Compositional Safe Approximation of Response Time Probability Density Function of Complex Workflows

LAURA CARNEVALI, Department of Information Engineering, University of Florence, Italy
MARCO PAOLIERI, Department of Computer Science, University of Southern California, USA
RICCARDO REALI and ENRICO VICARIO, Department of Information Engineering, University of Florence, Italy

We evaluate a stochastic upper bound on the response time Probability Density Function (PDF) of complex workflows through an efficient and accurate compositional approach. Workflows consist of activities having generally distributed stochastic durations with bounded supports, composed through sequence, choice/merge, and balanced/unbalanced split/join operators, possibly breaking the structure of well-formed nesting. Workflows are specified using a formalism defined in terms of Stochastic Time Petri Nets that permits decomposition into a hierarchy of subworkflows with positively correlated response times, guaranteeing that a stochastically larger end-to-end response time PDF is obtained when intermediate results are approximated by stochastically larger PDFs and when dependencies are simplified by replicating activities appearing in multiple subworkflows. In particular, an accurate stochastically larger PDF is obtained by combining shifted truncated Exponential terms with positive or negative rates. Experiments are performed on sets of manually and randomly generated models with increasing complexity, illustrating under which conditions different decomposition heuristics work well in terms of accuracy and complexity and showing that the proposed approach outperforms simulation having the same execution time.

CCS Concepts: • **Theory of computation → Stochastic approximation**; • **Mathematics of computing → Stochastic processes;**

Additional Key Words and Phrases: Stochastic workflows, response time Probability Density Function, randomly generated structured models, Stochastic Time Petri Nets, non-Markovian processes, compositional evaluation

ACM Reference format:
Laura Carnevali, Marco Paolieri, Riccardo Reali, and Enrico Vicario. 2023. Compositional Safe Approximation of Response Time Probability Density Function of Complex Workflows. *ACM Trans. Model. Comput. Simul.* 33, 4, Article 16 (October 2023), 26 pages.
https://doi.org/10.1145/3591205

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").
Authors' addresses: L. Carnevali, R. Reali, and E. Vicario, Department of Information Engineering, University of Florence, via di Santa Marta 3, Florence, Italy, 50139; emails: {laura.carnevali, riccardo.reali, enrico.vicario}@unifi.it; M. Paolieri, Department of Computer Science, University of Southern California, 941 Bloom Walk, Los Angeles, CA, USA, 90089; email: paolieri@usc.edu.

1049-3301/2023/10-ART16 $15.00
https://doi.org/10.1145/3591205

1 INTRODUCTION

Workflow models describe the production of an output through concurrent activities orchestrated by precedence constraints and control-flow constructs. Over the years, a set of elementary patterns has emerged to define workflows: the *sequence* pattern for consecutive activities, the *split*/*join* pattern for independent prerequisites, and the *choice*/*merge* for alternative activities. Complex workflows can also break the well-formed nesting properties of these elementary patterns by including **directed acyclic graphs (DAGs)** of dependencies between activities and *loops* that repeat some activity [29]. This abstraction has been successfully applied to a large variety of material and digital processes from multiple contexts, including supply chain management [15], administration [5, 33], composite web services [14], and cloud "functions as a service" [34].

When a workflow includes a stochastic model for the duration of activities and for the outcome of control-flow patterns, its quantitative evaluation can provide valuable metrics to achieve tradeoffs between performance goals (e.g., average response time, subtask dispersion, energy consumption) during different stages of system design and operation [6, 9, 17, 27]. This approach is particularly useful for the analysis of **Service Level Agreements (SLAs)** with *soft deadlines* and *penalty functions* [19, 26], which can be defined as *rewards* calculated from the **Probability Density Function (PDF)** of the *end-to-end response time* of the workflow. For example, the probability of missing a soft deadline $T_{\max}$ can be obtained by integrating the response time PDF over $[T_{\max}, +\infty)$; the expected penalty can be evaluated by integrating the product of the response time PDF and of a penalty function.

Notably, these metrics require not only summary statistics such as mean and variance but also the *entire response time PDF*: While simulation and other approximation methods can speed up its evaluation for complex workflows, many applications require additional *safety guarantees*. For example, soft deadlines require a stochastically ordered approximation (i.e., a PDF with integral over $[T_{\max}, +\infty)$ that is an upper bound of the exact result) to avoid underestimating the probability of missing a deadline. Even more critically, the evaluation of expected penalties depends on the accuracy of the PDF *at every point* instead of its cumulative integral.

However, exact analytical and numerical methods cannot be applied to large workflows because of two main sources of computational complexity: (1) activity durations have **general (i.e., non-Exponential) probability distributions (GEN)**, often within firm bounds enforced by design or by contract, which result in a non-Markovian stochastic process [13] of workflow execution, and (2) the interleaving of activities in concurrent subworkflows leads to a state-space explosion and to complex stochastic dependencies due to their overlapping GEN durations [4].

To address these issues, compositional approaches combine solutions of subworkflows in a bottom-up fashion, with different limitations imposed on the distribution types of activity durations and with different guarantees on the final error. When a workflow is acyclic and its activities have **Continuous Phase (CPH)** durations, the response time distribution can be evaluated through the bottom-up composition of CPH distributions proposed in Reference [1]; complexity can be reduced by limiting the number of phases of CPH distributions, without safety guarantees on the final error. In Reference [23], repetitions of CPH activities are allowed, but the approach suffers from state-space explosion. When activities with GEN duration are composed by fork/join, sequence, and repetition, Reference [39] derives mean and standard deviation of the response time (instead of its distribution) by an efficient bottom-up calculus. The response time distribution of acyclic, well-nested workflows with GEN durations is evaluated bottom-up in Reference [12] by combining results of Markov regenerative analysis of subworkflows with limited concurrency. The solution is extended in Reference [10] to repetition and unbalanced split/join constructs, while *guaranteeing stochastic order* through the method used to simplify dependencies and through the class of PDFs used to approximate the response time of subworkflows.

In this article, we propose an efficient and accurate compositional technique to evaluate a stochastic upper bound on the response time PDF of complex workflows. Specifically, workflows consist of activities with GEN duration and bounded support, composed through sequence, choice/merge, and balanced/unbalanced split/join constructs and specified by a structured formalism, defined in terms of **Stochastic Time Petri Nets (STPNs)** [37], that permits the workflow decomposition into a hierarchy of subworkflows with positively correlated response times (Section 2). This work significantly extends the solution methods of References [10, 12] in the following aspects: We provide a thorough formalization for the approximate derivation of conservative measures that estimate the complexity of evaluation of a subworkflow, driving the workflow decomposition into subworkflows analyzed in isolation (Section 3); we define a novel stochastically ordered approximant for the response time PDF of subworkflows by combining shifted truncated **Exponential (EXP)** terms, each having positive or negative rate depending on the concavity of the response time **Cumulative Distribution Function (CDF)** (Section 4); notably, we provide an extensive experimentation on manually and randomly generated models with increasing complexity, illustrating which decomposition heuristics work better under which conditions, outperforming simulations with the same computation times (Section 5). Finally, we draw our conclusions (Section 6). We recall STPNs in Appendix A and bottom-up numerical analysis of well-nested workflows in Appendix B, and we report theorem proofs in Appendix C.

2 WORKFLOW MODEL

We specify workflows through a class of STPNs (Section 2.1 and Appendix A) that is sufficient to represent a variety of control patterns [29, 40] while making their structure of composition explicit and guaranteeing positive correlation among the response times of subworkflows (Section 2.2).

2.1 STPN Blocks

Workflows are defined by recursive composition of *blocks*, each specified by 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 STPN transitions through nested constructs modeling sequential behavior (sequence, choice/merge) and concurrent behavior (split/join), and through acyclic constructs breaking well-formed nesting by unbalanced fork and join operations (simple split, simple join). In particular, we consider the following types of blocks:

- An elementary *activity* is represented by an STPN with a single transition with GEN duration connecting the initial and final places (e.g., S in Figure 1(a)). GEN transitions have expolynomial (also termed exponomial [32]) PDFs defined as the sum of products of exponential and mononomial terms, i.e., $f(x) = \sum_{m=1}^{M} c_m \prod_{n=0}^{N_m} x_n^{\alpha_{mn}} e^{-\lambda_{mn} x_n}$, with analytical representation over the entire domain or piecewise-defined over multiple subdomains.
- SEQ{$\textsc{Block}_1, \ldots, \textsc{Block}_n$} is a *sequence* of n blocks $\textsc{Block}_1, \ldots, \textsc{Block}_n$ (e.g., Y in Figure 1(a)).
- XOR{$\textsc{Block}_1, \ldots, \textsc{Block}_n, p_1, \ldots, p_n$} is an immediate random *exclusive choice* made of n initial **immediate (IMM)** transitions (i.e., with zero time-to-fire) connected to n alternative blocks $\textsc{Block}_1, \ldots, \textsc{Block}_n$ having probabilities $p_1, \ldots, p_n$, respectively, which, in turn, are connected to a final IMM *simple merge* transition (e.g., R_2 in Figure 1(a)). An XOR block is *balanced*, i.e., all the alternative paths started at the initial split are terminated at the final join.
- AND{$\textsc{Block}_1, \ldots, \textsc{Block}_n$} is a balanced split-join made of an initial IMM *parallel split* transition that forks execution along n concurrent blocks $\textsc{Block}_1, \ldots, \textsc{Block}_n$ and a final IMM *synchronization* transition that terminates the block (e.g., X in Figure 1(a)).

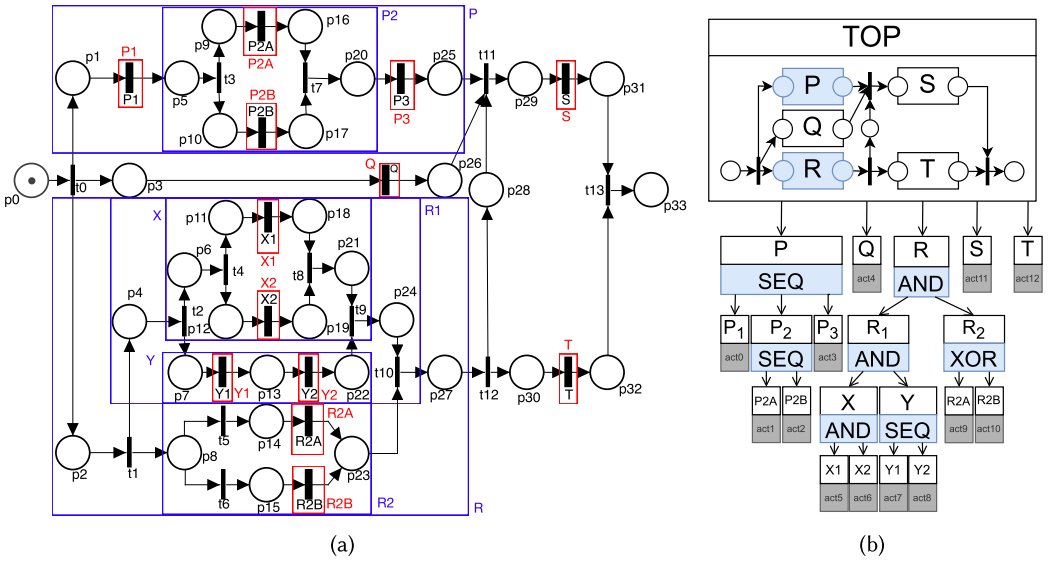

Fig. 1. (a) STPN model of a workflow: Blocks are highlighted by boxes, blue for composite blocks, and red for activity blocks; all transitions have uniform PDF over $[0, 1]$ (firing intervals and PDF types are not shown to reduce the cluttering). (b) Structure tree of the workflow of Figure 1(a): Composite blocks are filled with blue.

