UNIVERSITÀ DEGLI STUDI FIRENZE

PhD Program in Smart Computing
Dipartimento di Ingegneria dell'Informazione (DINFO)

A compositional approach for quantitative evaluation of stochastic workflows

Riccardo Reali

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

A compositional approach for quantitative evaluation of stochastic workflows

Riccardo Reali

XXXV ciclo — March 2023

Ai miei genitori Graziella e Roberto
Ai miei fratelli Tommaso e Ludovico
Ai miei nonni Vezio, Rita, Armanda ed Enrico
A Maria Chiara

Acknowledgments

This thesis work is the result of synergies with other people who supported me during the last three years, providing technical, scientific and emotional support.

I would like to thank my advisor Prof. Enrico Vicario for guiding me through this journey, providing me with the knowledge and the means to learn my way through the world of research, without ever doubting my potential and always giving me his trust. I would like to thank Prof. Laura Carnevali, for providing me with scientific, technical and, more importantly, emotional support in doing this complicated but rewarding work and for instilling in me her dedication in doing research. I would like to thank Prof. Marco Paolieri, for providing support in the research work conducted on the compositional quantitative analysis method and in the implementation of the Eulero library. I would like to thank PhD. Benedetta Picano, for providing support in the research work conducted on the auctions and matching game approach for the service selection problem. I would like to thank Professor Paola Cappanera and Doctors Leonardo Scommegna, Federico Tammaro and Leonardo Paroli, for contributing to some of the research works carried out during these three years. I would like to thank all the Software Technologies Laboratory fellows for making the work environment a stimulating crossroads of knowledge and ideas. I would like to thank Professors András Horváth, Giuliano Casale, Andrea Vandin, Kishor Trivedi and Katinka Wolter for the time spent to take part in the supervision of this thesis work. I would like to thank Simona Altamura and Antonella Gullotto, for always providing administrative support with speed and meticulous attention to details. I would like to thank Regione Toscana that supported this thesis work, as part of the Pegaso program.

I would like to thank my mother Graziella, my father Roberto and my brothers Tommaso and Ludovico, for supporting me in the most difficult moments without ever ceasing to believe in me. I would like to thank Maria Chiara, for being my first supporter and my point of reference, the shoulder to cry on and the smile to laugh with. I would like to thank the fellows from *Coro Giovanile Effetti Sonori*, *Coro Giovanile Toscano* and *Compagnia degli Irriducibili*, for joining me on this journey, gifting me moments of music, drama, enjoyment and sincere friendship, which are always good for both body and soul. I would like to thank my friends *Andrei*, for teaching me how enduring friendship can be, even when we lose contact of each other. Finally, I would like to thank all the other people that took part of this journey, helping me to reach this success.

Abstract

Workflows describe processes of concurrent activities orchestrated by precedence constraints and control-flow constructs and are successfully applied to a large variety of material and digital processes from multiple contexts (e.g supply chain management, composite web services, cloud 'functions as a service'. For people employed in this contexts, scientific methods are relevant to enhance the knowledge of the active processes, and to determine what are the better decisions to take to maximize the productivity or to minimize the loss of the considered business. From this perspective, the prediction of the completion time of a workflow is crucial to design and plan the activities of a business.

In this thesis, we evaluate a stochastic upper bound on the completion time Probability Density Function (PDF) of complex workflows through an efficient and accurate compositional approach. Workflows are specified as structure trees, a hierarchical representation that is based on Stochastic Time Petri Nets (STPNs), that permits decomposition into a hierarchy of subworkflows with positively correlated response times, guaranteeing that a stochastically larger completion 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. The method is implemented in Eulero, a novel Java library that enables both modeling of complex workflows through the structure tree and evaluation of their completion time PDF.

Predicting the completion time of a workflow can be exploited in an environment of competition, where a set of resource-constrained aggregators selects service providers for the implementation of workflows requested by a set of customers, managing the Service Level Agreement (SLA) on both the sides. In particular, each aggregator selects the implementation of elementary services through a Vickrey-Clarke-Groves-based auction game, where each provider bids Cumulative Distribution Function (CDF) of the offered service completion time. On awarding of the auction, the aggregator can safely predict end-to-end completion times that can guarantee to each customer, and the consequent reward that can obtain. The resource-constrained assignment problem between customers and aggregators is thus solved through a matching game with externalities and incomplete information that aims at maximizing efficient usage of resources in the system according to a collective utility function. The impact of possible cheating strategies to improve the utility of some players is also analyzed, together with the stability of the formulated matching game.

Then, the presented methods are then demonstrated in a Model-Driven Engineering (MDE) approach that implements Model-to-Model transformations to map cases of textile manufacturing district production to the proposed scientific application domain, enabling to perform of the methods to real-world contexts.

Contents

Chapter 1

Introduction

Workflow models describe processes of concurrent activities that are 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 [1]. This abstraction has been successfully applied to a large variety of material and digital processes from multiple contexts, including supply chain management [2], administration [3, 4], composite web services [5], cloud "functions as a service" [6].

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 [7, 8, 9, 10, 11]. This approach is particularly useful for the analysis of Service Level Agreements (SLAs) with *soft deadlines* and *penalty functions* [12, 13], 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 PDF over $[T_{\max}, +\infty)$; the expected penalty can be evaluated by multiplying the PDF by the penalty function and integrating over all response times. 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 an integral that is an upper bound of the integral 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.

At the same time, exact analytical and numerical methods cannot be applied to large workflows because of two main sources of computational complexity. In fact, 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 [14, 15] of workflow execution, but which can be critical to fit data collected by web applications. In principle, if the workflow model never reaches a state where two GEN activities overlap their durations, satisfying the *enabling restriction*, then transient analysis can still resort to numerical solution techniques [16, 17, 18]. Also, multiple concurrent GEN durations can be managed by approximation through Continuous PHase type (CPH) distributions [19, 20], also with the support of various tools [21, 22, 23]. In other cases, renewal theory can be exploited to evaluate the behavior of a workflow over time. In this case, the evaluation exploits *regenerative states*, i.e. states where the Markov property holds again, by evaluating a global and a local kernels and employing them in a set of equations of Volterra [24, 25, 26]. However, more often the interleaving of activities in concurrent subworkflows leads to break the enabling restriction, to significantly reduce the number of regeneration states, to get a state-space explosion and to complex stochastic dependencies due to their overlapping GEN durations [27], which disable the mentioned solutions.

This sources of complexity lead to the necessity of implementing compositional analysis approaches. A compositional approach can address both issues by avoiding the explicit representation of interleavings, limiting state space explosion and simplifying the stochastic processes of individual components [27]. In [28], mean and standard deviation of workflow response time are derived through an efficient bottom-up calculus when activities with GEN duration are composed by fork/join, sequence, and repetition; the approach is extended in [29] for the special case of Continuous Phase (CPH) durations. In [30], the completion time of an acyclic attack tree with CPH delays is evaluated by repeatedly composing CPH distributions in a bottom-up fashion, with possible approximation to limit the number of phases in their representation. In [31], the response time PDF of complex workflows with GEN durations is evaluated compositionally, by repeatedly applying Markov regenerative analysis to nested subworkflows (with limited concurrency), and then composing the resulting distributions bottom-up; the approach is extended in [32] to repetition blocks and unbalanced split/join constructs. Stochastic order is guaranteed by the class of approximating distributions and by the method used to simplify dependencies within the workflow.

In this thesis work, an efficient and accurate compositional technique to evaluate a stochastic upper bound on the response time PDF of complex workflows is illustrated. 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) [33], that permits the workflow decomposition into a hierarchy of subworkflows with positively correlated response times. In so doing, a formalization for the approximate derivation of conservative measures that estimate the complexity of evaluation of a subworkflow is provided, driving the workflow decomposition into subworkflows analyzed in isolation; a stochastically ordered approximant for the response time PDF of subworkflows is defined by combining shifted truncated Exponential (EXP) terms, each having positive or negative rate depending on the concavity of the response time CDF; finally, an extensive experimentation on manually and randomly generated models with increasing complexity is conducted, to validate the proposed technique and to investigate which decomposition heuristics work better and under which conditions.

The work conducted on the described compositional method leads to the realization of a Java library called *Eulero*. The library is designed to implement the main features of the compositional method. In particular, the library implements packages to model a workflow, perform analysis heuristics, and approximate probability distributions. Each of these packages includes a generic interface, which is designed with the intent of facilitating the extension and the addition of features (e.g., new analysis heuristics, new approximations). Finally, the library provides a package that implements a random workflow generator, which can be controlled to generate benchmarks of models, through which validating new quantitative evaluation techniques.

The evaluation of the response time of a workflow results to be a useful tool for a large number of application contexts [2, 3, 4, 5, 6], but it calls attention to the problem of how to effectively determine the stochastic parameters that characterize the workflow activity durations. One solution consists in characterizing them so as to fit statistics provided by individuals who are experts in the field, or data collected from software architecture logs in such a way to maximize the likelihood of an exponomial [34, 35] distribution with respect to the data. In competitive environments, these values can vary considerably among different parties. In fact, competitors can have different resource availability or different production capabilities, and these factors can significantly change the completion time of the entire workflow. However, this additional complexity factor enables a game-theoretical mechanism through which to decide to whom to delegate the execution of a task, and simultaneously determine what is the stochastic duration of the delegated task.

