Compositional safe approximation of response time distribution of complex workflows

Laura Carnevali[1], Marco Paolieri[2], Riccardo Reali[1], and Enrico Vicario[1]

[1] Department of Information Engineering, University of Florence, Italy
{laura.carnevali,riccardo.reali,enrico.vicario}@unifi.it
[2] Department of Computer Science, University of Southern California, USA
paolieri@usc.edu

Abstract. We propose a compositional technique for efficient evaluation of the cumulative distribution function of the response time of complex workflows, consisting of activities with generally distributed stochastic durations composed through sequence, choice/merge, split/join, and repetition blocks, with unbalanced split and join constructs that break the structure of well-formed nesting. Workflows are specified using a formalism defined in terms of stochastic Petri nets, that permits decomposition of the model into a hierarchy of sub-workflows with positively correlated response times, which guarantees a stochastically ordered approximation of the end-to-end response time when intermediate results are approximated by stochastically ordered distributions and when dependencies are simplified by replicating activities appearing in multiple sub-workflows. This opens the way to an efficient hierarchical solution that manages complex models by recursive application of Markov regenerative analysis and numerical composition of monovariate distributions.

Keywords: stochastic workflow · response time distribution · structured model · compositional evaluation · stochastic ordering.

1 Introduction

A workflow is an orchestration of concurrent and sequential activities, mainly shaped by constructs of split/join, choice/merge, and sequence, occasionally including dependencies that break well-formed nesting and repetitions that produce transient cycles [29]. This abstraction fits a large variety of material and digital processes, in multiple contexts such as supply chain management [18], administration [1], composite web services [12], cloud "functions as a service" [33].

When the model is associated with a measure of probability, quantitative evaluation may provide measures of interest for different stages of development and operation [28, 8, 14, 7], supporting the achievement of a tradeoff between some of them, such as the average response time, the subtask dispersion, and the energy consumption [26]. In particular, the Cumulative Distribution Function (CDF) of the end-to-end response time is relevant for the evaluation of the expected reward under a Service Level Agreement (SLA) with soft deadlines and penalty functions [16, 27]. In this case, a stochastically ordered approximation of the CDF produces a safe approximation of the expected reward.

However, practical feasibility faces a difficult combination of recurring complexities: activity durations follow general (i.e., non-Exponential) probability distributions (GEN), often supported within firm bounds enforced by design or by contract, which cast the problem in the class of non-Markovian processes [11]; the interleaving of actions in concurrent sub-workflows leads to explosion of the state space and complex dependencies due to the overlap among concurrent activities with GEN duration [6].

Compositional solution can address both complexities by avoiding explicit representation of interleavings, limiting state space explosion and simplifying the structure of underlying stochastic processes of individual components [6]. In [36], mean time and standard deviation of the completion time of a workflow of activities with GEN duration composed by fork/join, sequence, and repetition are derived through an efficient bottom-up calculus, which is extended in [22] for the special case of Continuous Phase (CPH) durations. In [2], the completion time of an acyclic attack tree with CPH delays is evaluated by repeatedly composing CPH distributions in a bottom-up approach, with possible approximation to compress their representation to maintain a bounded number of phases. In [10], the response time of an acyclic workflow obtained by well-formed nesting of activities with GEN durations is evaluated in a two-step approach, first applying Markov regenerative analysis to nested sub-workflows identified so as to have a limited degree of concurrency, and then repeatedly composing the resulting monovariate distributions bottom-up to obtain the workflow response time.

In this paper, we propose a compositional technique for efficient evaluation of the response time CDF of complex workflows consisting of activities with GEN durations composed through sequence, choice/merge, split/join, and *repetition blocks*, with *unbalanced split and join constructs that may break the structure of well-formed nesting.* Workflows are specified using a higher level formalism defined in terms of stochastic Petri nets, that permits decomposition of the model into a hierarchy of sub-workflows with positively correlated response times. This guarantees a stochastically ordered approximation of the end-to-end response time when completion times of intermediate sub-workflows are approximated by a stochastically ordered fitting distribution and when activities shared among multiple sub-workflows are replicated as independent random variables so as to reduce dependencies. These approximations allow an efficient hierarchical solution approach that manages complex models by recursive application of Markov regenerative analysis and numerical composition of monovariate distributions.

The rest of the paper is organized in four sections, addressing: specification of models and their representation as a *structure tree* (Section 2); decomposition of models into a hierarchy of sub-workflows identified in the structure tree by heuristics trading approximation for complexity, and re-composition of the end-to-end response time distribution (Section 3); results of numerical experimentation (Section 4); and conclusions (Section 5). Theorem proofs and other details are deferred to the Appendix (Section 6).

2 Modeling workflows with structured STPNs

We specify workflows with stochastic activity durations using a formalism that constrains expressivity of a class of stochastic Petri nets (Sections 2.1 and 2.2) so as to ensure positive correlation among their response times and to enable derivation of a structured representation of the model (Section 2.3).

2.1 Stochastic Time Petri Nets (STPNs)

Stochastic Time Petri Nets (STPNs) model concurrent timed systems: transitions represent the execution of activities, tokens within places account for the system logical state, and directed arcs from input places to transitions, and from transitions to output places, define precedence relations among activities [15]. A transition is enabled if each of its input places contains at least one token; at newly enabling, a transition samples a time-to-fire from a CDF with support between an Earliest Firing Time (EFT) and a Latest Firing Time (LFT); upon firing, it removes a token from each input place and adds one token to each output place. The choice among transitions with equal time-to-fire is solved by a random switch determined by probabilistic weights.

As shown in Fig. 1a, places are represented as circles, tokens as dots inside places, immediate (IMM) transitions (i.e., with zero time-to-fire) as thin bars (e.g., `as1`), deterministic (DET) transitions (i.e., with nonzero deterministic time-to-fire) as gray bars (e.g., `z1`), EXP transitions as white bars (e.g., `v1`), and other GEN transition as black bars (e.g, `q`). Where necessary, labels indicate rates, firing intervals $[EFT, LFT]$, CDF types (e.g., *uniform* or *expolynomial*).

2.2 STPN blocks

Workflows with stochastic durations can be specified using a fragment of the expressivity of STPNs, which is sufficient to represent a variety of workflow control

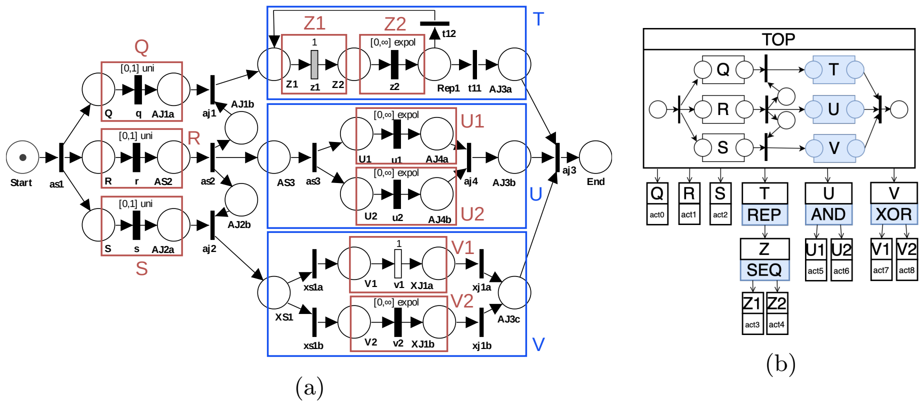


Fig. 1: (a) Workflow STPN and (b) its structure tree (composite blocks in blue).

patterns [29, 37] while making explicit a structure of composition enabling efficient timed analysis. To this end, we specify workflows by recursive composition of *blocks*, each defined as an STPN with a single *initial place* and a single *final place*. The execution of a block starts when a token is added to the initial place, and it eventually terminates, with probability 1 (w.p.1), when a token reaches the final place. Blocks compose elementary STPN transitions through nested constructs of concurrent (split/join) and sequential (sequence, choice/merge, repeat) behavior, or by acyclic compositions that break well-formed nesting through unbalanced fork and join operations (simple split, simple join):

$$\begin{aligned}\textsc{Block} := \textsc{Act} \mid \textsc{Seq}\{\textsc{Block}_1, \dots, \textsc{Block}_n\} \mid \text{AND}\{\textsc{Block}_1, \dots, \textsc{Block}_n\} \\ \mid \text{XOR}\{\textsc{Block}_1, \dots, \textsc{Block}_n, p_1, \dots, p_n\} \mid \textsc{Repeat}\{\textsc{Block}, p\} \\ \mid \text{DAG}\{\textsc{Block}_1, \dots, \textsc{Block}_n\}\end{aligned} \tag{1}$$