- DAG$\{\textsc{Block}_1, \ldots, \textsc{Block}_n\}$ is the composition of n blocks $\textsc{Block}_1, \ldots, \textsc{Block}_n$ in a DAG with a single initial place and a single final place, by means of IMM *simple split* transitions (i.e., with a single input place and multiple output places) and IMM *simple join* transitions (i.e., with multiple input places and a single output place) [29]. Since *simple split* and *simple join* operators are not necessarily balanced, a DAG block can break well-formed nesting of concurrent blocks. A DAG is termed *minimal* if it cannot be reduced by SEQ, XOR, and AND (e.g., in Figure 1(a), the top level is a minimal DAG with initial place `p0`, final place `p33`, simple split transitions `t0` and `t12`, and simple join transitions `t11` and `t13`).

We assign different priorities to the transitions of different blocks to exclude races between IMM transitions; in addition to reducing the number of possible firing sequences, this approach avoids the interaction of transition weights when XOR blocks are concurrently enabled (a formal semantics of IMM selection through weights and priorities is provided in Appendix A).

According to this definition, each workflow model specified as a composition of STPN blocks can be translated into a unique STPN. Conversely, the composition of blocks does not cover all the expressivity of STPNs. In particular, given that choices are expressed only by IMM transitions in balanced XOR blocks, workflow models cannot represent race selections where a choice is determined by the execution times of concurrent activities, e.g., early preemption of a timed activity by a timeout. 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 guarantee positive correlation among the completion times of different intermediate steps.

2.2 Structure Tree

Blocks combined as described in Section 2.1 enable the decomposition of a workflow model as a *structure tree* $S = \langle N, E, n_0 \rangle$, where N is the set of nodes (i.e., blocks), E is the set of directed edges connecting each block with its component blocks, and n_0 is the root node (i.e., the overall

workflow). Figure 1(b) shows the structure tree of the workflow STPN of Figure 1(a). Specifically, a block is depicted as a box labeled with the block name and with either the activity name (for elementary activity blocks) or the block type (for SEQ, XOR, AND, and DAG blocks). Moreover, the box of a DAG block also contains places and transitions connecting their component blocks.

The representation of workflows as hierarchical graphs with single-entry single-exit blocks are inspired by *program structure trees* [20] and *process structure trees* [35]. Similarly to these works, we ensure that the structure tree of a workflow is unique and robust to local changes (i.e., modifying a subworkflow in the STPN affects only the corresponding 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., if possible, SEQ and AND blocks are used instead of DAG block).

3 COMPLEXITY OF WORKFLOW ANALYSIS

We derive the response time PDF of a block either by bottom-up numerical analysis (Appendix B) if the block composes independent subworkflows in well-nested structures through SEQ, AND, and XOR blocks or by *forward transient analysis* [18] if the block is a DAG including dependencies among subworkflows composed in non-well-nested structures. We characterize the factors that affect the complexity of forward transient analysis (Section 3.1), and we exploit the state class graph of the underlying TPN to define measures to estimate such complexity (Section 3.2).

3.1 Complexity Factors

Evaluation of DAG blocks can be performed by forward transient analysis based on the method of stochastic state classes [18]. Specifically, after each firing, the analysis computes a stochastic state class $\Sigma = \langle m, D, f \rangle$ encoding a marking m (i.e., an assignment of tokens to places), a **Difference Bounds Matrix (DBM)** zone D [16] representing the joint support of the times-to-fire of the enabled transitions and the elapsed time, and a joint PDF f for such values. The analysis enumerates the tree of stochastic state classes reached within a time limit $t_{\max}$ and computes the block response time PDF as the derivative of the first-passage probability of the marking assigning one token to the final place of the STPN. The SIRIO library [31] of the ORIS tool [25] provides a closed-form implementation of the analysis provided that each transition has an expolynomial PDF. Note that *regenerative transient analysis* [18] based on the method of stochastic state classes, also available in the SIRIO library, is not used due to the very limited number of regeneration points (i.e., time instants at which the Markov condition is satisfied) reached by DAG blocks.

The complexity of forward transient analysis of an STPN can be estimated from the maximum number of concurrently enabled GEN transitions and the maximum number of firings of GEN transitions from the start to the end of the model execution, which can be efficiently derived by *nondeterministic analysis* of the underlying TPN. Specifically, nondeterministic analysis is sufficient to identify the set of feasible behaviors while avoiding the complexity of evaluation of their measure of probability, encoding the continuous set of executions of the STPN into a discrete representation termed *state class graph* [3, 36], where each vertex is a *state class* $S = \langle m, D \rangle$ made of a marking m and a DBM zone D for the times-to-fire of the enabled transitions, and each directed edge (S, t, S') is a succession relation from S to the state class $S' = \langle m', D' \rangle$ with marking m' and zone D' after the firing of transition t. The state class 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 [36]. Notably, the graph makes explicit the degree of concurrency among GEN timers (i.e., the number of GEN transitions enabled in each state class) and facilitates the derivation of the length of specific behaviors (i.e., the number of firings between selected state classes). Given that the STPN of a workflow has a final absorbing place $\mathtt{p_{fin}}$, all the paths in the

state class graph of the underlying TPN terminate in a state class with marking $\mathtt{p_{fin}}$ and no enabled transition.

For instance, forward transient analysis of the STPN of the workflow of Figure 1(a) is not affordable, which can be inferred from the large number of concurrently enabled GEN timers in the state class graph and the large number of firings of GEN transitions from the initial to the final state class. For more complex workflows, also non-deterministic analysis of the underlying TPN may become computationally demanding, pointing out the need of an efficient compositional solution not only to evaluate the workflow response time PDF but also to estimate the complexity of its analysis.

Note that the complexity of forward transient analysis of an STPN depends on the number of concurrently enabled GEN transitions and the number of firings of GEN transitions from the start to the end of the model execution due to the following reasons. First, complexity depends on the number and length of the firing sequences before the time limit $t_{\max}$ and thus on the number of stochastic state classes reached by $t_{\max}$, which, in turn, 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 expolynomial terms of the (monovariate) PDFs of the GEN transitions [30]. Our approach limits the number of stochastic state classes by decomposing a workflow into subworkflows analyzed in isolation. Moreover, the number of DBM zones and the number of expolynomial terms of the (multivariate) joint PDFs of stochastic state classes also affect complexity and depend on the number of concurrently enabled GEN transitions and on the number of firings after which a GEN transition is persistent [30]: In fact, at each firing, the number of zones increases polynomially with the number of persistent transitions, and the number of expolynomial terms increases linearly with the polynomial degree of the joint PDF. Additionally, if the analytical form of the joint PDF contains no EXP factor, then the polynomial degree increases linearly with the number of fired or disabled transitions. Our approach limits these complexity factors by workflow decomposition and by approximating the numerical form of the response time PDF of subworkflows analyzed in isolation with a piecewise PDF made of EXP terms. The approximation of the numerical form of the response time PDF of a subworkflow analyzed in isolation is needed to perform forward transient analysis of a higher-level workflow, given that the analysis of the workflow STPN requires each transition to have an expolynomial PDF.

3.2 Complexity Measures

According to the analysis of Section 3.1, we estimate the complexity of forward transient analysis of the STPN of a workflow through the maximum number of concurrently enabled GEN transitions and the maximum number of firings of GEN transitions from the start to the end of the workflow.

Definition 3.1 (Concurrency Degree of a TPN). The *concurrency degree* c of a TPN is the maximum number of concurrent GEN transitions in the state class graph.

Definition 3.2 (Sequencing Degree of a TPN). The *sequencing degree* q of a TPN is the maximum number of firings of GEN transitions from the initial to the final state class.

For complex workflows, the number of state classes may be significant and thus their enumeration may require a non-negligible amount of time. Due to the exponential complexity in the number of state classes, evaluation of q through enumeration of all paths from the initial to the final class would thus become too expensive for our aim to perform workflow decomposition and analysis in a very short time (i.e., a few tens of seconds for significantly complex models). Moreover, c and q would not tell how much of the complexity depends on the structure and the timings of the workflow itself rather than on the structure and timings of its subworkflows, which instead

becomes relevant to decide how to decompose the workflow. To cope with both aspects, we characterize c and q both for the TPN of the workflow and for a variant that hides the complexity of the subworkflows.

Definition 3.3 (Unexpanded TPN). The *unexpanded TPN* of a workflow is the TPN obtained from the workflow TPN by replacing each composite block with an activity block with the same duration.

We consider a workflow with structure tree Ω with depth D, i.e., the top level has depth 1 and the bottom level has depth D. We perform a bottom-up visit of Ω starting from depth $D-1$. At each level, for each composite block b, we derive the *complexity tuple* $\langle C, \bar{C}, q, \bar{q} \rangle$, where C and $\bar{C}$ are upper bounds on the concurrency degree of the TPN and of the unexpanded TPN of b, respectively, and q and $\bar{q}$ are the sequencing degree of the TPN and of the unexpanded TPN of b, respectively. For each level $d \in \{D-1, \ldots, 1\}$ and each composite block b, we perform the following operations.

- We derive a *variant of the unexpanded TPN* of b by replacing each composite block having duration $[l, u]$ with an activity block having duration $[L, U] \supseteq [l, u]$. In particular, the interval $[L, U]$ is computed by the previous iteration of the procedure, at the next lower level (note that, at the first iteration, i.e., at level $D-1$, each block consists of only activity blocks). Then, we derive a lower bound L and an upper bound U on the duration of block b itself, which are used by the next iteration of the procedure to derive a variant of the unexpanded TPNs of the composite blocks at the next higher level. Specifically,

$$L = \begin{cases} \sum_{z \in K_b} L_z & \text{if } b \text{ is SEQ, DAG} \\ \max_{z \in K_b} \{L_z\} & \text{if } b \text{ is AND} \\ \min_{z \in K_b} \{L_z\} & \text{if } b \text{ is XOR} \end{cases} \qquad U = \begin{cases} \sum_{z \in K_b} U_z & \text{if } b \text{ is SEQ, DAG} \\ \max_{z \in K_b} \{U_z\} & \text{if } b \text{ is AND, XOR} \end{cases}, \tag{1}$$

  where K_b is the set of child blocks of b, and, if $z \in K_b$ is a composite block, then L_z and U_z are the lower and upper bound on the duration of z, respectively, computed by the previous step of the procedure, otherwise (i.e., if z is an activity block) L_z and U_z are the minimum and maximum duration of z, respectively. Note that L and U are bounds due to the overapproximation of duration intervals of the composite blocks of b, and also due to the fact that dependencies among subworkflows of DAG blocks are not considered.

- We perform nondeterministic analysis of the variant of the unexpanded TPN of block b:

$$C = \max_{S \in \Gamma} \left\{ \sum_{t \in E_S} C_t \right\} \quad \bar{C} = \max_{S \in \Gamma} \left\{ \sum_{t \in E_S} 1 \right\}, \tag{2}$$

  where Γ is the set of state classes enumerated for the simplified TPN of block b, E_S is the set of GEN transitions enabled in state class S, and C_t is equal to 1 if the block corresponding to transition t is an activity block; otherwise, C_t is equal to the upper bound on the concurrency degree of the composite block corresponding to t, computed at the next lower level. Note that C and $\bar{C}$ are upper bounds due to the fact that behaviors of blocks are considered independently of each other, e.g., it may be the case that two concurrent blocks cannot both reach their maximum number of concurrent GEN timers at the same time.

- We efficiently derive q and $\bar{q}$ as follows (avoiding enumeration of paths from the initial to the final state class, which would have exponential complexity in the number of state classes):

$$q = \begin{cases} \sum_{z \in K_b} q_z & \text{if } b \text{ is SEQ, AND, DAG} \\ \max_{z \in K_b} \{q_z\} & \text{if } b \text{ is XOR} \end{cases} \qquad \bar{q} = \begin{cases} \sum_{z \in K_b} 1 & \text{if } b \text{ is SEQ, AND, DAG} \\ 1 & \text{if } b \text{ is XOR} \end{cases}, \tag{3}$$

  where K_b is the set of child blocks of b, and, if $z \in K_b$ is a composite block, then q_z is the sequencing degree of z computed by the previous step of the procedure at the next lower level, otherwise (i.e., if z is an activity block) q_z is equal to 1.