An example of the above comes from the area of Software as a Service (SaaS). In the model of Software as a Service (SaaS), software is supplied over the Internet, exploiting Service-Oriented Architecture (SOA) principles [36]. This enables delivery of self-contained and well-defined modules implementing different functionalities, that are independent of the context or state of other services, and that can be aggregated into composite services. In this scheme, services are granted in conformity with Service Level Agreements (SLAs) established between customers and providers [37, 38] so as to characterize required and actual Quality of Service (QoS) through Service Level Objectives (SLOs) and Service Level Indicators (SLIs), respectively. In turn, both SLOs and SLIs are then associated with specific thresholds and metrics, such as hard or soft deadlines and service time. While the SLO represents an individual promise made between provider and customer in the SLA, the SLI is the actual measurement of the productive uptime, i.e., it measures the compliance with the SLO.

Actual and effective deployment of the SaaS paradigm requires investigation of various aspects [37, 38, 39, 40]. In particular, service selection plays a key role to enable dynamic and flexible business models where different implementations of the same service can attain different objectives of quality. In cloud market, the service selection problem identifies the decision-making strategy through which cloud services and providers are matched together. In fact, the service selection scheme drastically impacts the customer experience and the provider revenue, aiming at properly assigning cloud services to the providers, to fulfill service requests maximizing both customer satisfaction and provider utility [37, 38, 39, 40]. Whereas the provider utility is mostly price driven, customer utility may involve other interesting qualities such as reliability in supplied services, or end-to-end (e2e) time, defined as the time elapsed between the buyer request submission and the workflow completion [41].

In this picture, service selection plays an important role in many contexts, since it defines the decision-making strategy through which activities and providers are matched together. In fact, the service selection scheme drastically impacts the customer experience and the provider revenue, aiming at properly assigning activities to the providers, to fulfill service requests maximizing both customer satisfaction and provider utility [37, 38, 39, 40]. Whereas the provider utility is mostly price driven, customer utility may involve other interesting quantities as reliability in supplied services, or end-to-end (e2e) time, defined as the time elapsed between the buyer request submission and the workflow completion [41]. Due to the heterogeneity of the quantities of interests involved in the provider-service assignment, interchangeably referred as the service selection, the corresponding decision-making process is conventionally identified by a bargaining procedure, where cloud services are offered to a set of bidders, i.e., the providers, and then sold to the winner [37, 38, 39, 40]. Mathematical tools of game theory permit modeling of complex interactions among interdependent rational players and their behavior expressed as choices over strategies [42, 43, 44]. In particular, auction theory focuses on game-theoretic properties of auction markets, along with the behavior of agents involved.

Auction theory provides a market mechanism for selecting services, whose essence is a game, where players are the bidders, strategies are the bids, and both allocations and payments are functions of the bids [42, 43, 44].

In this thesis work, a case of a hierarchical infrastructure where the association among customers and service providers is intermediated by aggregators is addressed. Customers demand to resource-constrained aggregators for workflow execution, exhibiting soft deadline SLOs over the workflow e2e times, expressed in terms of Cumulative Distribution Functions (CDFs). In turn, aggregators offer service outsourcing to a set of antagonist providers competing for service delivery, whose bids are represented by CDFs of service completion times. The designed stochastic framework applies matching theory to associate customers and aggregators, and a nested Vickrey-Clarke-Groves (VCG) auction to perform service selection. Experimentation investigates on the maximization of the total welfare, represented by the optimization of utilities for all of the parties involved in the scheme, i.e., aggregators and providers. Results confirm the robustness of the proposed matching approach in terms of the Price of Anarchy (PoA) [45, 46] paid to take decisions that rely on partial information metrics. The impact of delays on service execution has been analyzed, as well as the behaviour of both the aggregator and provider utility, under different assumptions about the competitiveness among providers.

Finally, the described approaches are applied to a concrete application example from the context of manufacturing districts, showing a Model-Driven Engineering (MDE) approach. MDE considers models as primary artifacts during all software development phases [47] and connects them with the practice of software engineering, making them living components of the system rather than using them for documentation and study purposes only. Some of the main parts of MDE are Domain-Specific Languages (DSLs) and Domain-Specific Modeling Languages (DSMLs), specialized in formalizing the structure and behavior of applications, and described using meta-models to map relationships, semantics, and constraints between concepts expressed in a domain [48]. This practice is also common in industrial contexts, where small DSLs are developed for narrow and well-understood domains [49]. Another important aspect that led to the widespread use of MDE and Model-Driven Design (MDD) are automated transformations: Model-to-Text (M2T) transformations are more commonly used to transform a particular model instance into text-based file formats or software artifacts available as source code (code generation), while Model-to-Model (M2M) transformations are used to translate a model into another model. Both techniques are referred to as "correct-by-construction" [48] given that they do not require any subsequent modification and they avoid manual, and thus error-prone, changes to the considered artifacts. In this thesis work, a metamodel of a manufacturing district is presented, and some M2M transformations are designed to map manufacturing district domains to the scientific application domains related to the presented researched methods. The realized mapping enables to perform compositional analysis and the proposed VCG auction-based service provider selection method to a real application case, demonstrating how the discussed topics have not only scientific relevance, but also impact in multiple application areas.

The thesis is organized as follows: in Chapter 2 a heuristics compositional method for the evaluation of complex workflows is presented, including a formalism for hierarchical workflow representation; in Chapter 3 it is illustrated the Eulero library, which is a Java library that implements the modeling capabilities and compositional method described in Chapter 2, enabling a headless evaluation of complex workflows; in Chapter 4 a combined matching and auction theory approach for service selection is presented; in Chapter 5 the described methods are applied on the specific case of a textile manufacturing district, following a MDE approach; finally, in Chapter 6, conclusions are summarized and some directions for future works are provided.

Chapter 2

A compositional approach for complex workflow evaluation

In this chapter, a compositional technique for efficient and accurate evaluation of the response time PDF of complex workflows is illustrated. Workflows are specified through a hierarchical structured formalism defined in terms of Stochastic Time Petri Nets (STPNs), which combines activities with GEN duration and bounded support through sequence, choice/merge, and split/join blocks, with *unbalanced split and join constructs that may break the structure of well-formed nesting.* The structured specification of the workflow is exploited to decompose the model into a hierarchy of subworkflows with positively correlated response times. Subworkflow are efficiently analyzed in isolation, and results are recombined in order to provide a stochastic upper bound of the CDF end-to-end response time of the workflow. Experiments show that the approach is efficient and accurate, and scales on high-complexity workflows workflows, significantly outperforming simulation with the same computation time.

The chapter is organized as follows. Section 2.1 presents the structured workflow model and recalls forward transient analysis and numerical analysis. Section 2.2 introduces a method to estimate the complexity of forward transient analysis of an STPN block based the results of nondeterministic analysis of the underlying TPN. Section 2.3 illustrates the proposed approach to explore the structure tree and decompose the workflow into a hierarchy of analyzable subworkflows. Section 2.4 presents experimental results.

2.1 Workflow modeling

In this section, stochastic workflows through a class of STPNs (Section 2.1.1) that supports the derivation of a structured representation and guarantees positive correlation among the response times of different subworkflows (Section 2.1.1) are specified. Then, it is derived the response time PDF of a workflow either by bottom-up fully numerical analysis, if the workflow consists of independent subworkflows composed in well-nested structures, or by forward transient analysis, if the workflow includes dependencies among subworkflows composed in not well-nested structures (Section 2.1.3).

2.1.1 Stochastic Time Petri Nets

In this subsection, syntax and semantics of Stochastic Time Petri Nets (STPNs) are recalled (Section 2.1.1).

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 associates 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 $x > LFT(t)$; 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.

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 *immediate* (IMM) if $EFT(t) = LFT(t) = 0$ and *timed* otherwise; a timed transition t is termed *exponential* (EXP) if $F_t(x) = 1 - \exp(-\lambda x)$ for some rate $\lambda \in \mathbb{R}_{>0}$, and *general* (GEN) otherwise. For each GEN transition t, it is assumed that F_t is 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 the Dirac impulse function $f_t(y) = \delta(y - \overline{y})$ as generalized PDF, with $\overline{y} = EFT(t) = LFT(t)$.

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. A transition is *firable* in a state if its time-to-fire is equal to zero. If multiple transitions are firable in a state $s = \langle m, \tau \rangle$, 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. When transition t fires, state 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').

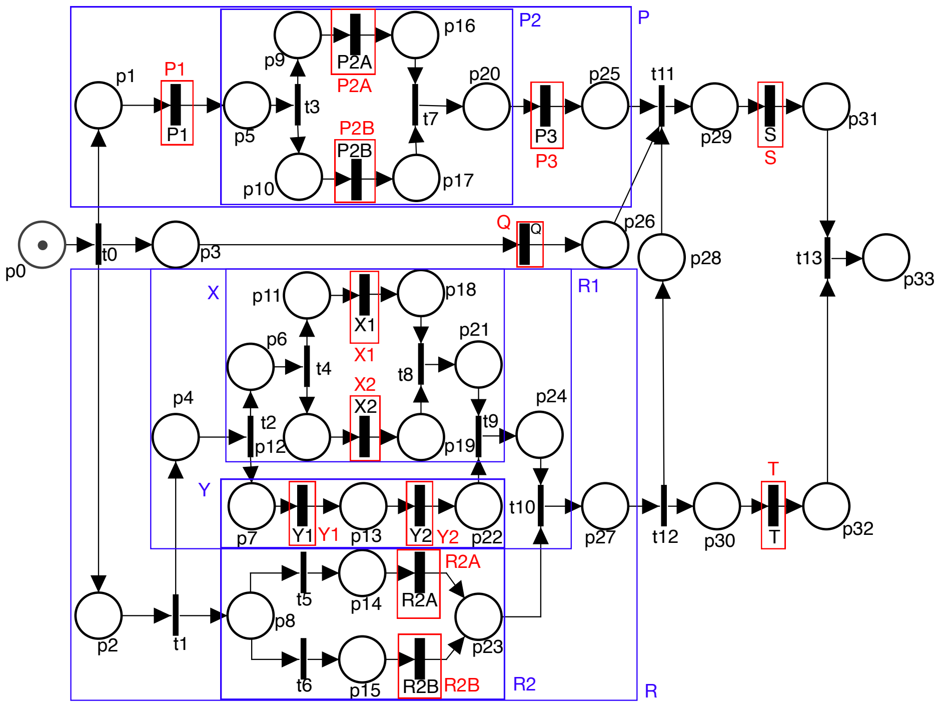


Figure 2.1: STPN model of a workflow: blocks are highlighted by boxes, blue for composite blocks and red for activity blocks; all transitions have uniform PDF over $[0, 1]$ (firing intervals and PDF types are not shown to reduce the cluttering).