Act is an elementary *activity* represented by an STPN with a single transition with GEN duration connecting the initial and final places (e.g., Q in Fig. 1a).
Seq$\{\textsc{Block}_1, \dots, \textsc{Block}_n\}$ is the *sequence* of n blocks $\textsc{Block}_1$, ..., $\textsc{Block}_n$ (e.g., Z in Fig. 1a).
XOR$\{\textsc{Block}_1, \dots, \textsc{Block}_n, p_1, \dots, p_n\}$ is made of an initial immediate random *exclusive choice*, with probabilities $p_1, \dots, p_n$, among n alternative blocks $\textsc{Block}_1$, ..., $\textsc{Block}_n$, each connected to a final IMM *simple merge* transition (e.g., V in Fig. 1a). XOR is said to be *balanced*, meaning that all the concurrent paths started at the initial split are terminated at the final join.
AND$\{\textsc{Block}_1, \dots, \textsc{Block}_n\}$ is a balanced split-join made of an initial IMM *parallel split* transition that forks execution along n concurrent blocks $\textsc{Block}_1$, ..., $\textsc{Block}_n$ and a final IMM *synchronization* transition that terminates the block (e.g., U in Fig. 1a).
Repeat$\{\textsc{Block}, p\}$ is a *structured cycle* that executes a body $\textsc{Block}$ and then repeats with constant probability $p > 0$ or terminates with probability $1 - p$ (e.g., $\{T\}$ in Fig. 1a).
DAG$\{\textsc{Block}_1, \dots, \textsc{Block}_n\}$ is the composition of blocks $\textsc{Block}_1$, ..., $\textsc{Block}_n$ in a Directed Acyclic Graph (DAG) with single initial and final places, with blocks of *simple split* [29], made of an IMM transition with a single input place and multiple output places, and *simple join*, made of an IMM transition with multiple input places and a single output place. Note that, since *simple split* and *simple join* are not necessarily balanced, a DAG can break well-formed nesting of concurrent blocks. A DAG is termed *minimal* if it cannot be reduced by composition operators $\textsc{Seq}$, XOR or AND (e.g., in Fig. 1a, Q, R, S, T, U, and V are composed in a minimal DAG with initial and final places `Start`, `End`, by means of simple splits `as1`, `as2`, and simple joins `aj1`, `aj2`, `aj3`).

By definition, each model specified as a composition of blocks can be translated into a unique STPN. Conversely, blocks do not cover all the expressivity of STPNs. In particular, since choices are expressed only by IMM transitions within balanced XOR or $\textsc{Repeat}$ blocks, a model cannot represent a race selection where a choice is determined by values sampled by concurrent activities, e.g., early preemption of a timed activity that may occur in a timeout mechanism. As a positive consequence, this restriction also rules out *anomalies* where the early completion of some intermediate step can result in a longer workflow duration, providing the basis to prove positive correlation among completion times of different intermediate points in the workflow.

2.3 Structure tree

Based on the grammar of Eq. (1), a workflow can be decomposed as a *structure tree* $S = \langle N, E, n_0 \rangle$: N is the set of nodes (blocks); E is the set of directed edges (connecting each block with its component blocks); n_0 is the root node. In Fig. 1b, a block is represented as a box labeled with the block name; the box is labeled also with the activity name (for Act blocks) or block type (for Seq, AND, XOR, Repeat blocks), or contains places and transitions connecting the component blocks (for DAG blocks). Hierarchical graphs with single-entry single-exit blocks are inspired by *program structure trees* [17] and *process structure trees* [34]: similarly, we ensure that the structure tree is unique and robust with respect to local changes (i.e., modifying a sub-workflow in the STPN affects only its subtree in the structure tree) by using maximal blocks (e.g., SEQ blocks with as many components as possible) and by matching DAG blocks with lowest priority (i.e., Seq or AND are used instead of DAG nodes, if possible).

3 Compositional evaluation of workflows response time

We evaluate the end-to-end response time CDF of a workflow by composition of the results of separate analyses of a hierarchy of sub-workflows. In so doing, we repeatedly apply both straight *numerical combination* of monovariate CDFs (Section 6.2 in the Appendix) and *Markov regenerative transient analysis* (Section 3.1) so as to leverage their different strengths. Numerical combination turns out to be efficient in the composition of independent sub-workflows through well-nested operators (AND, XOR, Seq, Repeat), but it is not feasible for sub-workflows with common dependencies (DAG). Markov regenerative analysis suffers from the degree of concurrency among activities with GEN durations, and its efficient implementation requires that sub-workflow durations be represented in analytic form, which may require approximated fitting of numerical results.

The structure tree is used to aggregate model components and to select solution techniques according to heuristics that trade approximation for complexity (Sections 3.2 and 3.3) while ensuring that the final result is a stochastic upper bound of the exact CDF of the end-to-end workflow response time (Section 3.4).

3.1 Regenerative transient analysis

The structure of a workflow naturally leads to concurrent execution of multiple activities, which in most practical cases are generally distributed (GEN), often within a bounded support. In this setting, the marking process $\{ M(t), t \geqslant 0 \}$ of the STPN representing a sub-workflow is a Markov Regenerative Process (MRP) if a new *regeneration point* (i.e., a time instant where the Markov property is satisfied) is eventually reached w.p.1 from any state [20]. In this case, transient probabilities $P_{ij}(t)$ of each marking j and initial regeneration i (including the enabling time of each GEN transition [25]) can be evaluated by numerical integration of Generalized Markov Renewal equations $P_{ij}(t) = L_{ij}(t) + \sum_{k\in\mathcal{R}} \int_0^t dG_{ik}(u)\, P_{kj}(t-u)$ for all i in the set of reachable regenerations $\mathcal{R}$ and for all j in the set of markings $\mathcal{M}$, where the *global kernel* $G_{ik}(t) := P\{X_1 = k,\, T_1 \leqslant t \mid X_0 = i\}$ characterizes the next regeneration point $T_1 \geqslant 0$ and regeneration $X_1 \in \mathcal{R}$, while the *local kernel* $L_{ij}(t) := P\{M(t) = j, T_1 > t \mid X_0 = i\}$ defines transient probabilities of the process until the next regeneration point.

In turn, kernels can be evaluated numerically if at most one GEN transition is enabled in each state [6], or if multiple GEN transitions are enabled concurrently but the number of firings between regeneration points is bounded [5]. For this larger class of MRPs, kernels can be evaluated using *stochastic state classes* [15], which encode the marking, joint Probability Density Function (PDF), and support of the times-to-fire of enabled transitions after each sequence of firings between any two regeneration points. This joint PDF is continuous, with piecewise representation over Difference Bounds Matrix (DBM) zones [9], and can be evaluated in closed-form by the ORIS tool (or the Sirio Java library) [24] provided that each transition has expolynomial PDF [32].

The complexity of regenerative transient analysis of an STPN can be efficiently estimated by *nondeterministic analysis* of the underlying TPN, which is sufficient to identify the set of feasible behaviors of the model while avoiding the complexity of evaluation of their measure of probability. This is obtained by enumeration of a *state class graph*, encoding the continuous set of executions of the STPN into a discrete representation [4, 35]: each vertex is a *state class* $S = \langle m, D\rangle$ including a marking m and a *DBM zone* D [13], i.e., a continuous set of values for the times-to-fire of enabled transitions. For each transition t that can fire first, the graph includes a directed edge (S, t, S') from S to the state class $S' = \langle m', D'\rangle$ with marking m' after the firing and the zone D' of reachable times-to-fire. The graph is finite under fairly general conditions (requiring that the number of reachable markings be finite and the earliest and latest firing times of transitions be rational values [35]), it permits detection of regeneration points (as classes where each GEN transition is newly enabled, disabled, or enabled since a deterministic time), and it makes explicit the number of transitions between regeneration points and the degree of concurrency among GEN timers.

3.2 Complexity heuristics

Complexity factors. Regenerative transient analysis incurs different factors of complexity in the enumeration of stochastic state classes, derivation of the local and global kernels, and solution of the Markov renewal equations:

– The results of [30] show that the number of stochastic state classes depends on the number of concurrently enabled GEN transitions, the number of firings after which a GEN transition is persistent (i.e., continuously enabled), and the number of expmonomial terms of the PDFs of the GEN transitions. In our approach, the number of stochastic state classes is kept limited by decomposing a workflow into sub-workflows that are separately analyzed.