The concurrency degree and the sequencing degree are exploited to define the *complexity heuristics* based on thresholds Θ_c and Θ_q on the concurrency and sequencing degree, respectively.

Definition 3.4 (Complexity of a Block). A block b with complexity tuple $\langle C, \bar{C}, q, \bar{q} \rangle$ is termed *easy* to analyze if $C \leqslant \Theta_c$ and $q \leqslant \Theta_q$ and *complex* to analyze otherwise. The block is also termed *internally easy* to analyze if $\bar{C} \leqslant \Theta_c$ and $\bar{q} \leqslant \Theta_q$ and *internally complex* to analyze otherwise.

For instance, for the workflow of Figure 1, the evaluation of complexity yields the tuple $\langle C, \bar{C}, q, \bar{q} \rangle = \langle 7, 3, 13, 5 \rangle$, confirming that forward transient analysis of the overall workflow is not affordable due to the large concurrency degree ($C = 7$) and sequencing degree ($q = 13$) among GEN transitions. Moreover, the unexpanded concurrency and sequencing degrees (respectively, $\bar{C} = 3$ and $\bar{q} = 5$) point out that the workflow complexity depends not on the structure of the top-level DAG block but rather on the complexity of block R that it contains, for which $\langle C, \bar{C}, q, \bar{q} \rangle = \langle 4, 2, 5, 2 \rangle$.

4 WORKFLOW EVALUATION

In this section, we illustrate our compositional solution to evaluate the response time PDF of a workflow (Section 4.1), we derive a stochastic upper bound for monovariate PDFs with bounded support (Section 4.2), and we prove that it is indeed a stochastic upper bound (Section 4.3).

4.1 Evaluation Heuristics

We evaluate the end-to-end response time by decomposing a workflow into a hierarchy of subworkflows and by composing the results of their separate analyses, repeatedly applying numerical analysis (Appendix B) and forward transient analysis (Section 3) to leverage their different strengths. On the one hand, numerical analysis combining monovariate PDFs turns out to be efficient in the composition of independent subworkflows through well-nested operators (i.e., SEQ, XOR, AND blocks), but it is not feasible for subworkflows with common dependencies encoded by not well-nested structures (i.e., DAG blocks). However, forward transient analysis manipulating multivariate joint PDFs enables the evaluation of such dependencies among subworkflows, but it suffers from the concurrency degree and the sequencing degree among activities with GEN duration, and its efficient implementation requires that subworkflow durations be represented in analytic form, which may require approximated fitting of numerical results.

We exploit the structure tree to aggregate the subworkflows and to select the solution techniques according to heuristics that trade approximation for complexity reduction while ensuring that the final result is a stochastic upper bound of the exact PDF of the workflow response time. As illustrated by Algorithm 1, first we perform a top-down visit of the structure tree to decompose the workflow into subworkflows, and then we perform a bottom-up visit to compose the results of their separate analyses. Specifically, at each step of the top-down visit, we perform the following operations to derive a stochastic upper bound $\phi(t)$ on the response time PDF of the current block b:

ALGORITHM 1: Evaluation of the response time PDF of a block

```
CompositionalAnalysis(b, Θc, Θq, h)
    input  : block b, concurrency degree threshold Θc, sequencing degree threshold Θq, heuristics h
    output : response time PDF φ(t) of b
1   if b is an activity block then
2   |   return the duration PDF of b
3   if b is a SEQ block or an XOR block or an AND block then
4   |   foreach block b_i of b do
5   |   |   φ_i(t) ← CompositionalAnalysis(b_i, Θc, Θq, h)
6   |   if b is a SEQ block then
7   |   |   return the PDF of b computed as illustrated in Appendix B
8   |   if b is an XOR block then
9   |   |   return the PDF of b computed as illustrated in Appendix B
10  |   if b is an AND block then
11  |   |   return the PDF of b computed as illustrated in Appendix B
12  if b is a DAG block then
13  |   return h(b, Θc, Θq)
```

ALGORITHM 2: Evaluation of the response time PDF of a DAG block by the SDF heuristics

```
SplitDependenciesFirst(b, Θc, Θq)
    input  : workflow block b
    output : response time PDF φ(t) of b, concurrency degree threshold Θc, sequencing degree
             threshold Θq
1   if b is internally complex to analyze then
2   |   return InnerBlockReplication(b, Θc, Θq)
3   if b is complex to analyze then
4   |   return InnerBlockAnalysis(b, Θc, Θq)
5   return the PDF of b computed through forward transient analysis
```

- If b is an activity block, then its exact response time PDF is its duration PDF (lines 1 and 2).
- If b is, or can be reduced to, a well-nested composition of independent subworkflows, then recursive numerical analysis efficiently evaluates the exact response time PDF (lines 3–11).
- If b is a DAG block, then one of two different heuristics can be recursively applied to reduce the block complexity until its forward transient analysis becomes affordable (lines 12 and 13).
  - Algorithm 2 shows the **Split Dependencies First (SDF)** heuristics: If b is internally complex to analyze, then it is split into the AND of two decoupled subworkflows by replicating the common nodes of the two subworkflows (*inner block replication*, lines 1 and 2); if b is complex to analyze, then some block is analyzed in isolation (*inner block analysis*, lines 3 and 4); otherwise (i.e., if b is easy to analyze), forward transient analysis is affordable (line 5).
  - Algorithm 3 shows the **Replace Block First (RBF)** heuristics, a variant of the SDF heuristics where inner block analysis (lines 1 and 2) is applied before inner block replication (lines 3 and 4).

In turn, inner block replication and inner block analysis introduce different approximations.

- As illustrated by Algorithm 4, performing inner block replication on a DAG block b consists in replicating some predecessors of a block v contained within b (line 3) to evaluate the

ALGORITHM 3: Evaluation of the response time PDF of a DAG block by the RBF heuristics

```
ReplaceBlockFirst(b, Θ_c, Θ_q)
    input  : workflow block b, concurrency degree threshold Θ_c, sequencing degree threshold Θ_q
    output : response time PDF φ(t) of b
1   if b contains at least one composite block and is complex to analyze then
2   |   return InnerBlockAnalysis(b Θ_c, Θ_q)
3   if b is internally complex to analyze then
4   |   return InnerBlockReplication(b, Θ_c, Θ_q)
5   return the PDF of b computed through forward transient analysis
```

ALGORITHM 4: Evaluation of the response time PDF of a DAG block by replicating some of its blocks

```
InnerBlockReplication(b, Θ_c, Θ_q)
    input  : workflow block b
    output : response time PDF φ(t) of b
1   order the predecessors of the final block of b by the values of C and q
2   v ← predecessor of the final block of b with max value of C (and, in case of tie, with max value of q)
3   b' ← b after replicating the predecessors of block v
4   return CompositionalAnalysis(b', Θ_c, Θ_q)
```

response time of v independently of the rest of the DAG (replicated blocks are identical) by recursively invoking the compositional analysis algorithm (line 4). The selected block v is the predecessor of the final block of b that has maximum upper bound on the concurrency degree and, in case of tie, maximum sequencing degree (lines 1 and 2).
Specifically, let $G = (V, E, v_I, v_F)$ be the DAG (e.g., the top-level DAG of Figure 1(b)) where V is the set of vertices (i.e., the blocks of b) plus a fictitious initial vertex v_I and a fictitious final vertex v_F (not shown in Figure 1(b)), both with zero-duration, and E is the set of edges (i.e., the precedence relations between blocks). First, the most complex vertex $v \in V \setminus \{v_I, v_F\}$ is identified (i.e., block T in Figure 1(b)). 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 (i.e., $K = \{\mathtt{R}\}$). The vertices in K and the edges to/from vertices in K are replicated (i.e., R' is added to V; $v_I \rightarrow R'$ and $R' \rightarrow T$ are added to E; $R \rightarrow T$ is removed from E). The DAG is then transformed into an AND of two blocks, one consisting of node v and its predecessors, and the other one consisting of the remaining nodes. For instance, as shown in Figure 2(a), after replication of the predecessor R of blocks S and T, the DAG of Figure 1(b) becomes an AND of two blocks, one made of S and its predecessors P, Q, and R, and the other one made of T and the replicated block R′. Then, the model becomes well-nested and thus it can be solved by numerical analysis.

- As illustrated by Algorithm 5, performing inner block analysis on a DAG block b consists in: performing compositional analysis of a composite block v contained within b (line 3), which yields a stochastic upper bound PDF $\phi(t)$ on the response time PDF of v; deriving a stochastic upper bound PDF $\hat{\phi}(t)$ on $\phi(t)$ by Lemma 4.2 (line 4); replacing block v with an activity block with duration $\hat{\phi}(t)$ (line 5); and, performing compositional analysis of block b (line 6). The selected block v is the composite block of b that has maximum upper bound on the concurrency degree and, in case of tie, maximum sequencing degree (lines 1 and 2).
  For instance, as shown in Figure 2(b), compositional evaluation of the example of Figure 1(b) through the RBF heuristics consists in analyzing blocks P and R in isolation and then

ALGORITHM 5: Evaluation of the response time PDF of a DAG block by analyzing one of its blocks

```
InnerBlockAnalysis(b, Θ_c, Θ_q)
    input  : workflow block b
    output : response time PDF φ(t) of b
1   order the blocks of b by the values of C and q
2   v ← block of b with max value of C (and, in case of tie, with max value of q)
3   φ'(t) ← CompositionalAnalysis(v, Θ_c, Θ_q)
4   φ̂(t) ← safe upper bound PDF of φ'(t)
5   b' ← b after replacing v with an activity block with PDF φ̂(t)
6   return CompositionalAnalysis(b', Θ_c, Θ_q)
```

approximating their response time PDF. Then, the model can be solved by forward transient analysis.

The SDF heuristics is more accurate than the RBF heuristics if the correlation between the response times of the replicated nodes and the response time of the overall workflow is low. Conversely, the RBF heuristics is more accurate than the SDF heuristics if the correlation between the response times of the nodes replicated by the SDF heuristics and the response time of the overall workflow is high. For instance, if all activity blocks of the model of Figure 1(b) had uniform response time PDF over $[0, 1]$, then the correlation between the response times of block R and of the overall workflow would be low, and thus thus replicating block R (as done by the SDF heuristics) would yield a more accurate result than analyzing in isolation blocks Q and R (as done by the RBF heuristics), as shown in Figure 2(c). Conversely, if the response times of all the activities of the subworkflow R were uniformly distributed over $[4, 8]$, then the correlation between the response times of R and of the overall workflow would be very high, and thus replicating block R would yield a less accurate result than analyzing in isolation blocks Q and R, as shown in Figure 2(c).

Note that replicating the shared dependencies of a block of a DAG typically reduces the DAG complexity more than analyzing in isolation a block of the DAG. Therefore, if a DAG were internally complex, then, after analyzing in isolation all the non-elementary blocks of the DAG, the RBF heuristics would be forced to replicate the shared dependencies of a node, thus yielding less accurate results than the SDF heuristics if replication alone were sufficient to make the DAG analysis affordable. According to this, the RBF heuristics is expected to outperform the SDF heuristics only in the specific case that: DAGs are not internally complex; DAGs are in top of the structure tree (otherwise they would consist of activity blocks only and inner block analysis could not be applied at all); and, the correlation between the response time of the overall workflow and the response times of the blocks of a DAG that would be replicated by the SDF heuristics is very high.

4.2 Safe Approximation of Duration Distributions

Definition 4.1 (Stochastic Order). Given two random vectors $\mathbf{X}_1$ and $\mathbf{X}_2$, "$\mathbf{X}_1$ 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.