2.1.2 Workflow model

This subsection illustrates a hierarchical representation to model workflows as structure trees, enabling efficient compositional analysis. An example of the structure tree is provided in Fig. 2.2. Here, blocks are composed by recursion to provide a different perspective of the workflow, enabling to catch insights about its complexity, and to perform efficient analysis of its response time. For example, block `TOP` contains 5 blocks `P`, `Q`, `R`, `S`, `T`. Each of them contains other blocks (e.g. `P` contains `P1`,

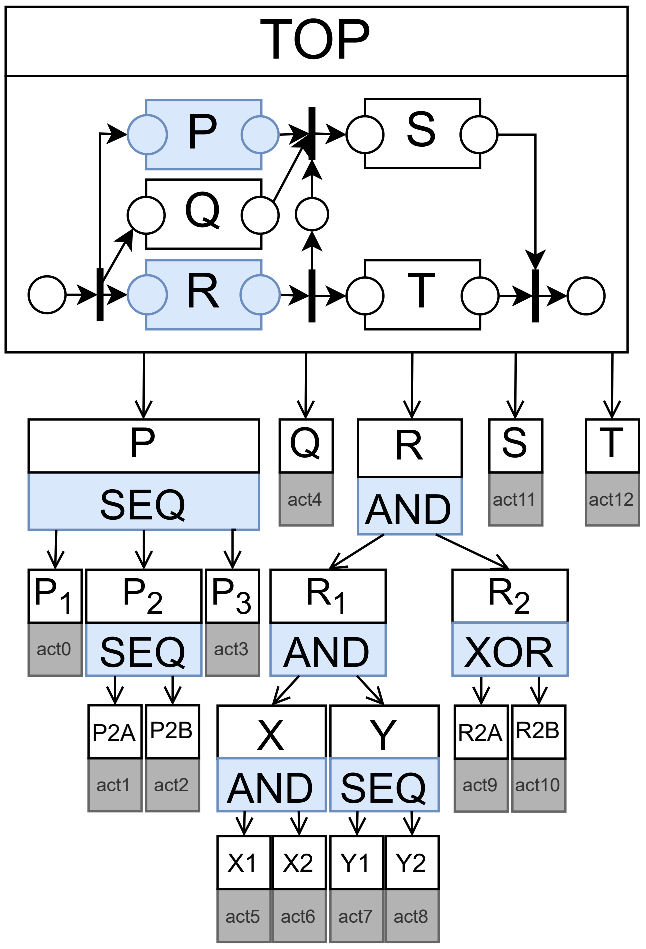


Figure 2.2: Structure tree of the workflow of Fig. 2.1: composite blocks are filled with blue.

P2, P3), and the recursion is repeated until the leaves of the tree, which represent the activities of the workflow. Then, each block can be mapped to a specific STPN, whose form depends on the type of the considered block type (SEQ, AND, XOR, DAG) and combines the STPN representations of the children of the block (e.g. the STPN related to block P, consists in implementing an STPN sequence of the STPNs related to blocks P1, P2 and P3). This formalism enables to hide the complexity of a flat STPN (as the one of Fig. 2.1), by providing a hierarchical representation that enables to catch more insights (Fig. 2.2).

STPN blocks

To support efficient analysis of stochastic workflows, a class of STPNs that is sufficient to represent a variety of workflow control patterns [1, 50] is considered, making explicit their structure of composition. Specifically, 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 concurrent behaviour (split/join) and sequential behaviour (sequence, choice/merge), and through acyclic constructs breaking well-formed nesting by means of unbalanced fork and join operations (simple split, simple join). In particular, the following types of block are considered:

- An elementary *activity* is represented by an STPN with a single transition with GEN duration connecting the initial and final places (e.g., block `S` in Figs. 2.1 and 2.2). GEN transitions have exponomial [35] PDFs. An exponomial PDF is 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 semi-symbolic representation over the entire domain or piecewise-defined over multiple subdomains. Note that exponomial PDFs enable for the representation of defective distributions [51], which can be used to model failures in the activities of a workflow [52, 53]. In particular, the mass at infinity of the defective distribution determines the probability of the occurrence of a failure. Moreover, defective distributions could be exploited in conjunction with loop patterns to implement recovery mechanisms, to circunvent the occurence of a failure.

- SEQ$\{$Block$_1, \ldots,$ Block$_n\}$ is a *sequence* of n blocks Block$_1$, ..., Block$_n$ (e.g., block `Y` in Figs. 2.1 and 2.2).

- XOR$\{$Block$_1, \ldots,$ 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 Block$_1$,...,Block$_n$ having probabilities $p_1, \ldots, p_n$, respectively, which, in turn, are connected to the final join place of the block (e.g., block `R`$_2$ in Figs. 2.1 and 2.2; random switch of transitions `R2A` and `R2B` is built by IMM transitions `t5` and `t6` which embeds the probability $p_{\texttt{R2A}}$ and $p_{\texttt{R2B}}$, respectively; then `R2A` and `R2B` are merged into the join place `p23`.). An XOR block is *balanced*, i.e., all the alternative paths started at the initial split are terminated at the final join place.

- AND$\{$Block$_1, \ldots,$ Block$_n\}$ is a balanced split-join made of an initial IMM *parallel split* transition that forks execution along n concurrent blocks Block$_1$, ..., Block$_n$ and a final IMM *synchronization* transition that terminates the block (e.g., `X` in Fig. 2.1; here, transition *mathttt*4 is the *split*, and transition *mathttt*8 is the synchronization). An AND block is *balanced*, i.e., all the concurrent paths started at the initial split are terminated at the final synchronization.

- DAG$\{$Block$_1, \ldots,$ Block$_n\}$ is the composition of n blocks Block$_1, \ldots,$ Block$_n$ in a Directed Acyclic Graph (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) [1]. 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 Fig. 2.1, 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`).

Different priorities are assigned 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. Note that IMM transitions of different blocks may be enabled at the same time, requiring state space analysis to enumerate all the possible firing sequences. Moreover, if the IMM transitions of an XOR block are concurrently enabled with the IMM transitions of other blocks, the probability to execute each block $\textsc{Block}_i$ becomes lower than p_i. To overcome these issues, the IMM transitions of different blocks and the IMM transitions of each DAG block have different priorities.

According to this definition, each workflow model specified as a composition of STPN blocks can be translated into a unique STPN. For example, the model defined specified as in Fig. 2.2 can be mapped to the STPN illustrated in Fig. 2.1. In particular, in the latter blue and red boxes respectively show the simple and the composite activities that are specified in the workflow shown in the former. 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, a workflow model cannot represent a race selection 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.

Note that due to the hierarchical representation, it would be possible to extend the block types that can be represented. In fact, maintaining a mapping with an STPN having single start and end places, which guarantee that the block execution ends w.p.1, the recursive composition of blocks would remain unaltered. This would enables the inclusion of block types representing different stochastic behaviors such as, inclusive OR, the K-out-N pattern, or even loops and cycles. In particular, by removing the assumption that every activity ends w.p.1, i.e., by enabling *failures* on activities, loops could be used to represent a form of *recovery* to be launched when failure occurs. Inclusion of loop patterns in the proposed formalism is preliminarily discussed in [32].

Structure tree

Blocks combined as depicted in Section 2.1.2 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). Fig. 2.2 shows the structure tree of the workflow STPN of Fig. 2.1. 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, AND, XOR, 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* [54] and *process structure trees* [55]. Similarly to these works, it is ensured 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).

2.1.3 Stochastic analysis of a workflow block

This subsection describes how to perform the stochastic evaluation of the response time of a block of a workflow. When a block is well-nested the evaluation can be solved by performing a fully numerical bottom-up evaluation. For example, to evaluate response time of block `P` in Fig. 2.2, the analytic-numerical response time of `P`, `P` and `P` (whose is in turn evaluated from the response time of `P` an `P`); then results enables to evaluate the response time of P, exploiting a fully numerical formula. In case, a block is not well-nested (e.g. block `TOP`), to cope with unbalanced dependencies, the forward transient analysis is exploited.

Fully numerical analysis

For workflows consisting of SEQ, AND, and XOR blocks, the fully analytic-numerical form of the response time PDF can be derived by bottom-up composition of the response time Cumulative Distribution Functions (CDFs) of the blocks, due to the fact that SEQ, AND, and XOR operators compose independent subworkflows in well-nested structures. Specifically, given n blocks $b_1, \ldots, b_n$ with response time PDFs $\phi_1(t), \ldots, \phi_n(t)$ and response time CDFs $\Phi_1(t), \ldots, \Phi_n(t)$, 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 $\phi_1(t), \ldots, \phi_n(t)$; $\forall\, t \in [0, t_{\max}]$:

$$\begin{aligned} \Phi_{\text{seq}}(t) &= \int_0^t \phi_1(t) \circledast \phi_2(t) \circledast \ldots \circledast \phi_n(t) dt \\ \phi_i(t) \circledast \phi_j(t) &= \int_0^t \phi_i(\tau)\phi_j(t-\tau)d\tau \end{aligned} \tag{2.1}$$

- 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; $\forall\, t \in [0, t_{\max}]$:

$$\Phi_{\text{xor}}(t) = p_1\, \Phi_1(t) + \ldots + p_n\, \Phi_n(t) \tag{2.2}$$

- 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:

$$\Phi_{\text{and}}(t) = \Phi_1(t) \cdot \ldots \cdot \Phi_n(t) \tag{2.3}$$

Then, the response time PDF ϕ_{seq}, ϕ_{and}, and ϕ_{xor} of a SEQ, an AND, and an XOR block, respectively, can be obtained as the derivative of the response time CDF of the block, e.g., $\phi_{\text{and}}(t) = d/dt\, \Phi_{\text{and}}(t)$.