- The results of [30] also show that the number of DBM zones and the number of expmonomial terms of the joint PDF of each stochastic state class depend on the same factors as the number of stochastic state classes. In particular, at each firing, the number of DBM zones increases polynomially with the number of persistent transitions, the number of expmonomial terms increases linearly with the polynomial degree of the joint PDF, and, in turn, if the analytical form of the joint PDF contains no EXP factor, the polynomial degree increases linearly with number of fired/disabled transitions.
  In our approach, the factors of complexity of stochastic state classes are kept limited not only by decomposing a workflow into sub-workflows, but also by approximating the numerical form of the response time distribution of a sub-workflow with a piecewise PDF made of EXP terms (the approximation is needed when a sub-workflow is analyzed in isolation, either through regenerative transient analysis or through numerical analysis, and then regenerative transient analysis of a higher-level sub-workflow has to be performed).
- According to [15], the derivation of the kernels and the solution of the Markov renewal equations have linear complexity in the number of stochastic state classes and in the number of DBM zones and expmonomial terms of the joint PDF of each stochastic state class. Moreover, the derivation of the kernels and the solution of the Markov renewal equations also have linear and quadratic complexity, respectively, in the number of time points, i.e., the number of times the time step is contained in the analysis time limit. In our approach, the derivation of the local kernel is limited to the evaluation of $L_{if}(t)$, where i is the initial regeneration and f is the final absorbing marking.

Complexity measures. To estimate the complexity of regenerative transient analysis of an STPN, we use the state class graph of the underlying TPN to compute the maximum number c of GEN transitions concurrently enabled in a state class (*concurrency degree*) and the maximum length r of paths of a regeneration epoch (*epoch length*), i.e., paths between regenerative state classes. However, the state class graph may be huge for complex blocks with high values of c and r, and would not tell how much of the complexity depends on the structure of the block itself and how much on the blocks it contains, which instead is relevant to decide how to decompose the block. To cope with both aspects, for each block b, we compute upper bounds C and R on c and r, respectively, and we also compute the concurrency degree $\bar{c}$ and the epoch length $\bar{r}$ of a *simplified* block $\bar{b}$, obtained by replacing each composite block of b with an (elementary) activity block. To this end, we perform a bottom-up visit of the structure tree:

- At the bottom level, we perform nondeterministic analysis of the TPN of each composite block, computing the tuple $\langle c, r, t_{\min}, t_{\max}\rangle$, where $t_{\min}$ and $t_{\max}$

are the minimum and the maximum execution time of the block, respectively, and can be derived as the minimum and the maximum duration, respectively, of paths between the initial and the final state class [35].

- At the next higher level, we perform nondeterministic analysis of the TPN of the simplified block $\bar{b}$, obtained by replacing each composite block b' of b with an activity block with duration interval equal to the min-max execution interval of b' (computed at the previous step). Evaluation of the complexity measures on the resulting state class graph $\bar{\Gamma}$ yields the tuple $\langle \bar{c}, \bar{r}, t_{\min}, t_{\max} \rangle$. Upper bounds on c and r are $C = \max_{S \in \Omega}\{\sum_{t \in E_S} C_t\}$ and $R = \max_{\rho \in \Psi_r}\{\sum_{e \in G_\rho} R_e\}$, respectively, where Ω is the set of state classes of $\bar{\Gamma}$, E_S is the set of transitions enabled in state class S, C_t is the concurrency degree upper bound of the block corresponding to transition t (1 for activity blocks), Ψ_r is the set of paths of $\bar{\Gamma}$ between regenerative state classes, G_ρ is the set of edges of path ρ, and R_e is the epoch length upper bound of the block corresponding to the transition of edge e (1 for activity blocks).
- Evaluation is repeated until the tuple $\langle \bar{c}, \bar{r}, C, R, t_{\min}, t_{\max} \rangle$ is computed for each composite block of the structure tree (thus also for the root block).

Then, we define the *complexity heuristics*: a block b (a simplified block $\bar{b}$) is *easy* to analyze if both C and R ($\bar{c}$ and $\bar{r}$) are not larger than some thresholds Θ_c and Θ_r, respectively, and *complex* otherwise, e.g., for the workflow STPN of Fig. 1, R is infinite, due to the concurrency between the Repeat block T and blocks S, U, and V (in fact, for the simplified STPN obtained replacing T and V with an activity block each, we have $\bar{c} = 3$ and $\bar{r} = 7$). The goal of our compositional analysis is to reduce the workflow complexity, e.g., by analyzing block T in isolation and replacing it with a transition approximating its duration.

3.3 Analysis heuristics

In the evaluation of the response time CDF $\Phi_b(t)$ of a block b, we consider four *actions*, each introducing a different approximation.

Action 1 (numerical analysis): If b is well-structured, i.e., neither it is a DAG or Repeat block nor it contains DAG or Repeat blocks (e.g., U in Fig. 1), evaluate $\Phi_b(t)$ by numerical analysis.

Action 2 (regenerative transient analysis): If the execution time CDF of each activity block in b is expressed in analytical form (e.g., T in Fig. 1), evaluate $\Phi_b(t)$ through regenerative transient analysis of the STPN of b, otherwise (i.e., if b has some activity CDF expressed in numerical form, e.g., the workflow of Fig. 1 after the response time CDF of T is evaluated separately) replace each numerical CDF with the analytical form of a stochastic upper bound CDF (by Lemma 2), and evaluate a stochastic upper bound on $\Phi_b(t)$ through regenerative transient analysis of the STPN of the resulting block (by Lemma 3).

Action 3 (inner block analysis): Evaluate the response time CDF of a composite block c (see Fig. 6a) contained in block b (through some action α_1), replace

c with an activity block having the computed CDF as execution time CDF (see Fig. 6b), and compute the response time CDF of the obtained block b' (through some action α_2). Note that α_1 and α_2 may yield $\Phi_b(t)$ (e.g., if both are action 1) or a stochastic upper bound on $\Phi_b(t)$ (e.g., if α_1 is action 1 and α_2 is action 2).

Action 4 (inner block replication): If b is a DAG block, replicate some predecessors of a block to evaluate its response time independently of the rest of the DAG (replicated blocks are identical). Specifically, let $G = (V, E, v_I, v_F)$ be the DAG where V is the set of vertices (i.e., blocks of b) plus (fictitious) zero-duration initial vertex v_I and final vertex v_F (not shown in Fig. 1), and E is the set of edges (i.e., precedence relations between blocks): identify vertex $v \in V \setminus \{v_I, v_F\}$ (e.g., block T in Fig. 1); let the set K of vertices in $V \setminus \{v_I, v_F\}$ that are predecessors both of v and of some node $u \in V$ not predecessor of v (i.e., $K = \{R\}$); replicate the vertices in K and the edges to/from nodes in K (i.e., add R' to V; add $v_I \rightarrow R'$ and $R' \rightarrow T$ to E); evaluate the response time CDF of v (by some action, see Fig. 6c) and replace it with an activity block with the computed CDF as execution time CDF (see Fig. 6d); and, evaluate the response time CDF of the obtained block (by some action), which is a stochastic upper bound on the response time CDF $\Phi_b(t)$ of the original block b (by Lemma 4).

We define *analysis heuristics* to visit the structure tree and repeatedly select an action until a safe approximation of the workflow response time is evaluated. To exploit our complexity heuristics, which characterizes both the complexity of the structure of a workflow and the complexity of the blocks that it contains, we consider three analysis heuristics that perform a *top-down* visit of the structure tree. Nevertheless, the approach is open to the definition of different heuristics.

Overall, if a block is, or can be reduced to, a well-structured composition of independent sub-workflows, then numerical analysis guarantees efficient and accurate evaluation. Conversely, Repeat blocks can be evaluated in isolation through regenerative transient analysis when their parallel composition with other blocks prevents the occurrence of regenerations. Finally, DAG blocks, representing dependent sub-workflows, can be evaluated through regenerative transient analysis, operating some simplification if the DAG is too complex to analyze (i.e., analyzing some block or some sub-workflow in isolation).

Specifically, **analysis heuristics 1** operates as follows on a visited block b:

1. If b is well-structured, then select action 1 (numerical analysis).
2. If b is a Seq or an AND or an XOR block, and contains DAG or Repeat blocks at some hierarchy level, then select action 3 (inner block analysis) as many times as the number of composite blocks of b (which are replaced with an activity block each) and then select action 1 (numerical analysis).
3. If b is a Repeat block, use the analysis heuristics to select the next action:
   (a) if b is easy to analyze, select action 2 (regenerative transient analysis);
   (b) otherwise, select action 3 (inner block analysis) to compute (through some action) the response time CDF of the block repeated by the loop.
4. If b is a DAG block, use the analysis heuristics to select the next action:

(a) if both b and the simplified block $\bar{b}$ are easy to analyze, then select action 2 (regenerative transient analysis);
(b) otherwise, until block b becomes easy to analyze, repeatedly select action 4 (inner block replication), each time analyzing in isolation one of the sub-workflows AND-joined by the final IMM transition of b.

For instance, the DAG in Fig. 1 is too complex to analyze: heuristics 1 performs regenerative transient analysis of the sub-workflow $\{Q, R, T\}$ (by replicating R), and then performs regenerative transient analysis of the obtained block.