In our compositional analysis, a block may be analyzed in isolation to reduce the workflow complexity and make its analysis affordable. In this case, forward transient analysis of the workflow can be performed provided that the numerical PDF of the block duration be approximated with an analytical PDF (required to be expolynomial for the analysis in ORIS). To obtain safe and accurate

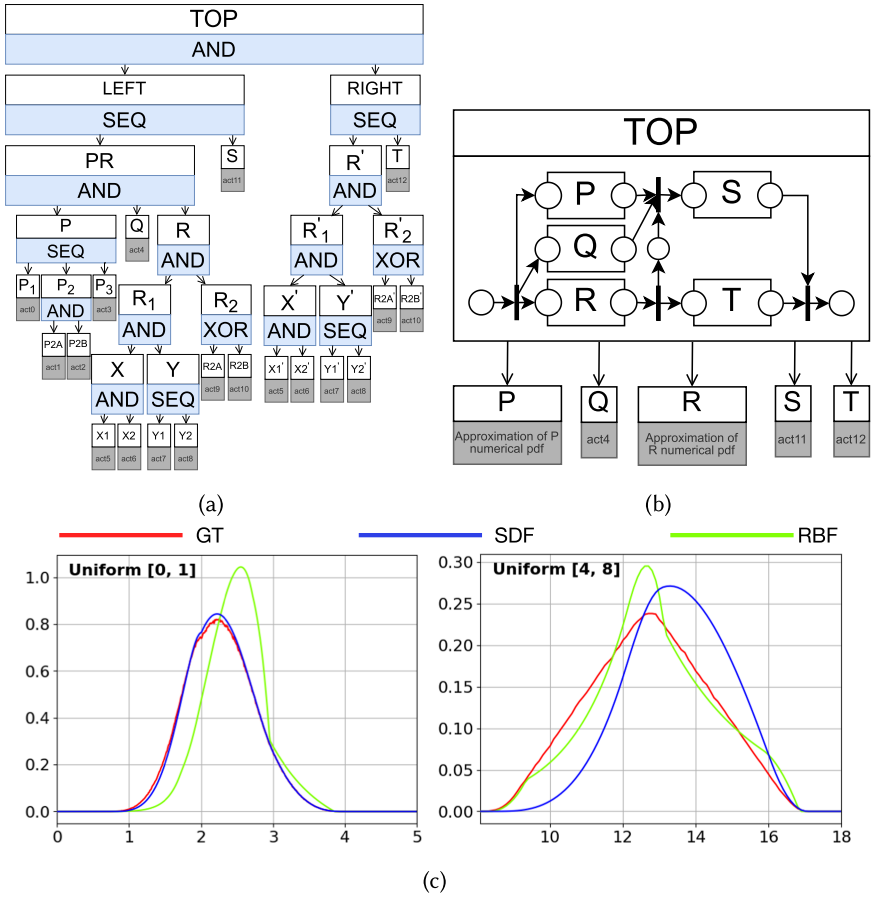

Fig. 2. The workflow of Figure 1(b) after executing (a) the SDF heuristics or (b) the RBF heuristics. (c) Stochastic upper bound on the workflow response time PDF assuming that the response time of each block has uniform PDF over [0, 1] (left) or that the response times of all the activities of block R only have uniform PDF over [4, 8] (right). In both cases, the GT curve is obtained by a 5-million-run simulation.

analytical approximations of the block PDF $f(x)$, we analyze the concavity of the CDF $F(x)$: for each "concave down" or "concave up" piece, our approximation $\hat{F}(x)$ uses the CDF of a shifted and truncated EXP of the form $\lambda\, e^{-\lambda x}$ with positive or negative rate λ, respectively. In particular, for each "concave down" ("concave up") piece, the positive (negative) rate is small (large) enough to guarantee stochastic order (i.e., $\hat{F}(x) \leqslant F(x)\ \forall x$) but as small (large) as possible to provide a close approximation. Figure 3 illustrates four cases for the concavity of $F(x)$: concave down (Figure 3(a)); concave up (Figure 3(b)); concave down, then concave up (Figure 3(c)); concave up, then concave down (Figure 3(d)). In the majority of cases in our experiments (Section 5), we observe CDFs changing concavity at most once, from upward to downward. Lemma 4.2 guarantees stochastic order of $\hat{F}$(x).

Lemma 4.2 (Stochastic Upper Bound PDF). *Let X be a random variable with numerical PDF $f(x_i)$ and CDF $F(x_i)$ with $x_i = x_0 + \delta i$ for $i = 1, \ldots, N$, $x_0 \in \mathbb{R}_{\geqslant 0}$, $\delta \in \mathbb{R}_{>0}$. Let $x_{i_1} < x_{i_2} < \cdots < x_{i_M}$ denote the inflection points of F; then, the random variable $\hat{X}$ with CDF $\hat{F}(x)$*

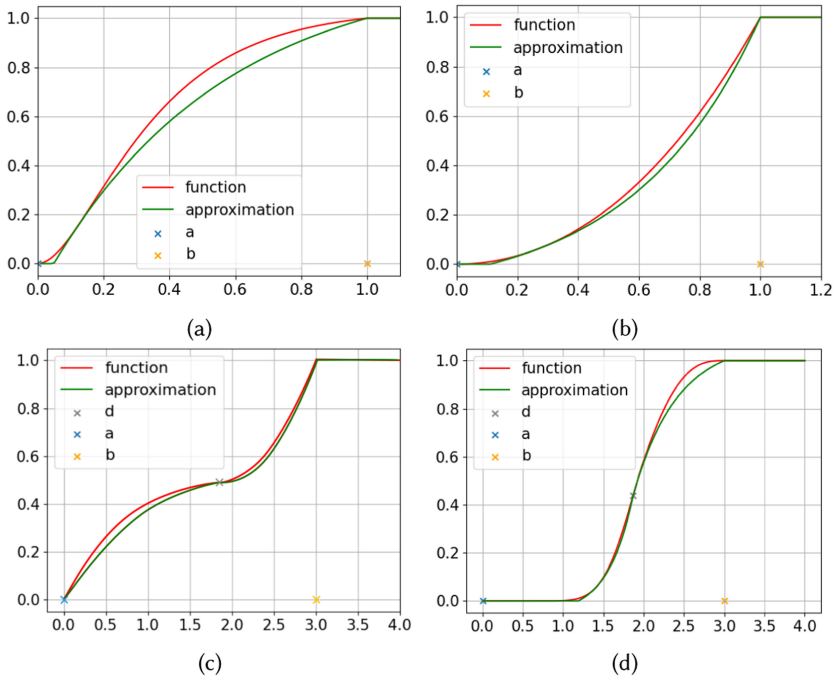

Fig. 3. Stochastic upper bound CDF: (a) fixed downward concavity, (b) fixed upward concavity, (c) changing concavity downward to upward, and (d) changing concavity upward to downward, of the approximated CDF.

such that

$$\hat{F}(x) = F(x_{i_{j-1}}) + [F(x_{i_j}) - F(x_{i_{j-1}})]\frac{1 - e^{-\lambda_j(x - x_{i_{j-1}})}}{1 - e^{-\lambda_j(x_{i_j} - x_{i_{j-1}})}} \quad \text{if } x \in [x_{i_{j-1}}, x_{i_j}], \text{ for all } j = 1, \ldots, M \quad (4)$$

where, for a downward (upward) concavity over $[x_{i_{j-1}}, x_{i_j}]$, $\lambda_j \in \mathbb{R}$ *is the largest positive (smallest negative) value such that* $\hat{F}(x) \leqslant F(x)\ \forall\, x \in [x_{i_{j-1}}, x_{i_j}]$ *is stochastically larger than* X, *i.e.,* $\hat{X} \geqslant_{st} X$.

4.3 Approximation Safety

We ensure that our compositional analysis method is safe when workflows are used to guarantee soft deadlines of SLAs. Specifically, our proofs (reported in the Appendix) hinge on the idea of *stochastic order* (recalled in Section 4.2) and on the following lemma on the order of independent replication of positively correlated random variables [2].

Lemma 4.3 (Order of Independent Replicas Under Positive Correlation). *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be a vector of positively correlated random variables, i.e.,* $\mathrm{Cov}[f(\mathbf{X}), g(\mathbf{X})] \geqslant 0$ *holds for all monotone nondecreasing functions* $f, g\colon \mathbb{R}^n \to \mathbb{R}$. *Then,* $\overline{\mathbf{X}}$ *is larger than* $\mathbf{X}$ ($\overline{\mathbf{X}} \geqslant_{st} \mathbf{X}$), *where* $\overline{\mathbf{X}}$ *is a vector of independent random variables with* $\overline{X}_i \sim X_i$ *for all* i.

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

Lemma 4.4 (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. $T(n)$ is lower than $T(n')$ ($T(n) \leqslant_{st} T(n')$), 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 node 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 of the workflow.

Lemma 4.5 (Stochastic Order of Inner Block Replication). *Given a DAG block $G = (V, E, v_I, v_F)$ and a node $v \in V$, 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, $T(v'_F) \geqslant_{st} T(v_F)$.*

5 EXPERIMENTATION

In this section, we consider a quantitative measure to evaluate the analysis accuracy with respect to a ground truth obtained by simulation (Section 5.1); we compare accuracy and complexity of the heuristics using sets of artificially and manually generated models (Section 5.2); we assess the approach scalability by significantly increasing the workflow complexity (Section 5.3); and, we evaluate how the accuracy of the heuristics varies with respect to the stochastic upper bound PDF used in Reference [10] to approximate the response time PDF of subworkflows (Section 5.4). The approach is implemented in the Eulero Java library [11], which supports definition of stochastic workflows, derivation of their response time PDFs, and random generation of workflows, exploiting the SIRIO library [31] of the ORIS tool [25] to build STPN blocks and evaluate accuracy. Experiments are performed on a single core of an Intel Xeon Gold 5120 CPU (2.20 GHz) with 32 GB of RAM.

5.1 Ground Truth and Accuracy Measure

The accuracy of the workflow response time PDF is evaluated with respect to a **ground truth (GT)** obtained by a 5-million-run simulation of the workflow STPN using the **Jensen–Shannon (JS)** Divergence [22, 24], which quantifies the distance between two PDFs f_a and f_b as

$$D_{JS}\left(f_{\mathrm{a}} \,||\, f_{\mathrm{b}}\right) = \frac{1}{2} D_{KL}\left(f_{\mathrm{a}} \,||\, Z\right) + \frac{1}{2} D_{KL}\left(f_{\mathrm{b}} \,||\, Z\right), \tag{5}$$

where $Z(t) = \frac{1}{2}\left(f_{\mathrm{a}}(t) + f_{\mathrm{b}}(t)\right) \forall\, t \in \Omega$ 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 $D_{KL}\left(\cdot \,||\, \cdot\right)$ is the **Kullback–Leibler (KL)** divergence defined as

$$D_{KL}\left(f_{\mathrm{a}} \,||\, f_{\mathrm{b}}\right) = \sum_{t \in \Omega} f_{\mathrm{a}}(t) \cdot \log\left(\frac{f_{\mathrm{b}}(t)}{f_{\mathrm{a}}(t)}\right). \tag{6}$$

To select the number of simulation runs needed to evaluate the ground truth, for each model randomly generated in Section 5.2.1, we performed 1-million-run, 2-million-run, ..., 5-million-run simulations, using a time tick nearly three orders of magnitude lower than the width of the support of the workflow response time. Then, we evaluated the JS divergence of the workflow response time PDF provided by each experiment with respect to the one computed by the 5-million-run simulation. Experimental results show that, for each model where both heuristics perform multiple approximations, the JS divergence of the 4-million-run simulation with respect to the 5-million-run simulation converges to a value that is at least one or two orders of magnitude lower than

the JS divergence of the heuristics from the 5-million-run simulation, which is sufficient for the context of use and indicates that a 5-million-run simulation can be considered as the ground truth.