For instance, the response time PDF of block R in Fig. 2.2 is $\phi_{\texttt{R}}(t) = d/dt\, (\Phi_{\texttt{R1}}(t) \cdot \Phi_{\texttt{R2}}(t))$, where: $\Phi_{\texttt{R1}}(t) = d/dt\, (\Phi_{\texttt{X}}(t) \cdot \Phi_{\texttt{Y}}(t))$ is the response time CDF of AND block R1; $\Phi_{\texttt{R2}}(t) = p_{\texttt{R2A}}\Phi_{\texttt{R2A}}(t) + p_{\texttt{R2B}}\Phi_{\texttt{R2B}}(t)$ is the response time CDF of XOR block R2; $\Phi_{\texttt{X}}(t) = d/dt\, (\Phi_{\texttt{X}_1}(t) \cdot \Phi_{\texttt{X}_2}(t))$ is the response time CDF of AND block X; $\Phi_{\texttt{Y}}(t) = \int_0^t \int_0^\tau \phi_{\texttt{Y1}}(x)\, \phi_{\texttt{Y2}}(\tau - x)\, dx\, d\tau$ is the response time CDF of SEQ block Y; finally, $\Phi_{\texttt{X}_1}(t)$, $\Phi_{\texttt{X}_2}(t)$, $\Phi_{\texttt{Y}_1}(t)$, $\Phi_{\texttt{Y}_1}(t)$, $\Phi_{\texttt{R2A}}(t)$, and $\Phi_{\texttt{R2B}}(t)$ are the response time CDFs of activity blocks $\texttt{X}_1$, $\texttt{X}_2$, $\texttt{Y}_1$, $\texttt{Y}_2$, R2A, and R2B, respectively.

Forward transient analysis

For workflows including DAG blocks, dependencies among subworkflows composed in not well-nested structures prevent derivation of the response time CDF by bottom-up fully numerical analysis, requiring evaluation of the marking process of the workflow STPN. Due to concurrent activities with GEN duration, the marking process typically reaches a limited number of regeneration points, i.e., time instants at which the Markov property is satisfied and future behaviour depends only on the current marking. Therefore, evaluation can be efficiently performed by forward transient analysis based on the method of stochastic state classes [25], which evaluates the transient probability of each marking without restrictions on the presence of regenerations.

Specifically, a *stochastic state class* $\Sigma = \langle m, D, f \rangle$ encodes a marking m, a *Difference Bounds Matrix* (*DBM*) *zone* D [56], i.e., a joint support for the times-to-fire of the enabled transitions and for the elapsed time, and a joint PDF f for such values. For each transition t that can fire first with probability μ, a succession relation $(\Sigma, t, \mu, \Sigma')$ is enumerated from Σ to the stochastic state class $\Sigma' = \langle m', D', f' \rangle$ after the firing, with new marking m', joint domain D' of the elapsed time and reachable times-to-fire, and their joint PDF f'. The analysis enumerates the tree of stochastic state classes reached within a given time limit $t_{\max}$, using the probability that each class is the last node reached within time $t \in [0, t_{\max}]$ to derive the probabilities of all markings at time t. Then, the workflow response time CDF can be derived as the transient probability of the absorbing marking assigning one token to the final place of the STPN, and the PDF as the derivative of the CDF. The SIRIO library [57] of the ORIS tool [58] provides a closed-form implementation of forward transient analysis provided that each transition has an expolynomial PDF. Specifically, expolynomial PDFs, also termed exponomials [35], are defined as the sum of products of exponential and polynomial terms, i.e., $f(x) = \sum_{m=1}^{M} c_m \prod_{n=0}^{N_m} x_n^{\alpha_{mn}} e^{-\lambda_{mn} x_n}$, and they may have a semi-symbolic representation over the entire domain or be piecewise-defined over multiple sub-domains.

For instance, forward transient analysis of the STPN overall workflow of Fig. 2.1 is not affordable, due to the large number of concurrently enabled GEN transitions, pointing out the need of a compositional solution for the evaluation of the workflow response time.

The complexity of forward 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. Specifically, the continuous set of executions of the STPN is encoded into a discrete representation termed *state class graph* [59, 60], 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 [60]. 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 behaviours (i.e., the number of firings between selected state classes). Given that the STPN of a workflow has a final absorbing place $\texttt{p}_{\texttt{fin}}$, all the paths in the state class graph of the underlying TPN terminate in a state class with marking $\texttt{p}_{\texttt{fin}}$ and no enabled transition.

For instance, nondeterministic analysis of the TPN underlying the workflow STPN of Fig. 2.1 enumerates 19859 state classes in non-negligible time larger than 5s, showing that an efficient compositional approach is needed also to estimate the complexity of forward transient analysis.

2.2 Workflow complexity

In this section, it is characterized how *concurrency* and *sequencing* among activities of a workflow affect the complexity of the forward transient analysis of its STPN representation (Section 2.2.1); then, it is shown how these complexity measures can be estimated on the state class graph of the underlying TPN (Section 2.2.2). The considered measures can be use to determine the complexity of a workflow. In fact, by defining some threshold values, it is possible to determine if a block is complex or not, based on the fact that thresholds are exceeded or not, respectively, as illustrated in Definition 4.

2.2.1 Complexity factors

Evaluation of DAG blocks can be performed by forward transient analysis based on the method of stochastic state classes [25]. 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 [56] 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 [57] of the ORIS tool [58] provides a closed-form implementation of the analysis provided that each transition has an exponomial PDF. Note that *regenerative transient analysis* [25] 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* [59, 60], 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 [60]. 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 Fig. 2.1 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 exponomial terms of the (monovariate) PDFs of the GEN transitions [61]. 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 exponomial 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 [61]: in fact, at each firing, the number of zones increases polynomially with the number of persistent transitions, and the number of exponomial terms increases linearly with the polynomial degree of the joint PDF. Moreover, if the analytical form of the joint PDF contains no EXP factor, 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 exponomial PDF.

2.2.2 Complexity measures

According to the analysis of Section 2.2.1, the complexity of forward transient analysis of the STPN of a workflow is estimated 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 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 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, c and q are characterized both for the TPN of the workflow and for a variant that hides the complexity of the subworkflows.

Definition 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.*

A workflow with structure tree Ω with depth D is considered, i.e., a workflow where the top-level has depth 1 and the bottom level has depth D. A bottom-up visit of Ω is performed starting from the second-to-last level, i.e., the level with depth $D - 1$. At each level, for each composite block b, *complexity tuple* $\langle C, \bar{C}, q, \bar{q} \rangle$ is derived, where C and $\bar{C}$ are upper bounds on the concurrency degree of the TPN and the unexpanded TPN of block b, respectively, and q and $\bar{q}$ are the sequencing degree of the TPN and of the unexpanded TPN of block b, respectively. For each level $d \in \{D-1, \dots, 1\}$ and each composite block b, the following operations are performed.

- A variant of the unexpanded TPN of b is derived 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, a lower bound L and an upper bound U on the duration of block b itself are derived and used by the next iteration of the procedure in order to derive a variant of the unexpanded TPNs of the composite blocks at the next higher level. Specifically:

$$L = \begin{cases} \sum_{z \in K_b} L_z & \text{if } b \text{ is SEQ, DAG} \\ \max_{z \in K_b}\{L_z\} & \text{if } b \text{ is AND} \\ \min_{z \in K_b}\{L_z\} & \text{if } b \text{ is XOR} \end{cases} \qquad U = \begin{cases} \sum_{z \in K_b} U_z & \text{if } b \text{ is SEQ, DAG} \\ \max_{z \in K_b}\{U_z\} & \text{if } b \text{ is AND, XOR} \end{cases} \tag{2.4}$$

  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.

- Nondeterministic analysis of the considered variant of the unexpanded TPN of block b is performed in order to compute C and $\bar{C}$:

$$C = \max_{S \in \Gamma} \left\{ \sum_{t \in E_S} C_t \right\} \quad \bar{C} = \max_{S \in \Gamma} \left\{ \sum_{t \in E_S} 1 \right\} \tag{2.5}$$

  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 behaviours 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.

- q and $\bar{q}$ are efficiently derived 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):

$$q = \begin{cases} \sum_{z \in K_b} q_z & \text{if } b \text{ is SEQ, AND, DAG} \\ \max_{z \in K_b}\{q_z\} & \text{if } b \text{ is XOR} \end{cases} \qquad \bar{q} = \begin{cases} \sum_{z \in K_b} 1 & \text{if } b \text{ is SEQ, AND, DAG} \\ 1 & \text{if } b \text{ is XOR} \end{cases} \tag{2.6}$$

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 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, let assume to consider thresholds $\Theta_c = 4$ and $\Theta_q = 10$ for complexity measures of *concurrency* and *sequencing*. For the workflow of Fig. 2.2, 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 with respect to the set thresholds due to the large concurrency degree ($C = 7$) and sequencing degree ($q = 13$) among GEN transitions. On the contrary, since the unexpanded concurrency and sequencing degrees (respectively, $\bar{C} = 3$ and $\bar{q} = 5$) do not exceed the set thresholds, determining that the unexpanded TPN is not complex, then it is suggested 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 $\langle C, \bar{C}, q, \bar{q} \rangle = \langle 4, 2, 5, 2 \rangle$.