To evaluate how approximating intermediate numerical CDFs impacts result accuracy and computational complexity with respect to decoupling dependent sub-workflows, we consider **analysis heuristics 2**, a variant of heuristics 1 that manages complex DAG blocks (point 4a) by repeatedly performing first action 3 (inner block analysis, replacing complex composite blocks with activity blocks), and then action 4, until the DAG is easy to analyze.

Finally, to show the efficacy of numerical analysis in the evaluation of well-structured workflows, we consider **analysis heuristics 3**, another variant of heuristics 1 that performs regenerative transient analysis in cases where heuristics 1 would perform numerical analysis.

3.4 Approximation safety

We now ensure that our compositional analysis method is safe when workflows are used to guarantee soft deadlines of SLAs. Our proofs (in the Appendix) hinge on the idea of *stochastic order* and on the following lemma on the order of independent replication of positively correlated random variables (r.v.s) [3].

Definition 1 (Stochastic order). *Given two random vectors* $\mathbf{X}_1$ *and* $\mathbf{X}_2$*, we say that "*$\mathbf{X}_1$ *is smaller than* $\mathbf{X}_2$*" (*$\mathbf{X}_1 \leqslant_{st} \mathbf{X}_2$*), if* $E[f(\mathbf{X}_1)] \leqslant E[f(\mathbf{X}_2)]$ *for all monotone nondecreasing functions* f*. For scalar* X_1 *and* X_2 *with CDFs* $F_1(x)$ *and* $F_2(x)$*, respectively, this is equivalent to* $F_1(x) \geqslant F_2(x)$ *for all* x*.*

Lemma 1 (Order of independent replicas under positive correlation). *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be a vector of positively correlated r.v.s, i.e.,* $\mathrm{Cov}[f(\mathbf{X}), g(\mathbf{X})] \geqslant 0$ *holds for all monotone nondecreasing* $f, g \colon \mathbb{R}^n \to \mathbb{R}$*. Then,* $\overline{\mathbf{X}} \geqslant_{st} \mathbf{X}$*, where* $\overline{\mathbf{X}}$ *is a vector of independent r.v.s with* $\overline{X}_i \sim X_i$ *for all* i*.*

In our regenerative analysis, numerical CDFs are replaced with analytical stochastic upper bound CDFs (which guarantee stochastic order for each known point of the numerical CDFs). The next lemma proves that such bound can be a piecewise CDF combining a shifted truncated EXP (*body*) and a shifted EXP (*tail*). The approximant accuracy could be improved by considering multiple pieces for the body (e.g., to better approximate multimodel CDFs).

Lemma 2 (Stochastic upper bound CDF). *Given a r.v.* X *with numerical CDF* $F(x)$ *with* $x \in D = \{a, a+\delta, \ldots, a+(L-1)\,\delta\}$*,* $a \in \mathbb{R}_{\geqslant 0}$*,* $\delta \in \mathbb{R}_{>0}$*, and* $L \in \mathbb{N}$*, let* $\hat{X}$ *be the r.v. with CDF* $\hat{F}(x)$ *s.t.* $\hat{F}(x) = 0\ \forall\, x < d$*,* $\hat{F}(x) = 0.75\,(1 -$

$e^{-\lambda_b(x-d)})/(1-e^{-\lambda_b(q_3-d)})$ $\forall\, x \in [d, q_3]$, $\hat{F}(x) = 0.25(1-e^{-\lambda_t(x-q_3)}) + 0.75$ $\forall\, x \in [q_3, \infty)$, where d is equal to a if $F(x)$ starts with downward concavity and equal to the abscissa of the intersection of the x-axis with the line tangent to the inflection point of $F(x)$ if $F(x)$ starts with upward concavity, q_3 is the third quartile of X, λ_b is the minimum of the values that maximize $\hat{F}(x)$ and satisfy $\hat{F}(x) \leqslant F(x)$ $\forall\, x \in D \cap [d, q_3]$, and λ_t is the minimum of the values that maximize $\hat{F}(x)$ and satisfy $\hat{F}(x) \leqslant F(x)$ $\forall\, x \in D \cap (q_3, \infty)$. Then, $\hat{X} \geqslant_{st} X$.

In our inner block analysis, a node n in the structure tree is replaced with an activity block with duration stochastically larger than the response time of n. The next lemma proves that, after this approximation, the response time of the obtained workflow is stochastically larger than the actual response time.

Lemma 3 (Stochastic order of inner block analysis). *Let $S = (N, E, n_0)$ be the structure tree of a workflow with root node $n_0 \in N$, and let $T(n)$ be the response time of the subtree rooted in $n \in N$. If n is replaced with n' s.t. $T(n) \leqslant_{st} T(n')$, yielding the new structure tree $S' = (N', E', n'_0)$, then $T(n_0) \leqslant_{st} T(n'_0)$.*

In our inner block replication, ancestors of a vertex v are replicated in a DAG block to evaluate the response time of v independently of the rest of the DAG. The next lemma proves that, also after this approximation, the response time of the obtained workflow is stochastically larger than the actual response time.

Lemma 4 (Stochastic order of inner block replication). *Given a DAG block $G = (V, E, v_I, v_F)$ and a vertex $v \in V$, let $T(v)$ be the response time of v, let K be the set of vertices in $V \setminus \{v_I, v_F\}$ that are predecessors both of v and of some node $u \in V$ not predecessor of v, let F be the set of edges in E to/from a node in K, and let $G' = (V', E', v'_I, v'_F)$ be the DAG s.t. V' includes all vertices in V plus a new node k' with $T(k') \sim T(k)$ $\forall\, k \in K$, and E' includes all edges in E plus an edge to/from each new node k' for each edge to/from the corresponding node $k \in K$. Then, $T(v'_F) \geqslant_{st} T(v_F)$.*

4 Experimentation

In this section, we answer the following questions on our proposed approach:

Q1. Is the approach feasible for the considered concurrency structures?
Q2. Is the approach accurate with respect to a simulated ground truth?
Q3. Does the approach obtain accurate results in reasonable times?

To this end, we consider eight models combining four structures that gradually increase the workflow complexity, evaluating how the approximation of intermediate numerical results and the replication of dependent events affect result accuracy and computational complexity. For each model, we compare a ground truth with the results of our analysis heuristics and simulation. For our complexity heuristics, we consider thresholds $\Theta_c = 3$ and $\Theta_r = 10$ on the concurrency degree and the epoch length, respectively, and we consider activity durations with uniform CDF over $[0, 1]$. Experiments are performed using a single core of an Intel Xeon Gold 5120 CPU (2.20 GHz) equipped with 32 GB of RAM.

4.1 Experimentation models

Fig. 2 shows the four structures used to build the experimentation models:

Simple DAG (Fig. 2a) has concurrency degree 3 and epoch length 8. Thus, it can be efficiently analyzed by regenerative transient analysis.

Complex DAG (Fig. 2b) has concurrency degree 5. It cannot be analyzed as a whole and needs to be decomposed by one of the analysis heuristics.

Complex AND (Fig. 2c) is a well-structured tree, with two instances of simple DAG as leaves. Once the latter are analyzed by regenerative transient analysis, the resulting tree can be analyzed numerically (analysis heuristics 1 and 2), or, given that it has concurrency degree 4, regenerative transient analysis can be applied to blocks A and F, and then to the resulting model (analysis heuristics 3).

Nested Repetitions (Fig. 2d) has two nested REPEAT blocks, with an instance of (simple or complex) DAG and of complex AND as leaves: once the latter are

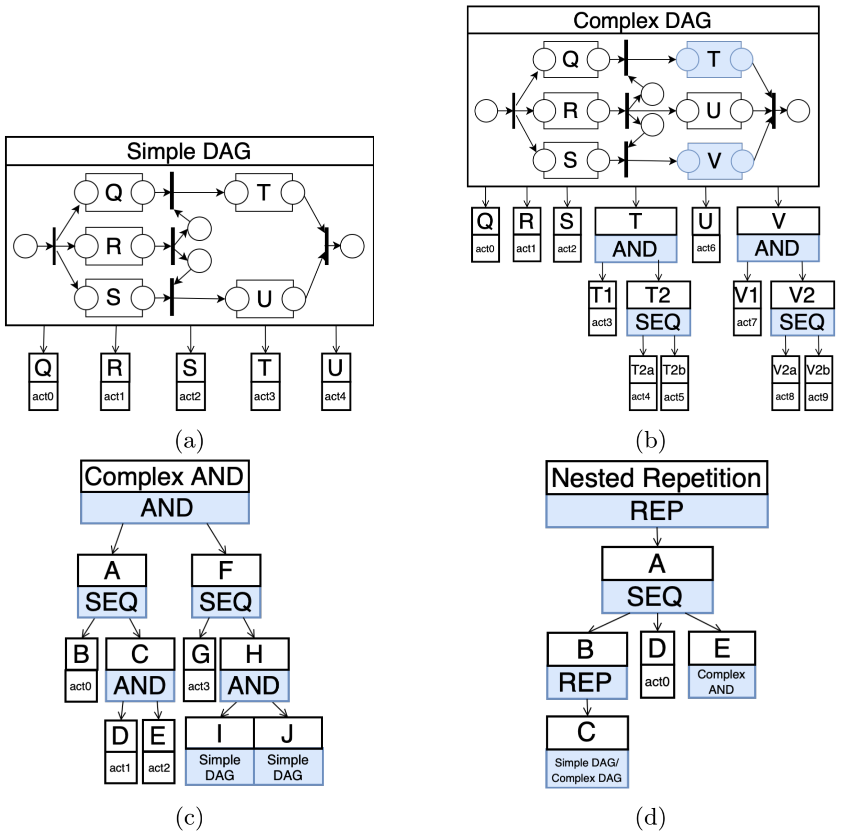


Fig. 2: **Structure elements used to compose experimentation models.**

analyzed in isolation, the resulting model has concurrency degree 1 and path length 3, and can be analyzed by regenerative transient analysis (all heuristics).

