarantee a certain notion of equivalence between the original model and the resulting simplifications [46, 47]. However, these techniques do not propose a method of locating typical patterns, and thus were not automated. Moreover, they do not suggest how to order the simplification operations if multiple transformations can be applied. Finally, some of the proposed performance bounds are not well-grounded [47].

Manual simplification of GSPN models has been considered before. In [21], GSPNs are simplified by using ad-hoc rules, not providing any error bounds. A simplification technique that provides bounds for specific performance measures between the original model and the resulting simple model includes decomposition and aggregation of the GSPN [48, 49]. The first step (decomposition) refers to partitioning the GSPN into subnets, such that the subnets are weakly dependent. Every subnet can then be efficiently analysed without unfolding the underlying CTMC [50, 47].

The second step (aggregation) aggregates the subnet according to performance-preserving rules.

Our approach takes up the ideas of model aggregation based on folding steps [12]. However, the steps in [12] incorporate ad-hoc assumptions and violating them may yield an unbounded estimation error with respect to the original model. In this work, we formulate an optimisation problem aiming at a maximal number of folding instantiations, subject to guarantees regarding an error budget. This enables us to balance performance fitness and generalisation of the resulting model in a principled manner.

9. Conclusion

In this work, we presented a novel technique for automated simplification of models that aims at improving performance analysis of business processes. Specifically, we proposed foldings of GSPNs, structural transformations of a GSPN that aggregate or eliminate the performance information. We also showed how to find an optimal sequence of applying folding to obtain a minimal model under a given error budget for the performance estimates. As such, foldings enable us to generalise a model in the performance dimension, while maximally preserving the process perspective of the original model. The evaluation of our technique showed that using our approach one can balance between model generality and its accuracy, with respect to the unfolded model that was discovered from a real-world event log. The proposed technique can be viewed as a model selection method for process models, in analogy to model selection techniques in machine learning.

Our future work directions include several aspects. First, we considered systems that show a steady-state and exhibit exponential firing rates. The former may result in inaccurate models during time-varying intervals, whereas the latter limits the expressiveness of the models in terms of service time distributions that can be captured. Therefore, we intend to lift our approach to more expressive models, which enable us to capture dependencies between the time-of-day and system load, and also relax the assumption of exponential firing delays.

Note that the latter relaxation will not have a direct impact on the folding framework as we defined it, since we consider costs only in terms of first-moments (expected sojourn times). If we extend the framework to consider variance, which is an important factor in predicting performance in non-exponential systems, we may need to use advanced asymptotic approaches from queueing theory to (re-)evaluate folding costs.

Furthermore, we aim at extending the framework to further performance measures beyond sojourn times, e.g., queue length and local wait times. To this end, the theoretical foundation of our approach needs to be adjusted accordingly. That is, the cost calculations per folding would have to be derived with respect to the new measure.

Acknowledgement. Research partially funded by the German Research Foundation (DFG) under grant agreement number 246594964. The work of Alexander Shleyfman was supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

References

[1] O. A. Soremekun, J. K. Takayesu, S. J. Bohan, Framework for analyzing wait times and other factors that impact patient satisfaction in the emergency department, The Journal of emergency medicine 41 (6) (2011) 686–692.

[2] M. S. Daskin, Service Science, Wiley. com, 2011.

[3] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[4] A. Rogge-Solti, M. Weske, Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays, in: S. Basu, C. Pautasso, L. Zhang, X. Fu (Eds.), ICSOC, Vol. 8274 of Lecture Notes in Computer Science, Springer, 2013, pp. 389–403.

[5] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining - predicting delays in service processes, in: M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, J. Horkoff (Eds.), Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings, Vol. 8484 of Lecture Notes in Computer Science, Springer, 2014, pp. 42–57. doi:10.1007/978-3-319-07881-6.
URL http://dx.doi.org/10.1007/978-3-319-07881-6

[6] W. M. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[7] W. M. P. van der Aalst, V. Rubin, H. Verbeek, B. F. van Dongen, E. Kindler, C. W. Günther, Process mining: a two-step approach to balance between underfitting and overfitting, Software & Systems Modeling 9 (1) (2010) 87–111.