2.3 Workflow evaluation

In this section, the compositional solution to evaluate the response time PDF of a workflow is illustrated (Section 2.3.1). Given some thresholds of the complexity measures, the method perform a top-down visit of the structure tree of the workflow, to estimate the complexity of each block in order to understand which blocks need to be evaluated in isolation. After such analysis, a bottom-up visit is performed to recombine the obtained results. In doing so, a stochastic upper bound for monovariate PDFs with bounded support is derived (Section 2.3.2), and a proof that the evaluated response time is a stochastic upper bound of the real response time is provided (Section 2.3.3).

2.3.1 Evaluation heuristics

The end-to-end response time is evaluated by decomposing a workflow into a hierarchy of subworkflows and by composing the results of their separate analyses, repeatedly applying numerical analysis (Section 2.1.3) and forward transient analysis (Section 2.1.3) 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). On the other hand, 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.

The structure tree is exploited 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 a top-down visit of the structure tree is performed to decompose the workflow into subworkflows, and a bottom-up visit is performed to compose the results of their separate analyses. Specifically, at each step of the top-down visit, given the concurrency and sequencing thresholds Θ_c and Θ_q, respectively, the following operations are performed to derive a stochastic upper bound $\phi(t)$ on the response time PDF of the current block b:

- If b is an activity block, then its exact response time PDF is its duration PDF (lines 1–2).
- 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 and its complexity measures exceed the thresolds Θ_c and Θ_q, then one of two different heuristics can be recursively applied to reduce the block complexity until its forward transient analysis becomes affordable (lines 12–13).
    - Algorithm 2 shows the *Split Dependencies First* (SDF) heuristics: if b is internally complex to analyze, it is split into the AND of two decoupled

subworkflows by replicating the common nodes of the two subworkflows (*inner block replication*, lines 1–2); if b is complex to analyze, some block is analyzed in isolation (*inner block analysis*, lines 3–4); 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 1–2) is applied before inner block replication (lines 3–4).

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

```
CompositionalAnalysis($b$, $\Theta_c$, $\Theta_q$, $h$)
    input : block $b$, concurrency degree threshold $\Theta_c$, sequencing degree
            threshold $\Theta_q$, heuristics h
    output: response time PDF $\phi(t)$ 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           $\phi_i(t) \leftarrow$ CompositionalAnalysis($b_i$, $\Theta_c$, $\Theta_q$, h)
6       if $b$ is a SEQ block then
7           return $\phi(t) = \frac{d}{dt}\int_0^t \phi_1(t) \circledast \phi_2(t) \circledast ... \circledast \phi_n(t) dt$
8       if $b$ is an XOR block then
9           return $\phi(t) = p_1\,\phi_1(t) + \ldots + p_n\,\phi_n(t)$
10      if $b$ is an AND block then
11          return $\phi(t) = \frac{d}{dt}\left[\int_0^t \phi_1(t)dt \cdot \ldots \cdot \int_0^t \phi_n(t)dt\right]$
12  if $b$ is a DAG block then
13      return h($b$ , $\Theta_c$, $\Theta_q$)
```

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

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

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) in order to evaluate the 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 1–2).

  Specifically, let $G = (V, E, v_I, v_F)$ be the DAG (e.g., the top-level DAG of Fig. 2.2) 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 Fig. 2.2), 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 Fig. 2.2). 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., $K = \{$`R`$\}$). The vertices in K and the edges to/from vertices in K are replicated (i.e., R' is added

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

```
ReplaceBlockFirst($b$, $\Theta_c$, $\Theta_q$)
    input : workflow block $b$, concurrency degree threshold $\Theta_c$, sequencing
            degree threshold $\Theta_q$
    output: response time PDF $\phi(t)$ of $b$
  1 if $b$ contains at least one composite block and is complex to analyze then
  2   | return InnerBlockAnalysis($b$ , $\Theta_c$, $\Theta_q$)
  3 if $b$ is internally complex to analyze then
  4   | return InnerBlockReplication($b$ , $\Theta_c$, $\Theta_q$)
  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($b$, $\Theta_c$, $\Theta_q$)
    input : workflow block $b$
    output: response time PDF $\phi(t)$ of $b$
  1 order the predecessors of the final block of $b$ by the values of $C$ and $q$
  2 $v \leftarrow$ predecessor of the final block of $b$ with max value of $C$ (and, in case
      of tie, with max value of $q$)
  3 $b' \leftarrow b$ after replicating the predecessors of block $v$
  4 return CompositionalAnalysis( $b'$, $\Theta_c$, $\Theta_q$)
```

to V; $v_I \to R'$ and $R' \to T$ are added to E; $R \to T$ 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 Fig. 2.3a, after replication of the predecessor `R` of blocks `S` and `T`, the DAG of Fig. 2.2 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.

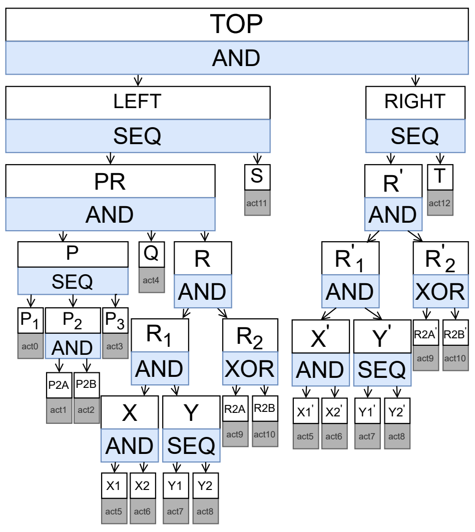
- As illustrated by Algorithm 5, performing inner block analysis on a DAG block b consists in: performing compositional analysis of a composite block v contained within b (line 3), which yields a stochastic upper bound PDF $\phi(t)$ on the response time PDF of v; deriving a stochastic upper bound PDF $\hat{\phi}(t)$ on $\phi(t)$ by Lemma 1 (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 1–2).

  For instance, as shown in Fig. 2.3b, compositional evaluation of the example of Fig. 2.2 through the RBF heuristics consists in analyzing blocks `P` and `R` in isolation and then approximating their response time PDF. Then, the model can be solved by forward transient analysis.

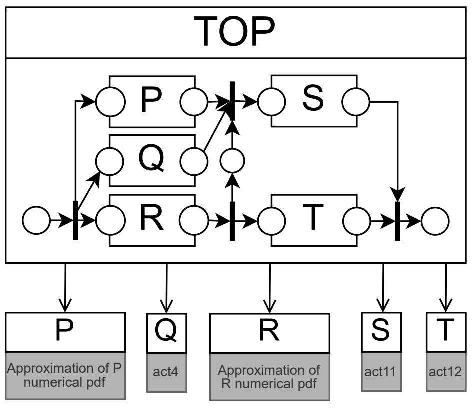
The SDF heuristics is more accurate than the RBF heuristics if the correlation between the response times of the replicated nodes and the response time of the overall workflow is low. Conversely, the RBF heuristics is more accurate than the SDF heuristics if the correlation between the response times of the nodes replicated

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

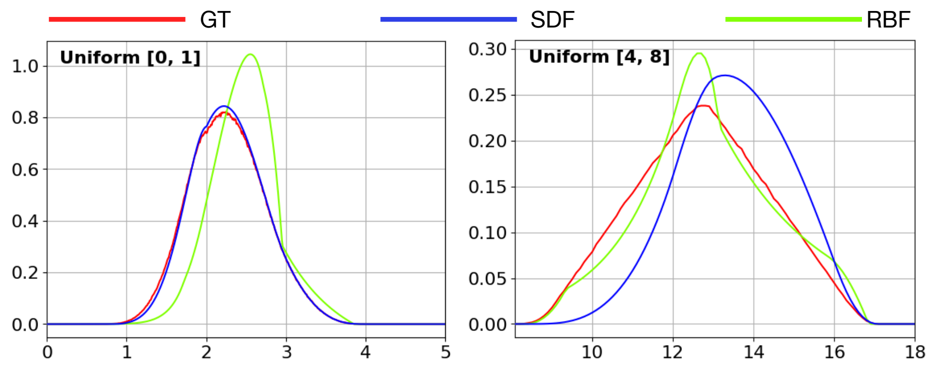
```
InnerBlockAnalysis($b$, $\Theta_c$, $\Theta_q$)
    **input** : workflow block $b$
    **output:** response time PDF $\phi(t)$ of $b$
  1 order the blocks of $b$ by the values of $C$ and $q$
  2 $v \leftarrow$ block of $b$ with max value of $C$ (and, in case of tie, with max value
      of $q$)
  3 $\phi'(t) \leftarrow$ CompositionalAnalysis($v$, $\Theta_c$, $\Theta_q$)
  4 $\hat{\phi}(t) \leftarrow$ safe upper bound PDF of $\phi'(t)$
  5 $b' \leftarrow b$ after replacing $v$ with an activity block with PDF $\hat{\phi}(t)$
  6 **return** CompositionalAnalysis( $b'$, $\Theta_c$, $\Theta_q$)
```



(a)


TOP
P
Q
R
S
T
P
Approximation of P numerical pdf
Q
act4
R
Approximation of R numerical pdf
S
act11
T
act12

(b)



(c)

Figure 2.3: The workflow of Fig. 2.2 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 ground truth (GT) curve is obtained by a 5-million-run simulation.

by the SDF heuristics and the response time of the overall workflow is high. For instance, if all activity blocks of the model of Fig. 2.2 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 Fig. 2.3c. 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 Fig. 2.3c.