5.2 Comparing the SDF and the RBF Heuristics

5.2.1 Models Analyzed More Accurately by the SDF Heuristics. We randomly generated a set of models by controlling the following parameters that characterize the structure tree: the depth D of the structure tree; the number B of concurrent and alternative blocks in AND and XOR blocks, respectively; and the number T of sequential blocks in SEQ blocks. Each model is generated by keeping the value of two parameters at 2, and varying the remaining parameters within $\{2, 4, 6\}$, leading to seven different models. Then, each model was implemented in two variants, one allocating DAG blocks on the bottom level of the structure tree, and the other one allocating them on the top level, for a total amount of 14 models. The type of the remaining blocks was randomly drawn, giving AND and SEQ blocks higher probability than XOR blocks. DAG blocks were randomly generated, too, assuming that they consisted of a maximum of seven blocks connected through paths having maximum length equal to 3. Simple activities have uniform duration PDF over $[0, 1]$.

Models were evaluated using the following threshold values Θ_c and Θ_q on the concurrency and the sequencing degree, respectively, of the workflow TPN: For the SDF heuristics, $\Theta_c = 3$ for models with DAGs at the bottom level and $\Theta_c = 2$ otherwise, and $\Theta_q = 7$; for the RBF heuristics, $\Theta_c = 3$ and $\Theta_q = 7$. Models were also evaluated by simulations (S) of the workflow STPN lasting as long as the analysis with the SDF heuristics. The workflow response time PDF computed by S is also averaged using a sliding window of width 3, which is found to be the value that produces the most accurate results, yielding a PDF referred to as the result of **Averaged Simulation (AS)**.

Table 1 reports the values of the JS divergence and computation times obtained by evaluating models having the DAG blocks at the bottom level (BOTTOM) and at the top level (TOP) of the structure tree for the mentioned techniques, and Figures 4 and 5 plot the workflow response time PDFs. For the BOTTOM set of models, the heuristics prove to be extremely efficient in terms of computation time as they are able to evaluate models of increasing complexity in minimal times, never exceeding 3 s. As evident both from the JS values and the PDFs of Figures 4 and 5, the two heuristics produce the same very accurate results, which is due to the model topology: Given that the models are well-nested in all the levels except for the bottom one, and that the bottom-level DAGs do not have composite internal blocks and are not complex to analyze, both heuristics solve the DAGs by forward transient analysis and then the rest of the model by numerical analysis, without introducing approximation. Note that the heuristics achieve better JS values than those obtained by the simulation, which are three orders of magnitude larger in the majority of the cases. The averaged simulation, obtained from the simulation using the optimal width of the sliding window according to the ground truth, improves the accuracy of the simulation, though remaining in the same order of magnitude. Also note that the JS divergence tends to smooth out the fluctuations of both types of simulation, which are instead clearly visible in the response time PDFs of Figures 4 and 5, where the heuristics are significantly better at approximating the ground truth.

In the TOP set of models, the computation times are again lower than 3 s, except for the D2B2T2 model for the RBF heuristics and the D2B2T4 model for the SDF heuristics, for which they are is in the order of 5 min and 10 s, respectively, due to complex DAGs (in terms of concurrency and sequencing degrees) that challenge forward transient analysis. Nevertheless, the computation times remain at least one order of magnitude lower than the time needed to achieve the same accuracy by simulation. The heuristics in fact prove to be very accurate, with JS values lower than those of simulation by at least one order of magnitude and up to four orders of magnitude. For the D2B6T2 model, the RBF heuristics achieves a JS value in the order of 10^{-1}, comparable

Table 1. JS Divergence and Computation Times of SDF Heuristic, RBF Heuristic, Simulation (S), and AS for the BOTTOM and the TOP Configuration Models of Section 5.2, Randomly Generated Using Different Values of the Depth D of the Structure Tree, the Number B of Concurrent and Alternative Blocks in AND and XOR Blocks, Respectively, and the Number T of Sequential Blocks in SEQ Blocks

BOTTOM									
D, B, T	**JS**				**Time**				
	SDF	**RBF**	**S**	**AS**	**SDF**	**RBF**	**S**	**AS**	**GT**
2 2 2	**0.00001**	**0.00001**	0.02915	0.01800	0.69s	0.39s	0.69s	0.69s	5518.22s
4 2 2	**0.00001**	**0.00001**	0.07012	0.04065	1.82s	1.70s	1.83s	1.83s	23268.95s
6 2 2	**0.00005**	**0.00005**	0.10118	0.04272	2.47s	2.26s	2.50s	2.50s	190396.48s
2 4 2	**0.00001**	**0.00001**	0.05532	0.03476	0.36s	0.33s	0.37s	0.37s	3167.33s
2 6 2	**0.00001**	**0.00001**	0.06945	0.04075	0.32s	0.28s	0.32s	0.32s	4573.85s
2 2 4	**0.00001**	**0.00001**	0.04672	0.02770	0.63s	0.43s	0.64s	0.64s	4526.35s
2 2 6	**0.00000**	**0.00000**	0.07568	0.05331	0.21s	0.12s	0.26s	0.26s	2344.53s
TOP									
D, B, T	**JS**				**Time**				
	SDF	**RBF**	**S**	**AS**	**SDF**	**RBF**	**S**	**AS**	**GT**
2 2 2	0.00188	**0.00151**	0.17183	0.09817	0.08s	277.53s	0.09s	0.09s	3618.21s
4 2 2	**0.01411**	**0.01411**	0.08571	0.05109	2.65s	1.59s	2.71s	2.71s	20660.48s
6 2 2	**0.00004**	0.01365	0.36369	0.24701	0.32s	1.01s	0.36s	0.36s	195106.73s
2 4 2	**0.00008**	0.00103	0.22228	0.13425	0.13s	6.89s	0.13s	0.13s	6513.53s
2 6 2	**0.00240**	0.19732	0.23625	0.13771	0.08s	10.44s	0.10s	0.10s	3347.82s
2 2 4	**0.00091**	**0.00091**	0.00223	0.00593	12.44s	8.95s	12.48s	12.48s	3238.71s
2 2 6	**0.00054**	0.01023	0.09563	0.05805	0.09s	0.30s	0.09s	0.09s	6040.17s

For each model (i.e., for each row), the best (i.e., lowest) JS value is highlighted in bold. The last column shows the computation time of the GT.

to that of simulation and averaged simulation, due to the fact that the workflow decomposition requires to execute inner block analysis three times, approximating PDFs that exhibit cusp points. The averaged simulation improves the simulation, but the JS values remain in the same order of magnitude. Finally, the SDF heuristics achieves lower JS values than the RBF heuristics in the majority of the cases, which is due to the fact that, in these cases, the DAGs are internally complex and thus the analysis of individual blocks in isolation is not sufficient to reduce complexity and make forward transient analysis affordable, forcing the RBF heuristic to perform both separate analysis of individual blocks and replication of shared dependencies of selected blocks. Overall, the proposed approach significantly outperforms both simulation and averaged simulation in accuracy and complexity, notably computing a duration PDF that is a stochastic upper bound on the actual workflow response time PDF, which instead cannot be guaranteed by simulation.

5.2.2 Models Analyzed More Accurately by the RBF Heuristics. We hand-crafted seven models M1, …, M7 with the aim of demonstrating cases where the RBF heuristics outperforms the SDF heuristics. With this purpose, the top-level block of each model is a DAG that is internally simple (i.e., the sequencing and the concurrency degrees of the workflow unexpanded TPN do not exceed their respective thresholds θ_s and θ_c for the RBF heuristics, respectively), each DAG has a maximum of two shared activities, and the top-level DAG complexity is obtained by embedding complex sub-workflows in the activities that are shared predecessors of multiple nodes. Moreover, to increase the correlation between the response time of the workflow and the response times of activities that are shared predecessor of multiple nodes, all the simple activities contained in (composite) shared predecessors of some node have uniformly distributed response time between

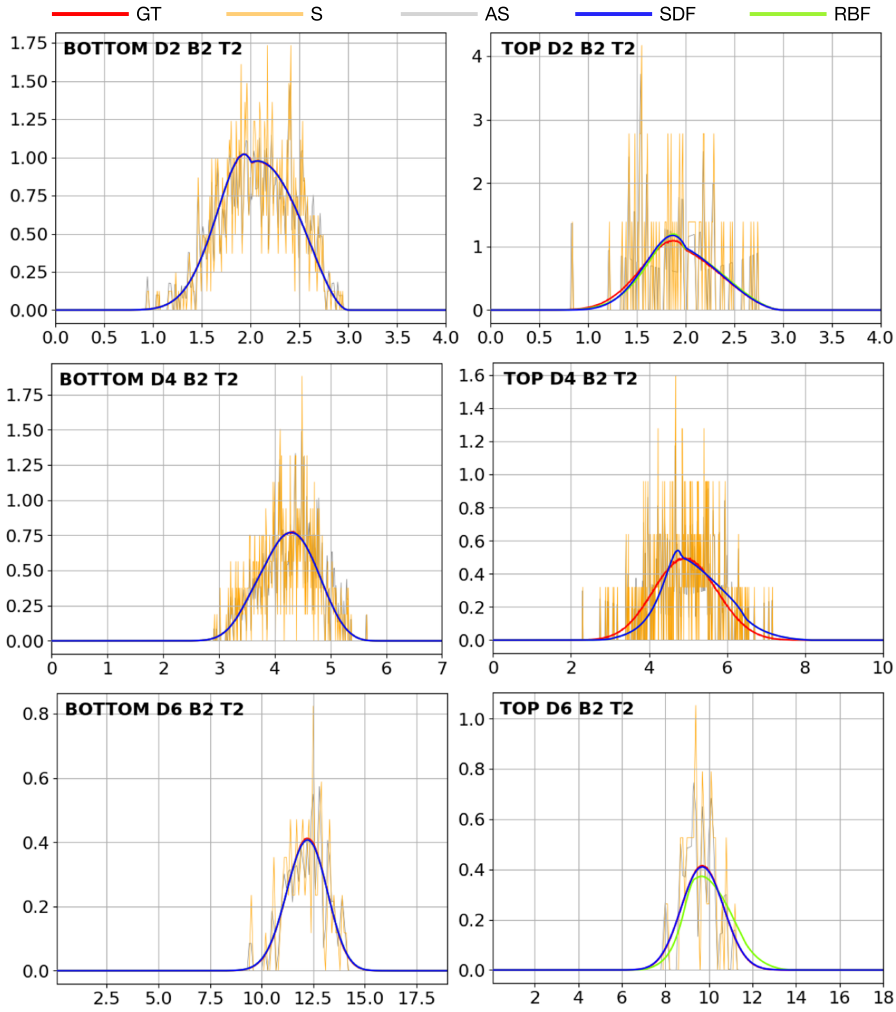

Fig. 4. Response time PDFs of SDF heuristics, RBF heuristics, simulation (S), AS, and GT for the BOTTOM and TOP models of Section 5.2.1, with different values of the depth D of the structure tree, $B = 2$ concurrent and alternative blocks in AND and XOR blocks, respectively, and $T = 2$ of sequential blocks in SEQ blocks.

4 and 8. In particular, M1 has two DAG blocks at the bottom level, M2 is a variant of M1 with more precedence relations in the top-level DAG, M3 has also two bottom-level DAG blocks, M4 is a variant of M3 with more composite blocks in the top-level DAG, M5 is a variant of M1 with a middle-level DAG (and no bottom-level DAG), M6 has a top-level DAG with more precedence relations than the previous models, and M7 is a variant of M6 with one more composite block in the top-level DAG.