Fig. 3 shows the models obtained combining the structure elements of Fig. 2.

Models 1a and 2b (Fig. 3a) are well-structured trees with two instances of (simple and complex, respectively) DAG and an instance of complex AND as leaves: once the latter composite blocks are analyzed, the resulting tree can be analyzed numerically (analysis heuristics 1 and 2), or, given that it has concurrency degree 4, regenerative transient analysis can be applied first to blocks A and K, and then to the resulting model (analysis heuristics 3).

Models 2a and 2b (Fig. 3b) are well-structured trees with an instance of (simple and complex, respectively) DAG and of Nested Repetitions as leaves: as for models 1 and 2, once the latter composite blocks are analyzed, the resulting tree can be analyzed numerically (analysis heuristics 1 and 2), or, given that it

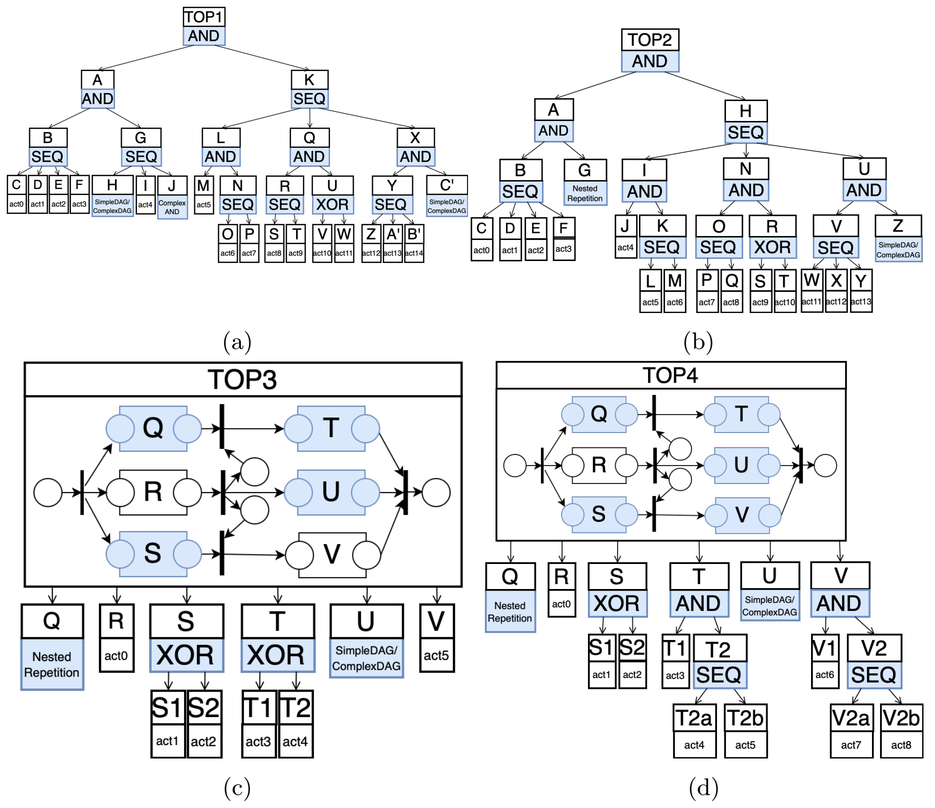


Fig. 3: **Four models used for experimentation**. Each model combines some of the structures defined in Fig. 2 and is tested in a simple variant, using instances of Simple DAG, and a complex variant, using instances of Complex DAG.

has concurrency degree 4, regenerative transient analysis can be applied first to blocks A and H, and then to the resulting model (analysis heuristics 3).

Models 3a and 3b (Fig. 3c) consist of a top DAG made of an instance of (simple and complex, respectively) DAG, one of Nested Repetitions, and two well-structured sub-trees. Once the instance of Nested Repetitions is evaluated, the model is still too complex and must be decomposed.

Models 4a and 4b (Fig. 3d) are variants of models 3a and 3b, respectively, with more complex well-structured trees. Once the instance of Nested Repetitions is evaluated, the model is still too complex and must be decomposed.

4.2 Experimentation results

For each workflow of Fig. 3, a ground truth is computed by performing simulation of the corresponding STPN through the SIRIO library of the ORIS tool [24, 31], increasing the number of simulation runs by 100 000 at a time until the Jensen-Shannon (JS) divergence [21, 23] between the PDFs of the last two computed response time CDFs is lower than 0.0001, which occurs for 500 000 runs. The JS divergence between two PDFs f_a and f_b is $D_{\mathrm{JS}}(f_a \,||\, f_b) := 0.5\, D_{\mathrm{KL}}(f_a \,||\, Z) + 0.5\, D_{\mathrm{KL}}(f_b \,||\, Z)$, where $Z(t) := 0.5\,(f_a(t) + f_b(t))\ \forall t \in \Omega$ is the random variable that averages the input variables, Ω is a set of equidistant time points covering the support of f_a and f_b, and $D_{\mathrm{KL}}(\cdot \,||\, \cdot)$ is the Kullback-Leibler (KL) divergence [21, 23] defined as $D_{\mathrm{KL}}(f_a \,||\, f_b) = \sum_{t \in \Omega} f_a(t) \cdot \log(f_b(t) / f_a(t))$.

Table 1 reports the computation times and the JS divergence from the ground truth achieved by the analysis heuristics 1, 2, and 3, and by a simulation having computation times comparable with times of heuristics 1, and Fig. 4 shows the response time PDFs used to compute the JS divergence. For models 1a, 1b, 2a and 2b, heuristics 1 outperforms heuristics 3 in terms of both accuracy and complexity, achieving JS divergence lower by one or two orders of magnitude and computation times lower by one order of magnitude, proving the efficacy of analyzing well-structured trees through numerical analysis rather than through

Model	Computation Times					JS divergence from the GT			
	GT	S	H1	H2	H3	S	H1	H2	H3
1a	2027.1 s	1.5 s	1.5 s	1.5 s	32.7 s	0.196 91	0.000 27	0.000 27	0.033 26
1b	2711.3 s	1.0 s	1.0 s	13.8 s	10.6 s	0.190 46	0.000 84	0.003 15	0.036 81
2a	2028.7 s	0.9 s	0.8 s	0.8 s	3.2 s	0.193 85	0.003 63	0.003 63	0.060 11
2b	2723.3 s	1.4 s	1.2 s	13.9 s	3.2 s	0.184 64	0.006 62	0.009 43	0.043 10
3a	1566.8 s	2.2 s	1.9 s	3.8 s	2.4 s	0.041 05	0.026 89	0.025 69	0.082 92
3b	2232.2 s	4.2 s	3.9 s	16.8 s	2.5 s	0.081 02	0.026 60	0.025 48	0.081 73
4a	1803.9 s	1.8 s	1.7 s	72.6 s	5.9 s	0.194 71	0.019 72	0.037 30	0.075 77
4b	2534.5 s	2.6 s	2.6 s	72.0 s	6.0 s	0.202 84	0.021 98	0.039 41	0.076 71

Table 1: For each model of Fig. 3, computation time and JS divergence from the ground truth (GT) of simulation (S), analysis heuristics 1 (H1), 2 (H2), 3 (H3). For each model, green cells indicate the best (i.e., lowest) values of computation time and JS divergence, while red cells indicate the worst (i.e., largest) values.

regenerative transient analysis. This gain is less evident for models 3a, 3b, 4a and 4b, which replace large part of high-level well-nested structures of models 1a, 1b, 2a and 2b with DAG blocks.

Heuristics 1 and 2 (differing in the way complex DAG blocks are managed) achieve the same accuracy and computation time on models 1a and 2a, because these models include simple DAG blocks. For the remaining models, heuristics 1 achieves comparable or slightly lower accuracy and significantly lower computation time, indicating that, as expected, replicating dependent events yields less complex models to analyze with respect to evaluating blocks in isolation and approximating intermediate numerical results. Moreover, for the considered model structures and stochastic parameters, event replication does not introduce more approximation than analytical fitting of intermediate numerical results.