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.

As mentioned in Section 2.1.2, the hierarchical nature of the structure tree enables for the extension of the considered workflow patterns. Adding new types of blocks to the formalism would reflect on the analysis technique. However, due to the compositional nature of the technique, in case a block type is added, it is sufficient to include operations that are tailored for that type of block. An example is provided in [32] to deal with loops. By introducing loops in a workflow, the state class graphs of the model will present an infinite number of classes, making the forward transient analysis unfeasible. To overcome this problem, loop block analysis can be performed by exploiting renewal theory, and in particular regenerative transient analysis, which makes the evaluation feasible by limiting the enumeration of state classes between of successive regeneration points [25].

2.3.2 Safe approximation of duration distributions

In this subsection, it is provided a definition of *stochastic ordering* among probability distributions, which is then demonstrated for the class of approximation provided in the compositional method to approximate PDFs.

Definition 5 (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 in order to reduce the complexity of the overall workflow and to make its stochastic 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 exponomial for the analysis in ORIS). To obtain safe and accurate analytical approximations of the block PDF $f(x)$, the concavity of the

CDF $F(x)$ is analyzed: 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 2.4 illustrates four cases for the concavity of $F(x)$: concave up, then concave down (Fig. 2.4a); concave down, then concave up (Fig. 2.4b); concave down (Fig. 2.4c); concave up (Fig. 2.4d). In the majority of cases in our experiments (Section 2.3), CDFs change concavity at most once, from upward to downward. The following lemma guarantees stochastic order of $\hat{F}$(x).

Lemma 1 (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, i.e. points where $\frac{d^2F(t)}{dt^2} = 0$; then, the random variable $\hat{X}$ with CDF $\hat{F}(x)$ such that*

$$\hat{F}(x) = F(x_{i_{j-1}}) + [F(x_{i_j}) - F(x_{i_{j-1}})]\frac{1 - e^{-\lambda_j (x - x_{i_{j-1}})}}{1 - e^{-\lambda_j (x_{i_j} - x_{i_{j-1}})}} \quad \text{if } x \in [x_{i_{j-1}}, x_{i_j}]_{j=1}^{M} \tag{2.7}$$

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., $\hat{X} \geqslant_{st} X$.

Proof of Lemma 1. 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 Eq. (2.7) 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 Eq. (2.7) 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, $\hat{X} \geqslant_{st} X$. □

As depicted in other works [19, 20, 21, 22, 23], the approximation of PDFs could be performed exploiting CPH distributions, which enables to fit data or analytic-numerical distributions through a semi-symbolical evaluation of parameters. Despite that, CPH distributions are not taken into account for this work. Firstly, to get an accurate approximation through CPH distributions, a large number of parameters should be exploited, making complex the process of parameter estimation. Estimation should then be ulteriorly complicated to guarantee stochastic ordering of the approximation with respect to the approximated numerical CDF. In any case, CPH distributions also present a very complex analytical form of the approximation, which leads to an explosion of additional complexity when analysis techniques such as forward transient analysis are performed [27]. For these reasons, despite CPH distributions are a common approach for the approximation of data or analytic-numerical distributions, we opted for the proposed approximant. Finally, note that the piecewise representation of the approximant combined with the truncated exponential form of each piece enables to be robust with respect to the form of the approximated function. In fact, the approximant segments the approximated function according to its inflection points, among which it is always monotone. Thus, by choosing to use a positive or negative rates exponential depending on the concavity of the function in a specific piece, it is easy to tight to the curve, ensuring a good level of accuracy. In addition, the accuracy of the approximant does not depend on the distributions associated with the elementary activities of the workflow. In fact, since the approximant is exploited to approximate the response time PDF of a block, which is obtained combining different workflow patterns, its analytic-numerical form does not belong to the same class of distributions chosen for the elementary activities.

2.3.3 Approximation safety

Our compositional analysis method is guaranteed to be safe when workflows are used to guarantee soft deadlines of SLAs. Specifically, our proofs hinge on the idea of *stochastic order* (recalled in Section 2.3.2) and on the following lemma on the order of independent replication of positively correlated random variables [62].

Lemma 2 (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 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.* $T(n)$ *is lower than* $T(n')$ $(T(n) \leqslant_{st} T(n'))$*, yielding the new structure tree* $S' = (N', E', n'_0)$*, then* $T(n_0) \leqslant_{st} T(n'_0)$*.*

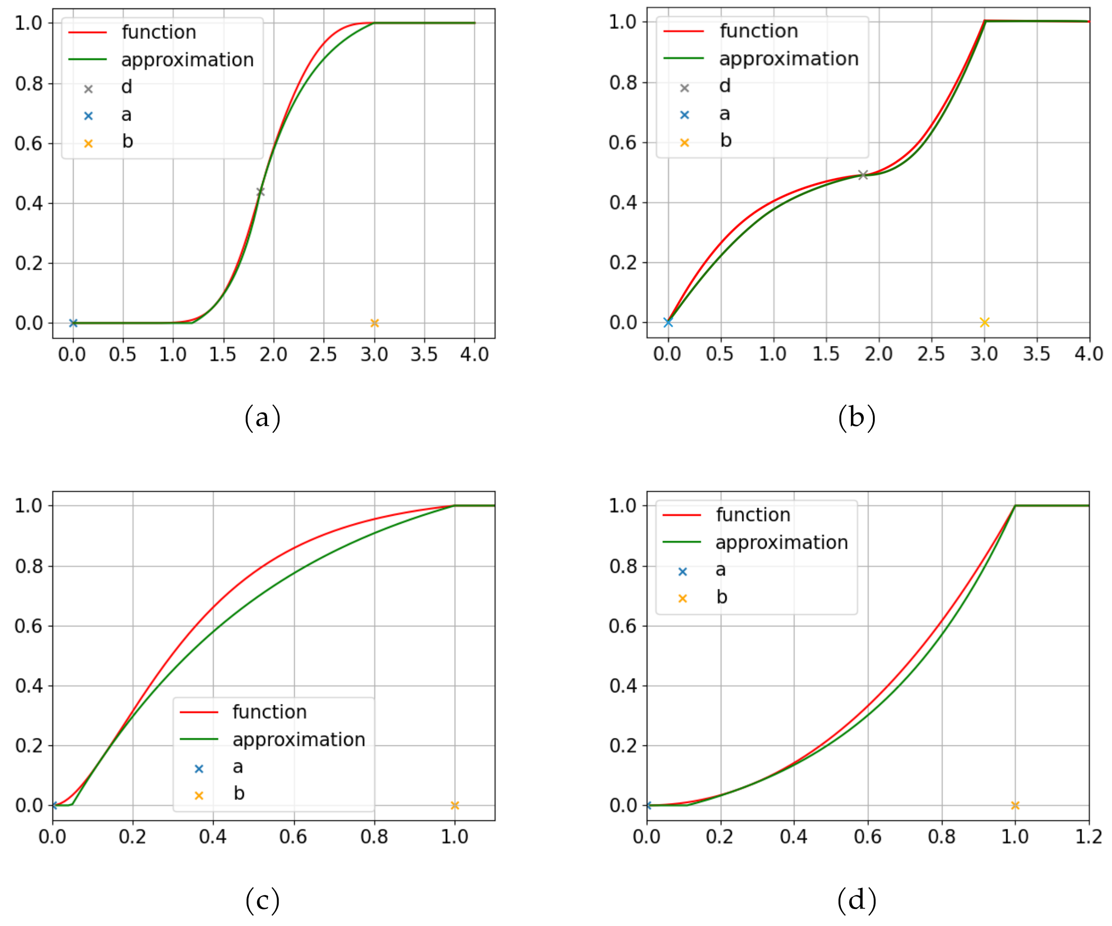


Figure 2.4: Stochastic upper bound CDF: (a) changing concavity upward to downward, (b) changing concavity downward to upward (c) fixed downward concavity and (d) fixed upward concavity of the approximated CDF.

Proof of Lemma 3. 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. $\square$

In our inner block replication, ancestors of a node v are replicated in a DAG block in order 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 (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, $T(v'_F) \geqslant_{st} T(v_F)$.*

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, the response time $T(v_F)$ of the DAG can be evaluated 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 2, $T(v'_F) \geqslant_{st} T(v_F)$. □

2.4 Experimentation

In this section, a quantitative measure to evaluate the analysis accuracy with respect to a ground truth obtained by simulation is illustrated (Section 2.4.1); accuracy and complexity of the heuristics are compared using sets of artificially and manually generated models (Section 2.4.2); the approach scalability is assessed by significantly increasing the workflow complexity (Section 2.4.3); and, the variation of accuracy of the heuristics with respect to the stochastic upper bound PDF used in [32] to approximate the response time PDF of subworkflows is evaluated (Section 2.4.4). The approach is implemented in the Eulero Java library [63] which supports definition of stochastic workflows, derivation of their response time PDFs, and random generation of workflows, exploiting the SIRIO library [57] of the ORIS tool [58] 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.