[8] J. C. Buijs, B. F. Van Dongen, W. M. van der Aalst, On the role of fitness, precision, generalization and simplicity in process discovery, in: On the Move to Meaningful Internet Systems: OTM 2012, Springer, 2012, pp. 305–322.

[9] D. Fahland, W. M. Van Der Aalst, Simplifying discovered process models in a controlled manner, Information Systems 38 (4) (2013) 585–605.

[10] S. J. van Zelst, B. F. van Dongen, W. M. P. van der Aalst, Avoiding over-fitting in ilp-based process discovery, in: Motahari-Nezhad et al. [51], pp. 163–171. doi:10.1007/978-3-319-23063-4_10.
URL http://dx.doi.org/10.1007/978-3-319-23063-4_10

[11] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, A. Mandelbaum, P ˆ3 -folder: Optimal model simplification for improving accuracy in process performance prediction, in: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, 2016, pp. 418–436.

[12] A. Senderovich, A. Rogge-Solti, A. Gal, J. Mendling, A. Mandelbaum, S. Kadish, C. A. Bunnell, Data-driven performance analysis of scheduled processes, in: Motahari-Nezhad et al. [51], pp. 35–52. doi:10.1007/978-3-319-23063-4_3.
URL http://dx.doi.org/10.1007/978-3-319-23063-4_3

[13] W. M. P. van der Aalst, M. Schonenberg, M. Song, Time prediction based on process mining, Information Systems 36 (2) (2011) 450–475.

[14] M. de Leoni, W. M. van der Aalst, M. Dees, A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs, Information Systems 56 (2016) 235–257.

[15] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, F. M. Maggi, Complex symbolic sequence encodings for predictive monitoring of business processes, in: Business Process Management, Springer, 2015, pp. 297–313.