With respect to simulation having computation time comparable to that of heuristics 1, all analysis heuristics achieve better accuracy with JS divergence lower by at least one order of magnitude, and up to three orders for heuristics 1. Note that simulation could be optimized to improve both result accuracy and computational complexity, while the analysis is in any case guaranteed to provide a stochastic upper bound on the workflow response time CDF.

Overall, the evaluation shows that the proposed approach is feasible for complex workflows, and achieves sufficient accuracy in reasonable time with respect to the ground truth. In particular, analysis heuristics 1 achieves JS divergence with order of magnitude between 10^{-4} and 10^{-2}, in at most 3.9 s.

5 Conclusions

We proposed a compositional approach to efficiently compute a stochastic upper bound on the response time CDF of complex workflows. The workflow is specified using structured STPNs to enable hierarchical decomposition into subworkflows, exploiting heuristics that apply either numerical analysis (if feasible) or regenerative transient analysis, taking into account different tradeoffs between solution accuracy and complexity. When evaluated on a suite of synthetic models of increasing complexity, the approach achieves sufficient accuracy in a limited computation time with respect to a ground truth obtained by simulation.

The approach is open to many extensions: the model could be extended with other constructs and dependencies among the execution times of activities, possibly affecting not well-formed nesting; performance could be improved by optimizing regenerative transient analysis based on the specific class of models and the specific reward to evaluate; other solution techniques could be integrated to analyze some block; to fit numerical distributions, other analytical approximants could be considered in the class of expolynomial functions, or in the class of piecewise CPHs over bounded supports [19]; and, applicability could be tested in various relevant domains where the evaluation of deadlines missed within a given time requires the computation of the response time CDF.

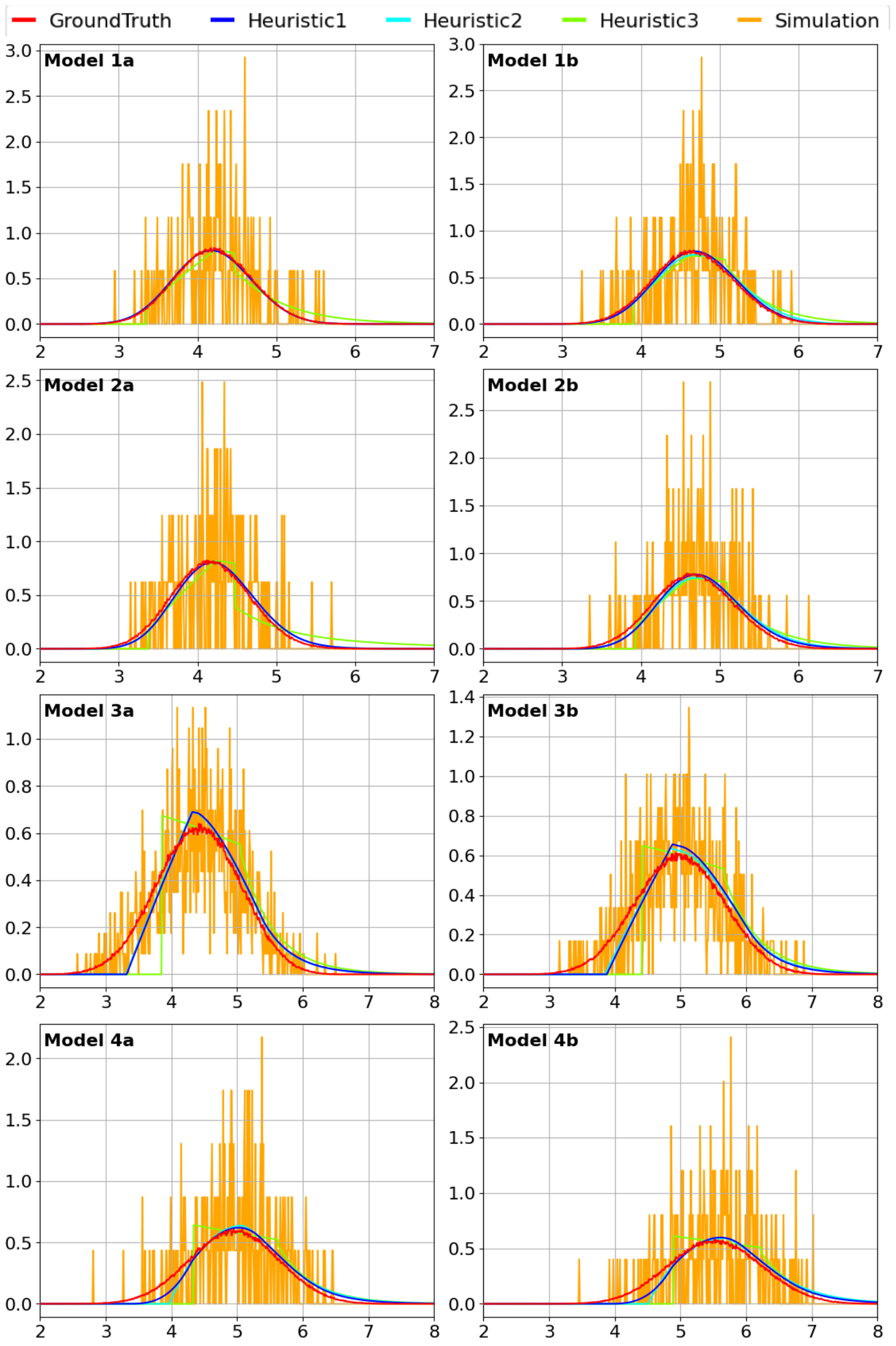


Fig. 4: **Response time PDFs of the workflow models of Fig. 3.**

References

1. Van der Aalst, W.M.: The application of Petri nets to workflow management. Journal of circuits, systems, and computers **8**(01), 21–66 (1998)
2. Arnold, F., Hermanns, H., Pulungan, R., Stoelinga, M.: Time-dependent analysis of attacks. In: Proc. Int. Conf. on Principles of Security and Trust. pp. 285–305. Springer (2014)
3. Baccelli, F., Makowski, A.M.: Multidimensional stochastic ordering and associated random variables. Operations Research **37**(3), 478–487 (1989)
4. Berthomieu, B., Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets **17**(3), 259–273 (1991)
5. Biagi, M., Carnevali, L., Paolieri, M., Papini, T., Vicario, E.: Exploiting non-deterministic analysis in the integration of transient solution techniques for Markov regenerative processes. In: Proc. Int. Conf. on Quantitative Evaluation of Systems. pp. 20–35. Springer (2017)
6. Bobbio, A., Telek, M.: Markov regenerative spn with non-overlapping activity cycles. In: Proc. Int. Comput. Perf. and Depend. Symp. pp. 124–133 (1995)
7. Bruneo, D., Distefano, S., Longo, F., Scarpa, M.: QoS assessment of WS-BPEL processes through non-Markovian stochastic Petri nets. In: Proceedings of IPDPS. pp. 1–12. IEEE (2010)
8. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: Qos-aware replanning of composite web services. In: Proc. IEEE Int. Conf. on Web Services. pp. 121–129. IEEE (2005)
9. Carnevali, L., Grassi, L., Vicario, E.: State-density functions over DBM domains in the analysis of non-Markovian models. IEEE Trans. on Soft. Eng. **35**(2), 178–194 (2009)
10. Carnevali, L., Reali, R., Vicario, E.: Compositional evaluation of stochastic workflows for response time analysis of composite web services. In: Proc. of the ACM/SPEC Int. Conf. on Performance Engineering. pp. 177–188 (2021)
11. Ciardo, G., German, R., Lindemann, C.: A characterization of the stochastic process underlying a stochastic Petri net. IEEE Trans. on Soft. Eng. **20**(7), 506–515 (1994)
12. Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., Weerawarana, S.: Business process execution language for web services (2002)
13. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: AVMFSS'89. LNCS, vol. 407, pp. 197–212. Springer (1990)
14. Gias, A.U., van Hoorn, A., Zhu, L., Casale, G., Düllmann, T.F., Wurster, M.: Performance engineering for microservices and serverless applications: The radon approach. In: Companion of the ACM/SPEC International Conference on Performance Engineering. pp. 46–49 (2020)
15. Horváth, A., Paolieri, M., Ridi, L., Vicario, E.: Transient analysis of non-Markovian models using stochastic state classes. Perf. Eval. **69**(7-8), 315–335 (2012)
16. Jensen, E.D., Locke, C.D., Tokuda, H.: A time-driven scheduling model for real-time operating systems. In: Rtss. vol. 85, pp. 112–122 (1985)
17. Johnson, R., Pearson, D., Pingali, K.: The program structure tree: Computing control regions in linear time. In: ACM Conf. on Programming Language Design and Implementation (PLDI). pp. 171–185. ACM (1994)
18. de Kok, T.G., Fransoo, J.C.: Planning supply chain operations: definition and comparison of planning concepts. Handbooks in operations research and management science **11**, 597–675 (2003)