2.4.1 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 [64, 65], which quantifies the distance between two PDFs f_a and f_b as:

$$D_{JS}(f_{\mathrm{a}}\,||\,f_{\mathrm{b}}) = \frac{1}{2}D_{KL}(f_{\mathrm{a}}\,||\,Z) + \frac{1}{2}D_{KL}(f_{\mathrm{b}}\,||\,Z), \tag{2.8}$$

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

$$D_{KL}(f_{\mathrm{a}}\,||\,f_{\mathrm{b}}) = \sum_{t\in\Omega} f_{\mathrm{a}}(t)\cdot\log\left(\frac{f_{\mathrm{b}}(t)}{f_{\mathrm{a}}(t)}\right). \tag{2.9}$$

To select the number of simulation runs needed to evaluate the ground truth, for each model randomly generated in Section 2.4.2, 1-million-run, 2-million-run, ..., 5-million-run simulations are performed, using a time tick nearly three orders of magnitude lower than the width of the support of the workflow response time. Then, 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 is evaluated. 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.

2.4.2 Comparing the SDF and the RBF heuristics

Models analyzed more accurately by the SDF heuristics

A set of models is randomly generated 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 7 different models. Since DAGs blocks are the only cases in which approximation of the e2e response time is introduced, to evaluate the impact of approximation at different depth of the structure tree, each model is 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 7 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 2.1 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 Figs. 2.5 and 2.6 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 Figs. 2.5 and 2.6, 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 Figs. 2.5 and 2.6, 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 to that of simulation and averaged simulation, due to the fact that the workflow decomposition requires to execute inner block analysis three

BOTTOM				
D, B, T	**JS**			
	SDF	**RBF**	**S**	**AS**
2 2 2	**0.00001**	**0.00001**	0.02915	0.01800
4 2 2	**0.00001**	**0.00001**	0.07012	0.04065
6 2 2	**0.00005**	**0.00005**	0.10118	0.04272
2 4 2	**0.00001**	**0.00001**	0.05532	0.03476
2 6 2	**0.00001**	**0.00001**	0.06945	0.04075
2 2 4	**0.00001**	**0.00001**	0.04672	0.02770
2 2 6	**0.00000**	**0.00000**	0.07568	0.05331

TOP				
D, B, T	**JS**			
	SDF	**RBF**	**S**	**AS**
2 2 2	0.00188	**0.00151**	0.17183	0.09817
4 2 2	**0.01411**	**0.01411**	0.08571	0.05109
6 2 2	**0.00004**	0.01365	0.36369	0.24701
2 4 2	**0.00008**	0.00103	0.22228	0.13425
2 6 2	**0.00240**	0.19732	0.23625	0.13771
2 2 4	**0.00091**	**0.00091**	0.00223	0.00593
2 2 6	**0.00054**	0.01023	0.09563	0.05805

Table 2.1: Jensen-Shannon divergence (JS) of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S), and averaged simulation (AS) for the BOTTOM and the TOP configuration models of Section 2.4.2, 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. 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 ground truth (GT).

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.

BOTTOM					
D, B, T	**Times**				
	SDF	**RBF**	**S**	**AS**	**GT**
2 2 2	0.69s	0.39s	0.69s	0.69s	5518.22s
4 2 2	1.82s	1.70s	1.83s	1.83s	23268.95s
6 2 2	2.47s	2.26s	2.50s	2.50s	190396.48s
2 4 2	0.36s	0.33s	0.37s	0.37s	3167.33s
2 6 2	0.32s	0.28s	0.32s	0.32s	4573.85s
2 2 4	0.63s	0.43s	0.64s	0.64s	4526.35s
2 2 6	0.21s	0.12s	0.26s	0.26s	2344.53s

TOP					
D, B, T	**Times**				
	SDF	**RBF**	**S**	**AS**	**GT**
2 2 2	0.08s	277.53s	0.09s	0.09s	3618.21s
4 2 2	2.65s	1.59s	2.71s	2.71s	20660.48s
6 2 2	0.32s	1.01s	0.36s	0.36s	195106.73s
2 4 2	0.13s	6.89s	0.13s	0.13s	6513.53s
2 6 2	0.08s	10.44s	0.10s	0.10s	3347.82s
2 2 4	12.44s	8.95s	12.48s	12.48s	3238.71s
2 2 6	0.09s	0.30s	0.09s	0.09s	6040.17s

Table 2.2: Computation times of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S), and averaged simulation (AS) for the BOTTOM and the TOP configuration models of Section 2.4.2, 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. 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 ground truth (GT).

Models analyzed more accurately by the RBF heuristics

The 7 models M1, …, M7 are hand-crafted 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 2 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 activites 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 4 and 8. In

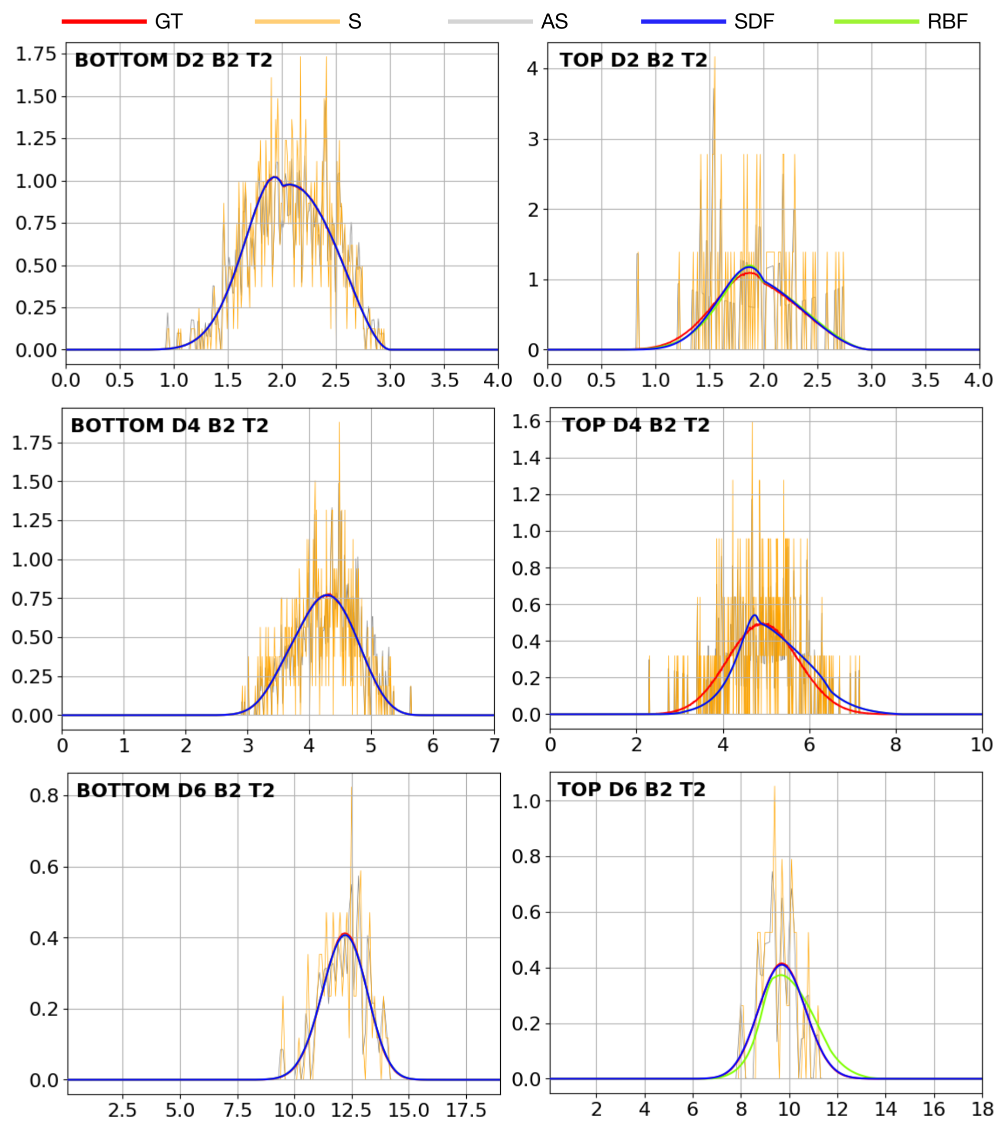


Figure 2.5: Response time PDFs of Split Dependencies First heuristics (SDF), Replace Block First heuristics (RBF), simulation (S), averaged simulation (AS), and ground truth (GT) for the BOTTOM and TOP models of Section 2.4.2, with different values of the depth D of the structure tree, $B = 2$ concurrent and alternative blocks in AND and XOR blocks, respectively, and $T = 2$ of sequential blocks in SEQ blocks.

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.

The accuracy of the heuristics, simulation, and averaged simulation are evaluated with respect to the ground truth, with the same experimental setup of Section 2.4.2, 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 2.3 and Fig. 2.7. 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, with a JS divergence gain of at least a factor of 4, and of nearly two orders of magnitude in the best cases. Overall, it was not trivial 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 2.3.1). Therefore, except when these conditions occur, the SDF heuristics is preferable with respect to the RBF heuristics.