[16] A. Rogge-Solti, W. M. van der Aalst, M. Weske, Discovering stochastic petri nets with arbitrary delay distributions from event logs, in: Business Process Management Workshops, Springer, 2013, pp. 15–27.
[17] A. Rozinat, R. Mans, M. Song, W. M. P. van der Aalst, Discovering simulation models, Information Systems 34 (3) (2009) 305–327.
[18] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.
[19] S. J. Leemans, D. Fahland, W. M. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: Business Process Management Workshops, Springer, 2014, pp. 66–78.
[20] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with generalized stochastic Petri nets, John Wiley & Sons, Inc., 1994.
[21] G. Balbo, S. C. Bruell, S. Ghanta, Combining queueing networks and generalized stochastic petri nets for the solution of complex models of system behavior, IEEE Transactions on Computers 37 (10) (1988) 1251–1268.
[22] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications; 2nd Edition, Wiley, 2006.
[23] W. Whitt, The queueing network analyzer, Bell System Technical Journal 62 (9) (1983) 2779–2815.
[24] W. M. van der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow patterns, Distributed and parallel databases 14 (1) (2003) 5–51.
[25] J. R. Jackson, Jobshop-like queueing systems, Management science 10 (1) (1963) 131–142.
[26] M. Bramson, Stability of queueing networks, Springer, 2008.
[27] A. Mandelbaum, M. I. Reiman, On pooling in queueing networks, Management Science 44 (7) (1998) 971–981.
[28] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining for delay prediction in multi-class service processes, Information Systems 53 (2015) 278–295.
[29] R. W. Hall, Queueing methods for services and manufacturing, 1990.
[30] S. I. Resnick, Adventures in stochastic processes, Springer Science & Business Media, 2013.
[31] J. W. Panagiotis Konstantopoulos, Stationary and stability of fork-join networks, Journal of Applied Probability 26 (3) (1989) 604–614.
URL http://www.jstor.org/stable/3214417
[32] S.-S. Ko, R. F. Serfozo, Response times in m/m/s fork-join networks, Advances in Applied Probability 36 (3) (2004) 854–871.
URL http://www.jstor.org/stable/4140412
[33] W. Feller, An introduction to probability theory and its applications: volume I, Vol. 3, John Wiley & Sons New York, 1968.
[34] S. Balsamo, A. Marin, Composition of product-form generalized stochastic petri nets: a modular approach, in: Proc. ESM, 2009, pp. 26–28.
[35] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, Data and Knowledge Engineering (DKE) 68 (9) (2009) 793–818.
[36] A. Polyvyanyy, J. Vanhatalo, H. Völzer, Simplified computation and generalization of the refined process structure tree, in: M. Bravetti, T. Bultan (Eds.), WS-FM, Vol. 6551 of Lecture Notes in Computer Science, Springer, 2010, pp. 25–41. doi:10.1007/978-3-642-19589-1.
URL http://dx.doi.org/10.1007/978-3-642-19589-1
[37] A. Polyvyanyy, L. García-Bañuelos, M. Dumas, Structuring acyclic process models, Inf. Syst. 37 (6) (2012) 518–538. doi:10.1016/j.is.2011.10.005.
URL http://dx.doi.org/10.1016/j.is.2011.10.005
[38] C. Ouyang, M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede, J. Mendling, From business process models to process-oriented software systems, ACM Trans. Softw. Eng. Methodol. 19 (1). doi:10.1145/1555392.1555395.
URL http://doi.acm.org/10.1145/1555392.1555395
[39] J. Koehler, R. Hauser, Untangling unstructured cyclic flows - A solution based on continuations, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Agia Napa, Cyprus, October 25-29, 2004, Proceedings, Part I, Vol. 3290 of Lecture Notes in Computer Science, Springer, 2004, pp. 121–138. doi:10.1007/978-3-540-30468-5_10.
URL http://dx.doi.org/10.1007/978-3-540-30468-5_10
[40] D. X. Shaw, G. Cho, The critical-item, upper bounds, and a branch-and-bound algorithm for the tree knapsack problem, Networks 31 (4) (1998) 205–216. doi:10.1002/(SICI)1097-0037(199807)31:4<205::AID-NET1>3.0.CO;2-H.
URL http://dx.doi.org/10.1002/(SICI)1097-0037(199807)31:4<205::AID-NET1>3.0.CO;2-H
[41] I. Gurobi Optimization, Gurobi optimizer reference manual (2015).
URL http://www.gurobi.com
[42] J. C. Buijs, B. F. van Dongen, W. M. van der Aalst, Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity, International Journal of Cooperative Information Systems 23 (01) (2014) 1440001.
[43] S. Smirnov, H. A. Reijers, M. Weske, T. Nugteren, Business process model abstraction: a definition, catalog, and survey, Distributed and Parallel Databases 30 (1) (2012) 63–99.
[44] C. W. Günther, W. M. van der Aalst, Fuzzy mining–adaptive process simplification based on multi-perspective metrics, in: Business Process Management, Springer, 2007, pp. 328–343.
[45] S. Mafazi, G. Grossmann, W. Mayer, M. Schrefl, M. Stumptner, Consistent abstraction of business processes based on constraints, Journal on Data Semantics 4 (1) (2014) 59–78.
[46] L. Zerguini, On the estimation of the response time of the business process, in: 17th UK Performance Engineering Workshop, University of Leeds, Citeseer, 2001.
[47] L. Zerguini, K. M. van Hee, A new reduction method for the analysis of large workflow models., in: Promise, 2002, pp. 188–201.
[48] G. Ciardo, K. S. Trivedi, A decomposition approach for stochastic petri net models, in: Petri Nets and Performance Models, 1991. PNPM91., Proceedings of the Fourth International Workshop on, IEEE, 1991, pp. 74–83.
[49] C. M. Woodside, Y. Li, Performance petri net analysis of communications protocol software by delay-equivalent aggregation, in: Petri Nets and Performance Models, 1991. PNPM91., Proceedings of the Fourth International Workshop on, IEEE, 1991, pp. 64–73.
[50] J. Freiheit, J. Billington, New developments in closed-form computation for gspn aggregation, in: ICFEM, Springer, 2003, pp. 471–490.
[51] H. R. Motahari-Nezhad, J. Recker, M. Weidlich (Eds.), Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings, Vol. 9253 of Lecture Notes in Computer Science, Springer, 2015. doi:10.1007/978-3-319-23063-4.
URL http://dx.doi.org/10.1007/978-3-319-23063-4