We evaluated the accuracy of the heuristics, simulation, and averaged simulation with respect to the ground truth, with the same experimental setup of Section 5.2.1, except for the simulation time and averaged simulation time, which is at least equal to that of the RBF heuristics (i.e., the most accurate heuristics for the benchmark). Results are shown in Table 2 and Figure 6. The SDF heuristics performs the analysis in relatively less time than the RBF heuristics, though results are comparable. As expected, the RBF heuristics yields more accurate results than the SDF heuristics,

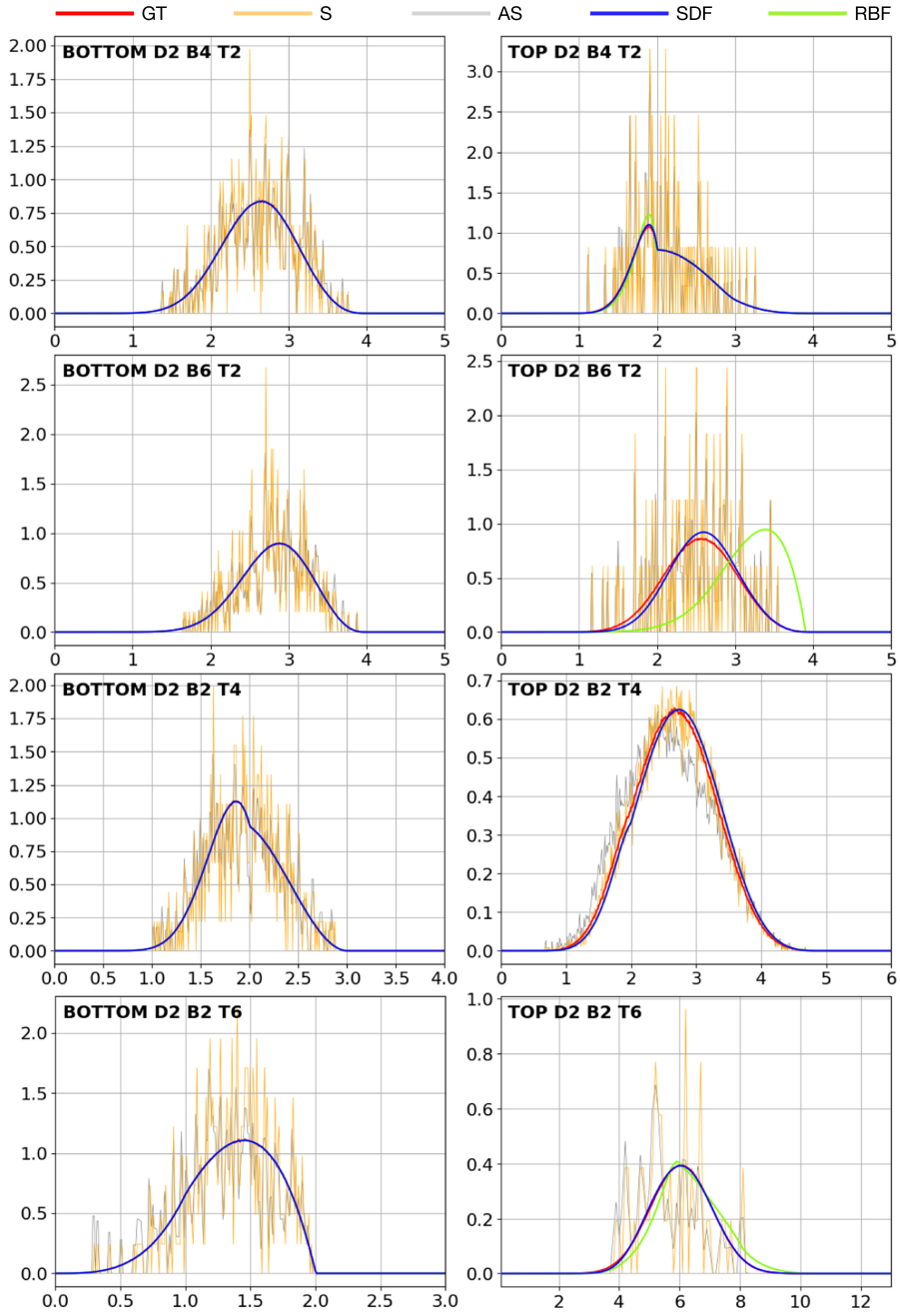

Fig. 5. Response time PDFs of SDF heuristics, RBF heuristics, simulation (S), AS, and GT for the BOTTOM and TOP models of Section 5.2.1 with depth $D = 2$ of the structure tree, and different values of the number B of concurrent and alternative blocks in AND and XOR blocks, respectively, and the number T of sequential blocks in SEQ blocks.

Table 2. JS Divergence and Computation Times of SDF Heuristic, RBF Heuristic, Simulation (S), and AS for the Models of Section 5.2.2

Model	JS				Time				
	SDF	RBF	S	AS	SDF	RBF	S	AS	GT
1	0.04506	**0.01254**	0.03382	0.02127	1.42s	1.28s	1.31s	1.31s	9287.89s
2	0.04526	**0.01256**	0.05248	0.03376	0.46s	0.64s	0.66s	0.66s	9549.46s
3	0.03756	**0.00665**	0.04640	0.04326	0.20s	0.31s	0.33s	0.33s	5776.45s
4	0.03737	**0.00660**	0.01350	0.01126	0.16s	0.85s	0.86s	0.86s	6657.49s
5	0.12228	**0.00909**	0.05926	0.03398	0.21s	0.22s	0.22s	0.22s	6692.07s
6	0.03837	**0.00820**	0.01449	0.01015	0.13s	0.84s	0.86s	0.86s	6050.73s
7	0.03962	**0.00456**	0.00434	0.00852	0.21s	3.08s	3.09s	3.09s	8979.62s

For each model (i.e., for each row), the best (i.e., lowest) JS value is highlighted in bold.

Table 3. Computation Times of SDF Heuristic, RBF Heuristic, Simulation (S), and AS for the BOTTOM and TOP Models of Section 5.3, with Structure Tree Depth $D \in \{4, 6\}$, $B = 4$ Concurrent and Alternative Blocks in AND and XOR Blocks, Respectively, and $T = 4$ Sequential Blocks in SEQ Blocks

DAG D, B, T	SDF	RBF	S	AS
BOTTOM 4, 4, 4	3.04 s	15.63 s	3.07 s	3.07 s
TOP 4, 4, 4	1.25 s	2.44 s	1.28 s	1.28 s
BOTTOM 6, 4, 4	33.25 s	156.94 s	42.16 s	42.16 s
TOP 6, 4, 4	22.92 s	5.73 s	38.66 s	38.66 s

with a JS divergence gain of at least a factor of 4, and of nearly two orders of magnitude in the best cases. Overall, we were not able to randomly generate a benchmark of models for which the RBF heuristics significantly outperforms the SDF heuristics, confirming that the RBF heuristics is more accurate only under very restrictive conditions (discussed at the end of Section 4.1). Therefore, except when these conditions occur, the SDF heuristics is preferable with respect to the RBF heuristics.

5.3 Increasing Workflow Complexity

We further stressed the evaluation complexity by generating workflows with parameters $D = 4$, $B = 4$, $T = 4$ and $D = 6$, $B = 4$, $T = 4$, notably obtaining huge models having up to 448 and 7,168 simple activities, respectively, for which obtaining a ground truth via stochastic simulation is definitely not viable. The computation times are shown in Table 3 and the analysis results in Figure 7. Although the computation times increase with respect to the cases described in Section 5.2, the obtained results highlight that extremely complex models can be evaluated in relatively short times. In particular, for the majority of the models with structure tree depth $D = 4$, the computation times of the heuristics do not exceed 4 s. Notably, the results obtained by simulation during such amount of time are too noisy to represent a valid alternative to the proposed analysis heuristics. Though obtaining accurate simulation results is not viable for these complex models, rare event simulation methods [7, 8, 28, 38] could be applied to evaluate rewards that focus on selected behaviors.

For models with structure tree depth $D = 6$, the computation times of the heuristics slightly increase, but always without exceeding 35 s, except for the BOTTOM D6B4T4 model, for which the RBF heuristics executes in nealry 157 s (which is still affordable). Despite this, the results achieved by simulation are much worse than those obtained for the models with depth $D = 4$, due to

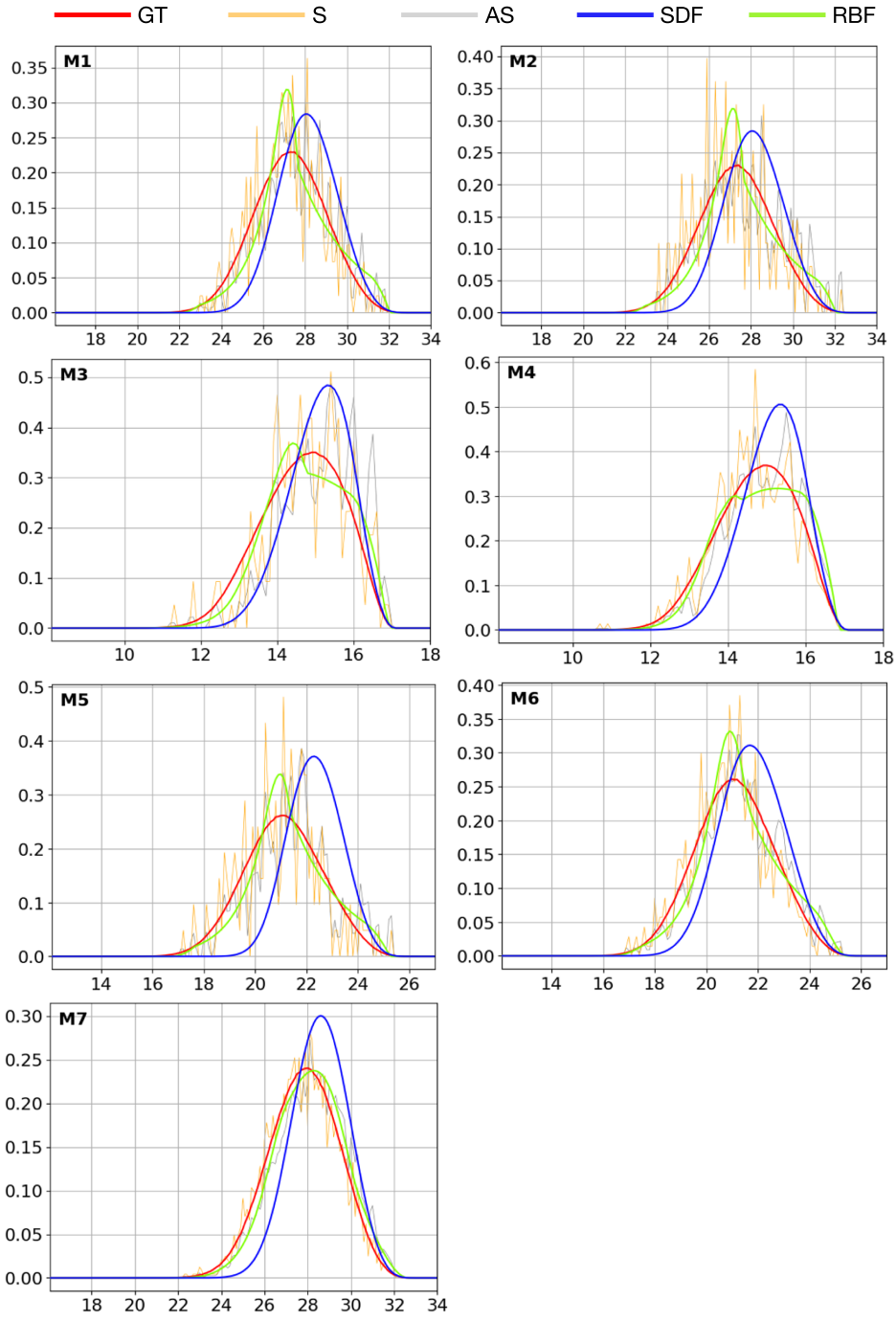

Fig. 6. Response time PDFs of SDF heuristic, RBF heuristic, simulation (S), AS, and GT for the models of Section 5.2.2.