19. Korenčiak, L., Krčál, J., Řehák, V.: Dealing with zero density using piecewise phase-type approximation. In: European Workshop on Performance Engineering. pp. 119–134. Springer (2014)
20. Kulkarni, V.: Modeling and analysis of stochastic systems. Chapman & Hall (1995)
21. Lin, J.: Divergence measures based on the shannon entropy. IEEE Trans. on Information theory **37**(1), 145–151 (1991)
22. Liu, Y., Zheng, Z., Zhang, J.: Markov model of web services for their performance based on phase-type expansion. In: Proc. DASC-PICOM-CBDCOM-CYBERSCITECH. pp. 699–704. IEEE (2019)
23. Nielsen, F.: On a generalization of the jensen-shannon divergence and the js-symmetrization of distances relying on abstract means. arXiv preprint arXiv:1904.04017 (2019)
24. Paolieri, M., Biagi, M., Carnevali, L., Vicario, E.: The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems. IEEE Trans. on Soft. Eng. **47**, 1211–1225 (2021)
25. Paolieri, M., Horváth, A., Vicario, E.: Probabilistic Model Checking of Regenerative Concurrent Systems. IEEE Trans. Softw. Eng. **42**(2), 153–169 (Feb 2016)
26. Pesu, T., Kettunen, J., Knottenbelt, W.J., Wolter, K.: Three-way Optimisation of Response Time, Subtask Dispersion and Energy Consumption in Split-Merge Systems. In: Proceedings of VALUETOOLS 2017. pp. 244–251. ACM (2017)
27. Rahman, J., Lama, P.: Predicting the end-to-end tail latency of containerized microservices in the cloud. In: 2019 IEEE International Conference on Cloud Engineering (IC2E). pp. 200–210. IEEE (2019)
28. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Information Systems **54**, 1–14 (2015)
29. Russell, N., Ter Hofstede, A.H., Van Der Aalst, W.M., Mulyar, N.: Workflow control-flow patterns: A revised view. BPM Center Report BPM-06-22, BPMcenter. org pp. 06–22 (2006)
30. Sassoli, L., Vicario, E.: Close form derivation of state-density functions over DBM domains in the analysis of non-Markovian models. In: Proc. Int. Conf. on Quantitative Evaluation of Systems. pp. 59–68. IEEE (2007)
31. SIRIO Library: https://github.com/oris-tool/sirio (2020)
32. Trivedi, K.S., Sahner, R.: Sharpe at the age of twenty two. ACM SIGMETRICS Performance Evaluation Review **36**(4), 52–57 (2009)
33. Van Eyk, E., Iosup, A., Abad, C.L., Grohmann, J., Eismann, S.: A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. pp. 21–24 (2018)
34. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. **68**(9), 793–818 (2009)
35. Vicario, E.: Static analysis and dynamic steering of time-dependent systems. IEEE Trans. Softw. Eng. **27**(8), 728–748 (Aug 2001)
36. Zhang, Y., Zheng, Z., Lyu, M.R.: WSPred: A time-aware personalized QoS prediction framework for Web services. In: IEEE Int. Symp. on Software Reliability Engineering. pp. 210–219. IEEE (2011)
37. Zheng, Z., Trivedi, K.S., Qiu, K., Xia, R.: Semi-Markov models of composite web services for their performance, reliability and bottlenecks. IEEE Trans. on Services Computing **10**(3), 448–460 (2015)

6 Appendix

We recall syntax and semantics of STPNs (Section 6.1) and numerical analysis of well-structured workflows (Section 6.2), and we report theorem proofs (Section 6.3) and a graphical representation of analysis heuristics (Section 6.4).

6.1 Formal syntax and semantics of STPNs

Syntax An STPN is a tuple $\langle P, T, A^-, A^+, EFT, LFT, F, W, Z \rangle$ where: P and T are disjoint sets of places and transitions, respectively; $A^- \subseteq P \times T$ and $A^+ \subseteq T \times P$ are pre-condition and post-condition relations, respectively; EFT and LFT associate each transition $t \in T$ with an earliest firing time $EFT(t) \in \mathbb{Q}_{\geqslant 0}$ and a latest firing time $LFT(t) \in \mathbb{Q}_{\geqslant 0} \cup \{\infty\}$ such that $EFT(t) \leqslant LFT(t)$; F, W, and Z associate each transition $t \in T$ with a Cumulative Distribution Function (CDF) F_t for its duration $\tau(t) \in [EFT(t), LFT(t)]$ (i.e., $F_t(x) = P\{\tau(t) \leqslant x\}$, with $F_t(x) = 0$ for $x < EFT(t)$ and $F_t(x) = 1$ for $x > LFT(t)$), a weight $W(t) \in \mathbb{R}_{>0}$, and a priority $Z(t) \in \mathbb{N}$, respectively.

As usual in stochastic Petri nets, a transition t is called *immediate* (IMM) if $EFT(t) = LFT(t) = 0$ and *timed* otherwise; a timed transition t is called *exponential* (EXP) if $F_t(x) = 1 - \exp(-\lambda x)$ for some rate $\lambda \in \mathbb{R}_{>0}$, or *general* (GEN) otherwise. For each GEN transition t, we assume that F_t can be expressed as the integral function of a probability density function (PDF) f_t, i.e., $F_t(x) = \int_0^x f_t(y)\,dy$. Similarly, an IMM transition $t \in T$ is associated with a generalized PDF represented by the Dirac impulse function $f_t(y) = \delta(y - \overline{y})$ with $\overline{y} = EFT(t) = LFT(t)$. A place $p \in P$ is called an *input* or *output* place for a transition $t \in T$ if $(p, t) \in A^-$ or $(t, p) \in A^+$, respectively.

Semantics The state of an STPN is a pair $s = \langle m, \tau \rangle$, where $m : P \to \mathbb{N}$ is a *marking* assigning a number of tokens to each place and $\tau : T \to \mathbb{R}_{\geqslant 0}$ associates each transition with a *time-to-fire*. A transition is *enabled* by a marking if each of its input places contains at least one token. The next transition t to fire in a state s is selected from the set E of enabled transitions having time-to-fire equal to zero and maximum priority with probability $W(t)/\sum_{t_i \in E} W(t_i)$. When t fires, s is replaced with $s' = \langle m', \tau' \rangle$, where: m' is derived from m by removing a token from each input place of t, yielding an intermediate marking m_{tmp}, and adding a token to each output place of t; τ' is derived from τ by: *i*) reducing the time-to-fire of each *persistent* transition (i.e., enabled by m, m_{tmp} and m') by the time elapsed in s; *ii*) sampling the time-to-fire of each *newly-enabled* transition t_n (i.e., enabled by m' but not by m_{tmp}) according to F_{t_n}; and, *iii*) removing the time-to-fire of each *disabled* transition (i.e., enabled by m but not by m').

6.2 Numerical analysis of well-structured workflows

We derive the numerical form of the response time CDF of a block by combining bottom-up the numerical forms of the response time CDFs of the blocks that it contains, provided that the block has a well-formed structure, i.e., it is not a DAG block and it does not contain DAG blocks, and that it does not contain REPEAT blocks. Specifically, given n blocks $b_1, \ldots, b_n$ with response time CDF $\Phi_1(t), \ldots, \Phi_n(t)$ and PDF $\phi_1(t), \ldots, \phi_n(t)$, respectively:

- the response time CDF $\Phi_{\text{seq}}(t)$ of a SEQ block made of $b_1, \ldots, b_n$ is derived by performing subsequent convolutions of $\phi_1(t), \ldots, \phi_n(t)$ $\forall\, t \in [0, t_{\max}]$:

$$\Phi_{\text{seq}}(t) = \Phi^{1,n}(t)$$

$$\Phi^{1,i}(t) = \int_0^t \int_0^\tau \phi^{1,i-1}(x)\,\phi_i(\tau - x)\,dx\,d\tau \;\; \forall\, i \in \{2, \ldots, n\} \tag{2}$$

$$\phi^{1,i-1}(t) = \frac{d}{dt}\Phi^{1,i-1}(t) \;\; \forall\, i \in \{2, \ldots, n-1\}, \;\; \phi^{1,1}(t) = \phi_1(t)$$