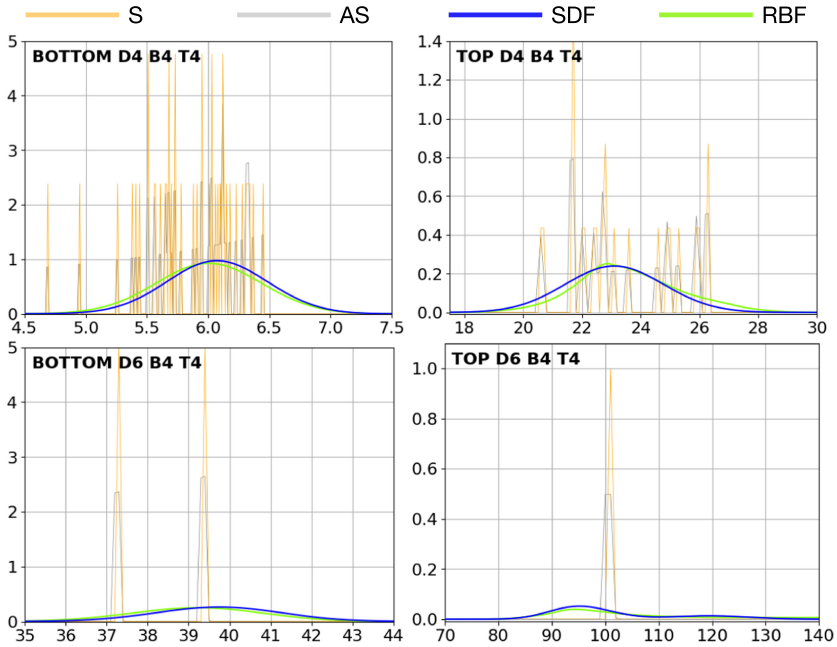

Fig. 7. Response time PDFs of SDF heuristic, RBF heuristic, simulation (S), and AS for the BOTTOM and TOP models of Section 5.3, with structure tree depth $D \in \{4, 6\}$, $B = 4$ concurrent and alternative blocks in AND and XOR blocks, respectively, and $T = 4$ sequential blocks in SEQ blocks.

the significant increase in the time needed to complete a simulation run (simulation time is the minimum time, larger than the analysis time, that is needed to perform an integer number of runs).

5.4 Sensitivity to the Stochastic Upper Bound PDF

We performed a sensitivity analysis with respect to the stochastic upper bound PDF used to approximate the numerical solutions of inner blocks. To this end, we considered two randomly generated models with parameters $D = 4, B = 2, T = 2$ and $D = 6, B = 2, T = 2$, respectively, and DAG blocks at the top level consisting of a maximum of seven blocks connected through paths of maximum length equal to 2. Due to the model structure and parameters, both heuristics apply inner block analysis rather than inner block replication, and thus these models are evaluated through the SDF heuristics. We performed two experiments: In the first case, we used the approximation defined in Section 4.2 (A1); in the second case, we used a variant with bounded support of the approximant proposed in Reference [10], approximating numerical PDFs with support $[a, b]$ with a truncated Exponential PDF with support $[\delta, b]$, defined as $p(t) := \lambda e^{-\lambda(t-\delta)}/(1 - e^{-\lambda(b-\delta)})$, where δ is the intersection point of the x-axis with the line that is tangent to the inflection point of the approximated CDF and λ is the rate of the Exponential, computed to impose the stochastic upper bound with respect to the approximated function (A2). The accuracy of the resulting PDFs is evaluated using JS divergence with respect to a ground truth obtained by 5-million-run simulation.

For both models, the SDF heuristics with the approximant A1 obtains JS values at least four times lower than with approximant A2 (i.e., 0.01023 for the D4B2T2 model and 0.01680 for the D6B2T2 model, compared to 0.04164 and 0.07939, respectively). As expected, the PDFs in Figure 8 show that A1 results in a curve that is visually closer to the ground truth, pointing out that the stochastic upper bound defined in Section 4.2 is more accurate than the alternative.

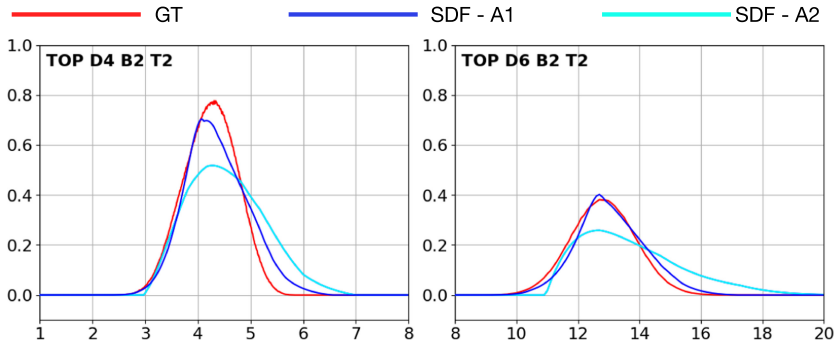

Fig. 8. Response time PDFs of the GT and SDF heuristics using the approximant PDF of Section 4.2 (A1) and a variant with bounded support of the one proposed in Reference [10] (A2) for the models of Section 5.4, with structure tree depth $D \in \{4, 6\}$, $B = 2$ concurrent and alternative blocks in AND and XOR blocks, respectively, $T = 2$ sequential blocks in SEQ blocks, and DAG blocks at the top level.

6 CONCLUSIONS

We presented a compositional approach to efficiently compute an accurate stochastic upper bound on the response time PDF of complex workflows. A workflow is specified as a composition of STPN blocks to enable hierarchical decomposition into subworkflows, evaluated by combining numerical analysis and forward transient analysis according to heuristics that achieve different tradeoffs between accuracy and complexity. Experiments on suites of manually and randomly generated models of increasing complexity show that the approach achieves sufficient accuracy in a very limited computation time, notably outperforming simulation having the same computation time. Moreover, note that the approach derives a response time PDF that is proved to be a stochastic upper bound on the actual workflow response time PDF, which cannot be guaranteed by simulation.

The workflow model could be extended with other constructs, possibly affecting well-formed nesting, provided that positive correlation is guaranteed among the response times of different subworkflows. The approach is open to the definition of other heuristics to explore the structure tree and decompose the workflow, to the exploitation of other analytical approximants in the class of expolynomial functions or piecewise CPHs over bounded supports [21] to fit numerical PDFs, and to the integration of other solution techniques to evaluate the response time PDF of a block. 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 PDF.

APPENDICES

A STOCHASTIC TIME PETRI NETS

A.1 Syntax

STPNs [37] model concurrent stochastic systems. As shown in Figure 1(a), transitions are depicted as bars and represent the stochastic duration of activities (e.g., transition `Q`); tokens within places are represented as dots within circles, respectively, and model the discrete logical state of the system (e.g., place `p0` contains one token); directed arcs from input places to transitions, and from transitions to output places, are depicted as directed arrows and model precedence relations among activities (e.g., place `p3` is an input place for transition `Q`).

Formally, 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 associates each transition $t \in T$ with a 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 $x > LFT(t)$; W and Z associate each transition $t \in T$ with a weight $W(t) \in \mathbb{R}_{>0}$ and a priority $Z(t) \in \mathbb{N}$, respectively. Other features, such as inhibitor arcs, enabling functions, and update functions, could be added to the model and are actually supported by the ORIS tool [25].

As in Petri nets, for a transition $t \in T$, a place $p \in P$ is termed an *input* place if $(p, t) \in A^-$ and is termed an *output* place if $(t, p) \in A^+$. As in stochastic Petri nets, a transition t is termed IMM if $EFT(t) = LFT(t) = 0$ and *timed* otherwise; a timed transition t is termed EXP if $F_t(x) = 1 - \exp(-\lambda x)$ for some rate $\lambda \in \mathbb{R}_{>0}$, and GEN otherwise. For each GEN transition t, we assume that F_t is the integral function of a 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 the Dirac impulse function $f_t(y) = \delta(y - \overline{y})$ as generalized PDF, with $\overline{y} = EFT(t) = LFT(t)$. In particular, IMM transitions are depicted as thin black bars (e.g., transition `t0`), and GEN transitions as thick black-filled bars (e.g., transition `Q` has non-EXP CDF possibly with bounded support).

A.2 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 (e.g., transition `t0` is enabled). Upon enabling, a transition samples a time-to-fire according to its CDF F_t. When the time-to-fire of an enabled transition has elapsed, the transition becomes *firable*. When a transition fires, one token is removed from each of its input places and one token is added to each of its output places (e.g., the firing of transition `t0` removes one token from place `p0` and adds one token to each of places `p1` and `p2`), and the times-to-fire of the transitions enabled before and after the token moves are reduced by the time-to-fire of the fired transition (e.g., when transition `Q` fires before transition `X1`, the time-to-fire of `X1` is reduced by the time-to-fire of `Q`). Formally, when a transition t fires in a state $s = \langle m, \tau \rangle$, s is replaced by a new state $s' = \langle m', \tau' \rangle$, 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').

If multiple transitions are firable in a state $s = \langle m, \tau \rangle$, then the next transition to fire is determined by a random switch based on probabilistic weights and priorities associated with transitions, i.e., the next transition t to fire is selected with probability $W(t)/\sum_{t_i \in E} W(t_i)$ from the set E of transitions that are enabled by m and have time-to-fire equal to zero and maximum priority. Given that this condition occurs only in limit cases of synchronization among IMM or DET transitions, weights and priorities are not depicted in the graphical representation of STPNs, and, if omitted, weights are assumed to be equal to 1 and priorities are assumed to be equal to 0.

B NUMERICAL ANALYSIS

For blocks composing independent subworkflows by SEQ, AND, and XOR operators, the numerical form of the response time PDF can be derived by bottom-up composition of the response time PDFs or CDFs of the blocks. Specifically, given n blocks $b_1, \ldots, b_n$ with response time PDFs $\phi_1(t), \ldots, \phi_n(t)$, respectively, and response time CDFs $\Phi_1(t), \ldots, \Phi_n(t)$, respectively: the response

time PDF $\phi_{\text{seq}}(t)$ of a SEQ block made of $b_1, \ldots, b_n$ is derived through subsequent convolutions of $\phi_1(t), \ldots, \phi_n(t)\ \forall\, t \in [0, t_{\max}]$, i.e., $\phi_{\text{seq}}(t) = \phi_1(t) \circledast \phi_2(t) \circledast \cdots \circledast \phi_n(t)$, where the convolution of PDFs $\phi_i(t)$ and $\phi_j(t)$ is $\phi_i(t) \circledast \phi_j(t) = \int_0^t \phi_i(\tau)\, \phi_j(t-\tau)\, d\tau$; 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 $\Phi_1(t), \ldots, \Phi_n(t)$ according to $p_1, \ldots, p_n$, respectively, $\forall\, t \in [0, t_{\max}]$, i.e., $\Phi_{\text{xor}}(t) = p_1\, \Phi_1(t) + \cdots + p_n\, \Phi_n(t)$; and, 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 $\Phi_1(t), \ldots, \Phi_n(t)\ \forall\, t \in [0, t_{\max}]$ given that the response times of $b_1, \ldots, b_n$ are independent random variables, $\Phi_{\text{and}}(t) = \Phi_1(t) \cdot \ldots \cdot \Phi_n(t)$.