- the response time CDF $\Phi_{\text{and}}(t)$ of an AND block made of $b_1, \ldots, b_n$ is the CDF of the maximum among the response times of $b_1, \ldots, b_n$, which is derived as the product of $\Phi_1(t), \ldots, \Phi_n(t)$ $\forall\, t \in [0, t_{\max}]$ due to the fact that the response times of $b_1, \ldots, b_n$ are independent random variables:

$$\Phi_{\text{and}}(t) = \Phi_1(t) \cdot \ldots \cdot \Phi_n(t) \tag{3}$$

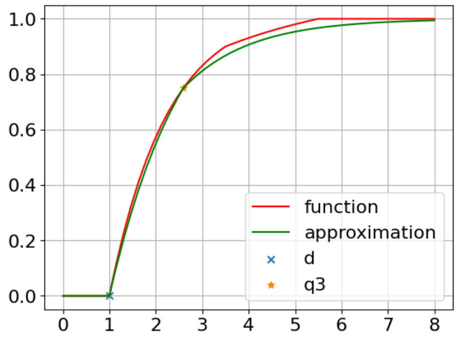
- the response time CDF $\Phi_{\text{xor}}(t)$ of an XOR block made of $b_1, \ldots, b_n$ is derived as the weighted sum of $\Phi_1(t), \ldots, \Phi_n(t)$ $\forall\, t \in [0, t_{\max}]$:

$$\Phi_{\text{xor}}(t) = p_1\,\Phi_1(t) + \ldots + p_n\,\Phi_n(t) \tag{4}$$

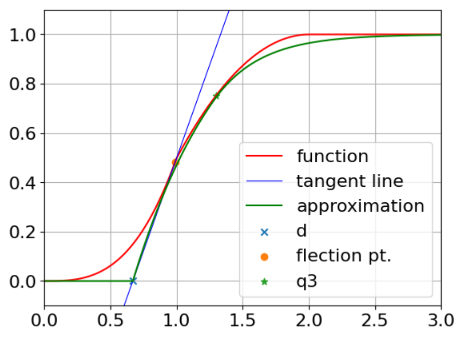
6.3 Theorem proofs

Proof of Lemma 2. By construction, $\hat{F}(x) \leqslant F(x)$ $\forall\, x \in D \cap [d, \infty)$, and $\hat{F}(x) = 0$ $\forall\, x < d$. Starting from the point with abscissa a, the PDF of $F(x)$ may be either increasing or decreasing: If the PDF is increasing, then it will reach a maximum point that comprises an inflection point for the CDF $F(x)$, where the concavity changes from upward to downward (see Fig. 5b). By construction, the tangent line to the inflection point intersects the x-axis in a point whose abscissa d is larger than a, otherwise the CDF should have downward concavity before the inflection point, which contradicts the hypothesis. Otherwise (i.e., if the PDF is decreasing), $F(x)$ has downward concavity (see Fig. 5a), d is selected equal to a, and stochastic order is verified for $\forall\, x \in D \cap [a, \infty)$. Therefore, $\hat{X} \geqslant_{st} X$. □

Proof of Lemma 3. The duration of the sub-workflow associated with any node m (SEQ, AND, XOR, REPEAT, DAG) is a monotone nondecreasing function of the durations of the sub-workflows associated with its children; respectively, the sum (SEQ), max (AND), random mixture (XOR), series (REPEAT), max over all paths from the initial to the final node (DAG). By definition of stochastic order, if a child n is replaced with n' s.t. $T(n) \leqslant_{st} T(n')$, then $T(m) \leqslant_{st} T(m')$ for the new node m'. By recursion, $T(n_0) \leqslant_{st} T(n_0')$ for the new root n_0'. □



(a) **Downward concavity.**



(b) **Upward concavity.**

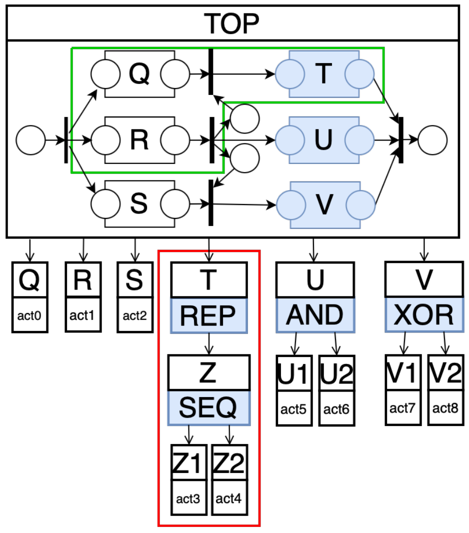
Fig. 5: **Stochastic upper bound CDF: concavity of approximated CDF.**

Proof of Lemma 4. Since DAG edges denote AND-join dependencies, the response time of a vertex v is $T(v) = D(v) + \max(T(k_1), \ldots, T(k_n))$ where $D(v)$ is the duration of the block associated with v and $T(k_1), \ldots, T(k_n)$ are the response times of its predecessors. By visiting the vertices of G in topological order, we can evaluate the response time $T(v_F)$ of the DAG as an expression combining nonnegative block durations $D(v)\ \forall v \in V$ through monotone nondecreasing operators (i.e., summation and maximum). The intermediate values of this expression obtained during the visit are the response times $T(\cdot)$ of the nodes of G, which, by construction, are positively correlated. In the evaluation of $T(v'_F)$ in G', the random variable $T(k)$ of each node $k \in K$ is replaced with the independent replica $T(k') \sim T(k)$. Then, by Lemma 1, we obtain $T(v'_F) \geqslant_{st} T(v_F)$. □

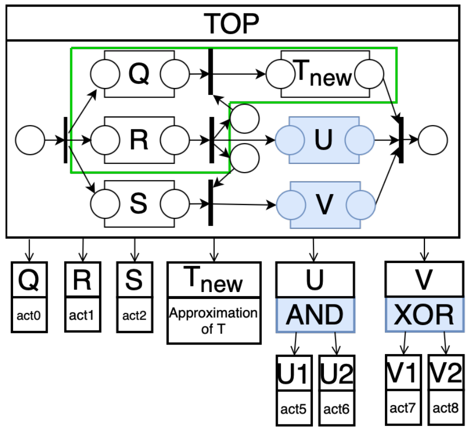
6.4 Analysis actions

Fig. 6 illustrates the application of the sequence of actions 3 and 4 on the structure tree presented in Fig. 1b. In particular, in Fig. 6a, a red and a green box identify two sub-structures on which actions 3 and 4 are applied, respectively:

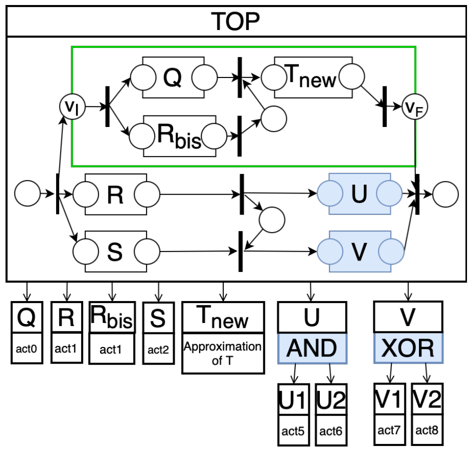
- Action 3: is applied as follows: some action (depending on the considered analysis heuristic) is used to evaluate the response time of the sub-structure in the red box, which is then replaced with an activity block T_{new} associated with the evaluated response time CDF (see Fig. 6b).
- Action 4 evaluates the sub-structure in the green box independently of the rest of the DAG: the blocks that are shared with the rest of the DAG (i.e., block R) are replicated (i.e., block R_{bis} is added), also adding two fictitious zero-duration nodes v_I and v_F (see Fig. 6c); then, this sub-structure is evaluated though some action (depending on the considered analysis heuristic); finally, the sub-structure is replaced with an activity block QRT_{new} associated with the evaluated response time CDF (see Fig. 6d).



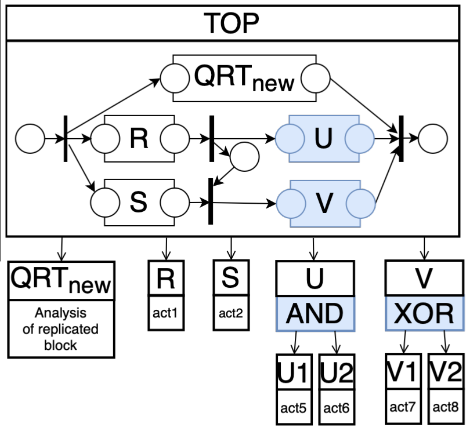
(a)



(b)



(c)



(d)

Fig. 6: **A graphical representation for actions 3 and 4.** (a) The structure tree presented in Fig. 1b, where the red and the green boxes indicate the sub-structures subject to actions 3 and 4, respectively. (b) The structure tree after the application of action 3, which replaces the sub-structure in the red box with an activity block. (c) The structure tree after the replication of block R during action 4. (d) The structure tree after the execution of action 4, which replaces the sub-structure in the green box with an activity block.