Then, the response time PDF ϕ_{and} and ϕ_{xor} of an AND and an XOR block, respectively, can be obtained as the derivative of the response time CDF of the block, e.g., $\phi_{\text{and}}(t) = d/dt\, \Phi_{\text{and}}(t)$, e.g., for block R in Figure 1, the response time PDF is $\phi_{\mathsf{R}}(t) = d/dt\, (\Phi_{\mathsf{R1}}(t) \cdot \Phi_{\mathsf{R2}}(t))$, where $\Phi_{\mathsf{R1}}(t) = d/dt\, (\Phi_{\mathsf{X}}(t) \cdot \Phi_{\mathsf{Y}}(t))$ and $\Phi_{\mathsf{R2}}(t) = p_{\mathsf{R2A}} \Phi_{\mathsf{R2A}}(t) + p_{\mathsf{R2B}} \Phi_{\mathsf{R2B}}(t)$ are the response time CDFs of blocks R1 and R2, respectively, and, $\Phi_{\mathsf{X}}(t) = d/dt\, (\Phi_{\mathsf{X}_1}(t) \cdot \Phi_{\mathsf{X}_2}(t))$ and $\Phi_{\mathsf{Y}}(t) = \int_0^t \int_0^\tau \phi_{\mathsf{Y1}}(x)\, \phi_{\mathsf{Y2}}(\tau - x)\, dx\, d\tau$ are the response time CDFs of blocks X and Y, respectively, and, finally, $\Phi_{\mathsf{X}_1}(t)$, $\Phi_{\mathsf{X}_2}(t)$, $\Phi_{\mathsf{Y}_1}(t)$, $\Phi_{\mathsf{Y}_1}(t)$, $\Phi_{\mathsf{R2A}}(t)$, and $\Phi_{\mathsf{R2B}}(t)$ are the response time CDFs of blocks X_1, X_2, Y_1, Y_2, R2A, and R2B, respectively.

C THEOREM PROOFS

Proof of Lemma 4.2. By construction, $\hat{F}(x) \leqslant F(x)\ \forall\, x \in D \cap [a, b]$. Indeed, for every sub-support $[x_{i_{j-1}}, x_{i_j}]$ between two successive inflection points $x_{i_{j-1}}$ and x_{i_j}, $F(x)$ has downward or upward concavity. In the first case, the rate of the truncated exponential of Equation (4) is taken as a positive value resulting in a negative rate. Hence, approximation $\hat{F}(x)$ has downward concavity too. Moreover, the rate is chosen as the smallest value for which stochastic ordering with respect to $F(x)$ is guaranteed $\forall\, x \in D \cap [x_{i_{j-1}}, x_{i_j}]$. Then, stochastic ordering is guaranteed for all supports $[x_{i_{j-1}}, x_{i_j}]$ where $F(x)$ has downward concavity. When $F(x)$ has upward concavity, the rate of the truncated exponential of Equation (4) is taken as a negative value resulting in a positive rate. Approximation $\hat{F}(x)$ has upward concavity, too, and a rate chosen as the largest value for which stochastic ordering with respect to $F(x)$ is guaranteed $\forall\, x \in D \cap [x_{i_{j-1}}, x_{i_j}]$. Then, stochastic ordering is guaranteed for all supports $[x_{i_{j-1}}, x_{i_j}]$ where $F(x)$ has upward concavity. Therefore, $\hat{X} \geqslant_{st} X$. □

Proof of Lemma 4.4. The duration of the subworkflow associated with any node m (SEQ, AND, XOR, REPEAT, DAG) is a monotone nondecreasing function of the durations of the subworkflows 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'. □

Proof of Lemma 4.5. 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 4.3, we obtain $T(v_F') \geqslant_{st} T(v_F)$. □

REFERENCES

[1] Florian Arnold, Holger Hermanns, Reza Pulungan, and Mariëlle Stoelinga. 2014. Time-dependent analysis of attacks. In *Proceedings of the International Conference on Principles of Security and Trust*. Springer, 285–305.

[2] François Baccelli and Armand M. Makowski. 1989. Multidimensional stochastic ordering and associated random variables. *Operat. Res.* 37, 3 (1989), 478–487.

[3] B. Berthomieu and M. Diaz. 1991. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Soft. Eng.* 17, 3 (1991), 259–273. https://doi.org/10.1109/32.75415

[4] Andrea Bobbio and Miklos Telek. 1995. Markov regenerative SPN with non-overlapping activity cycles. In *Proceedings of the International Computer Performance and Dependability Symposium*. 124–133.

[5] Paolo Bocciarelli, Andrea D'Ambrogio, Andrea Giglio, and Emiliano Paglia. 2020. Modeling resources to simulate business process reliability. *ACM Trans. Model. Comput. Simul.* 30, 3 (2020), 1–25.

[6] Dario Bruneo, Salvatore Distefano, Francesco Longo, and Marco Scarpa. 2012. Stochastic evaluation of QoS in service-based systems. *IEEE Transactions on Parallel and Distributed Systems* 24, 10 (2012), 2090–2099.

[7] James Bucklew. 2013. *Introduction to Rare Event Simulation*. Springer Science & Business Media.

[8] Carlos E. Budde, Marco Biagi, Raúl E. Monti, Pedro R. D'Argenio, and Mariëlle Stoelinga. 2020. Rare event simulation for non-markovian repairable fault trees. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'20)*. Springer, 463–482.

[9] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. 2005. QoS-aware replanning of composite web services. In *Proceedings of the IEEE International Conference on Web Services*. IEEE, 121–129.

[10] Laura Carnevali, Marco Paolieri, Riccardo Reali, and Enrico Vicario. 2021. Compositional safe approximation of response time distribution of complex workflows. In *Proceedings of the International Conference on Quantitative Evaluation of Systems (QEST'21)*, Vol. 12846. Springer, 83–104.

[11] Laura Carnevali, Riccardo Reali, and Enrico Vicario. 2022. Eulero: A tool for quantitative modeling and evaluation of complex workflows. In *Proceedings of the International Conference on Quantitative Evaluation of Systems (QEST'22)*.

[12] Laura Carnevali, Riccardo Reali, and Enrico Vicario. 2021. Compositional evaluation of stochastic workflows for response time analysis of composite web services. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 177–188.

[13] Gianfranco Ciardo, Reinhard German, and Christoph Lindemann. 1994. A characterization of the stochastic process underlying a stochastic Petri net. *IEEE Trans. Softw. Eng.* 20, 7 (1994), 506–515.

[14] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. 2003. Business process execution language for web services. http://xml.coverpages.org/BPELv11-May052003Final.pdf.

[15] Ton G. de Kok and Jan C. Fransoo. 2003. Planning supply chain operations: Definition and comparison of planning concepts. In *Handbooks in Operations Research and Management Science*, Vol. 11. Elsevier, 597–675.

[16] David L. Dill. 1990. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the Annual Conference on Automatic Verification Methods for Finite State Systems (AVMFSS'89), Lecture Notes in Computer Science*, Vol. 407. Springer, 197–212. https://doi.org/10.1007/3-540-52148-8_17

[17] Alim U. Gias, André van Hoorn, Lulai Zhu, Giuliano Casale, Thomas F. Düllmann, and Michael Wurster. 2020. Performance engineering for microservices and serverless applications: The RADON approach. In *Companion of the ACM/SPEC International Conference on Performance Engineering*. 46–49.

[18] András Horváth, Marco Paolieri, Lorenzo Ridi, and Enrico Vicario. 2012. Transient analysis of non-Markovian models using stochastic state classes. *Perf. Eval.* 69, 7-8 (2012), 315–335.

[19] E. Douglas Jensen, C. Douglas Locke, and Hideyuki Tokuda. 1985. A time-driven scheduling model for real-time operating systems. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS'85)*, Vol. 85. 112–122.

[20] Richard Johnson, David Pearson, and Keshav Pingali. 1994. The program structure tree: Computing control regions in linear time. In *ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI'94)*. ACM, 171–185.

[21] L'uboš Korenčiak, Jan Krčál, and Vojtěch Řehák. 2014. Dealing with zero density using piecewise phase-type approximation. In *European Workshop on Performance Engineering*. Springer, 119–134.

[22] Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory* 37, 1 (1991), 145–151.

[23] Yanjie Liu, Zheng Zheng, and Jiantao Zhang. 2019. Markov model of web services for their performance based on phase-type expansion. In *Proceedings of the IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, and International Conference on Cyber Science and Technology Congress (DASC'19/PiCom'19/CBDCom'19/CyberSciTech'19)*. IEEE, 699–704.

[24] Frank Nielsen. 2019. On a generalization of the jensen-shannon divergence and the JS-symmetrization of distances relying on abstract means. arXiv:1904.04017. Retrieved from https://arxiv.org/abs/1904.04017.

[25] Marco Paolieri, Marco Biagi, Laura Carnevali, and Enrico Vicario. 2021. The ORIS tool: Quantitative evaluation of non-markovian systems. *IEEE Trans. Softw. Eng.* 47, 6 (June 2021), 1211–1225.
[26] Joy Rahman and Palden Lama. 2019. Predicting the end-to-end tail latency of containerized microservices in the cloud. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E'19)*. IEEE, 200–210.
[27] Andreas Rogge-Solti and Mathias Weske. 2015. Prediction of business process durations using non-markovian stochastic petri nets. *Inf. Syst.* 54 (2015), 1–14.
[28] Gerardo Rubino and Bruno Tuffin. 2009. *Rare Event Simulation using Monte Carlo Methods*. John Wiley & Sons.
[29] Nick Russell, Arthur H. M. Ter Hofstede, Wil M. P. Van Der Aalst, and Nataliya Mulyar. 2006. *Workflow Control-flow Patterns: A Revised View*. BPM Center Report BPM-06-22, BPMcenter. Org (2006), 06–22.
[30] Luigi Sassoli and Enrico Vicario. 2007. Close form derivation of state-density functions over DBM domains in the analysis of non-Markovian models. In *Proceedings of the International Conference on Quantitative Evaluation of Systems*. IEEE, 59–68.
[31] SIRIO Library. 2022. Retrieved from https://github.com/oris-tool/sirio.
[32] Kishor S. Trivedi and Robin Sahner. 2009. SHARPE at the age of twenty two. *SIGMETRICS Perform. Eval. Rev.* 36, 4 (March 2009), 52–57. https://doi.org/10.1145/1530873.1530884
[33] Wil M. P. Van der Aalst. 1998. The application of Petri nets to workflow management. *J. Circ. Syst. Comput.* 8, 01 (1998), 21–66.
[34] Erwin Van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, and Simon Eismann. 2018. A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures. In *Companion of the ACM/SPEC International Conference on Performance Engineering*. 21–24.
[35] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. 2009. The refined process structure tree. *Data Knowl. Eng.* 68, 9 (2009), 793–818. https://doi.org/10.1016/j.datak.2009.02.015
[36] Enrico Vicario. 2001. Static analysis and dynamic steering of time-dependent systems. *IEEE Trans. Softw. Eng.* 27, 8 (Aug. 2001), 728–748. https://doi.org/10.1109/32.940727
[37] Enrico Vicario, Luigi Sassoli, and Laura Carnevali. 2009. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. Softw. Eng.* 35, 5 (2009), 703–719.
[38] Manuel Villén-Altamirano and José Villén-Altamirano. 2011. The rare event simulation method RESTART: Efficiency analysis and guidelines for its application. In *Network Performance Engineering*. Springer, 509–547.
[39] Yilei Zhang, Zibin Zheng, and Michael R. Lyu. 2011. WSPred: A time-aware personalized QoS prediction framework for web services. In *IEEE International Symposium on Software Reliability Engineering*. IEEE, 210–219.
[40] Zheng Zheng, Kishor S. Trivedi, Kun Qiu, and Ruofan Xia. 2015. Semi-Markov models of composite web services for their performance, reliability and bottlenecks. *IEEE Trans. Serv. Comput.* 10, 3 (2015), 448–460.

Received 1 February 2022; revised 11 September 2022; accepted 23 March 2023