Model	**JS**			
	SDF	**RBF**	**S**	**AS**
1	0.04506	**0.01254**	0.03382	0.02127
2	0.04526	**0.01256**	0.05248	0.03376
3	0.03756	**0.00665**	0.04640	0.04326
4	0.03737	**0.00660**	0.01350	0.01126
5	0.12228	**0.00909**	0.05926	0.03398
6	0.03837	**0.00820**	0.01449	0.01015
7	0.03962	**0.00456**	0.00434	0.00852

Model	**Time**				
	SDF	**RBF**	**S**	**AS**	**GT**
1	1.42s	1.28s	1.31s	1.31s	9287.89s
2	0.46s	0.64s	0.66s	0.66s	9549.46s
3	0.20s	0.31s	0.33s	0.33s	5776.45s
4	0.16s	0.85s	0.86s	0.86s	6657.49s
5	0.21s	0.22s	0.22s	0.22s	6692.07s
6	0.13s	0.84s	0.86s	0.86s	6050.73s
7	0.21s	3.08s	3.09s	3.09s	8979.62s

Table 2.3: Jensen-Shannon divergence (JS) and computation times of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S) and averaged simulation (AS) for the models of Section 2.4.2. For each model (i.e., for each row), the best (i.e., lowest) JS value is highlighted in bold

2.4.3 Increasing workflow complexity

The evaluation complexity is further stressed by generating workflows with parameters $D = 4$, $B = 4$, $T = 4$ and $D = 6$, $B = 4$, $T = 4$, notably obtaining huge

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

Table 2.4: Computation times of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S), and averaged simulation (AS) for the BOTTOM and TOP models of Section 2.4.3, 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.

models having up to 448 and 7168 simple activities, respectively, for which obtaining a ground truth via stochastic simulation is definitely not viable. The computation times are shown in Table 2.4 and the analysis results in Fig. 2.8. Although the computation times increase with respect to the cases described in Section 2.4.2, 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 [66, 67, 68, 69] 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 nearly 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 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).

2.4.4 Sensitivity to the stochastic upper bound PDF

A sensitivity analysis with respect to the stochastic upper bound PDF used to approximate the numerical solutions of inner blocks is performed. To this end, two randomly generated models are considered. In particular, these models are generated using 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 7 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. Two experiments are conducted: in the first case, the approximation defined in Section 2.3.2 is used (A1); in the second case, a variant with bounded support of the approximant proposed in [32] is used, 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 Fig. 2.9 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 2.3.2 is more accurate than the alternative.

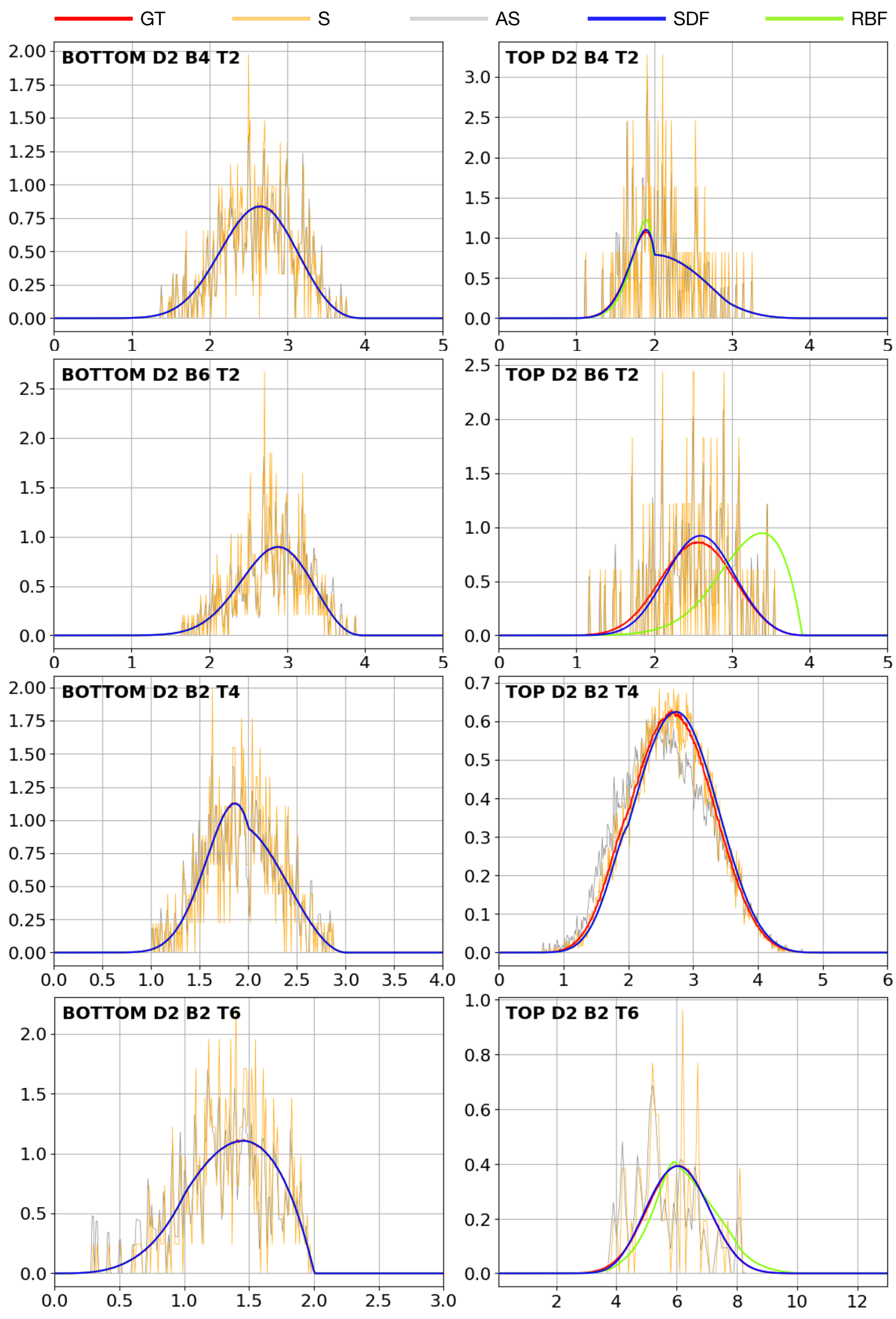


Figure 2.6: Response time PDFs of Split Dependencies First heuristics (SDF), Replace Block First heuristics (RBF), simulation (S), averaged simulation (AS), and ground truth (GT) for the BOTTOM and TOP models of Section 2.4.2 with depth $D = 2$ of the structure tree, and different values of the number B of concurrent and alternative blocks in AND and XOR blocks, respectively, and the number T of sequential blocks in SEQ blocks.

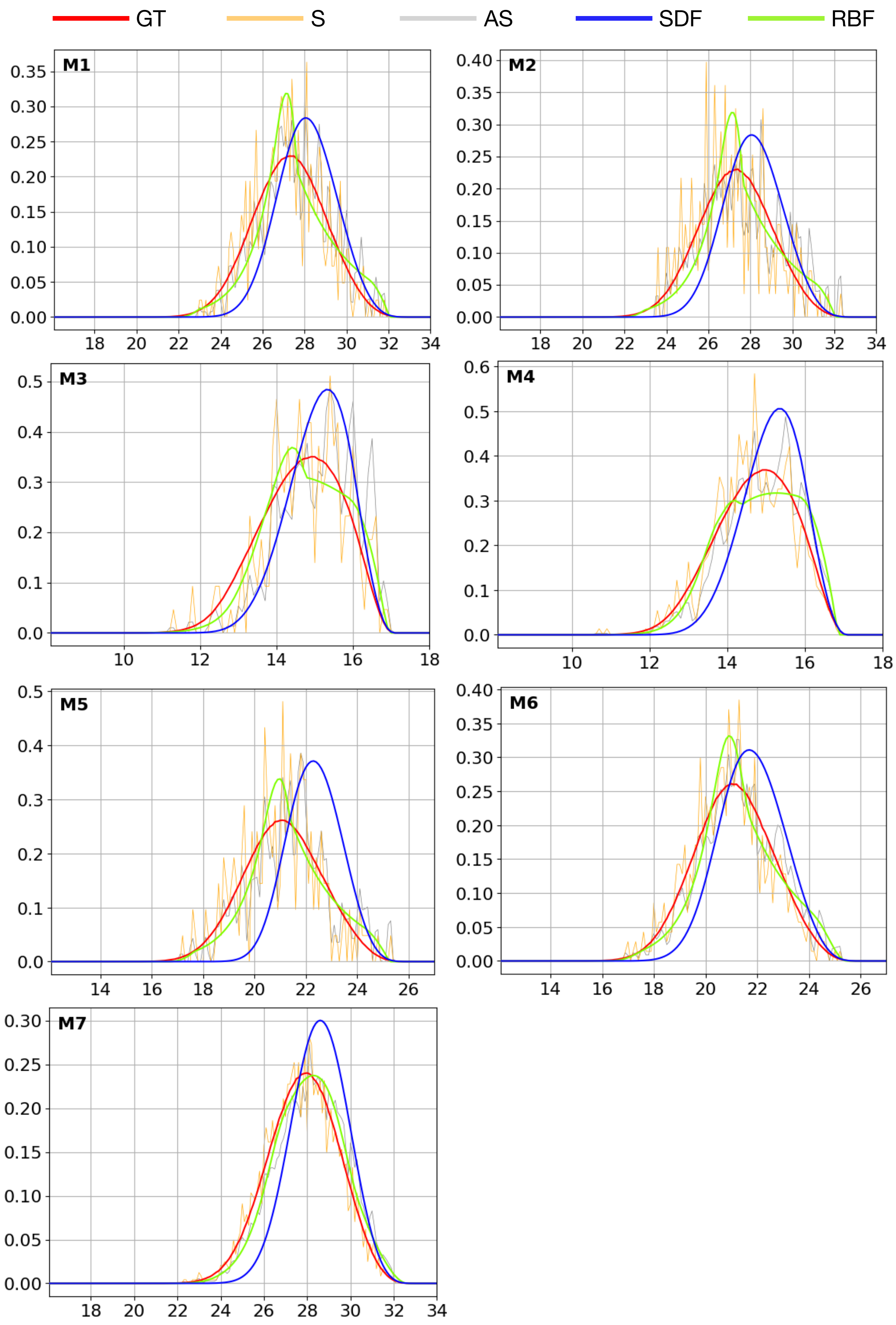


Figure 2.7: Response time PDFs of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S), averaged simulation (AS), and ground truth (GT) for the models of Section 2.4.2.

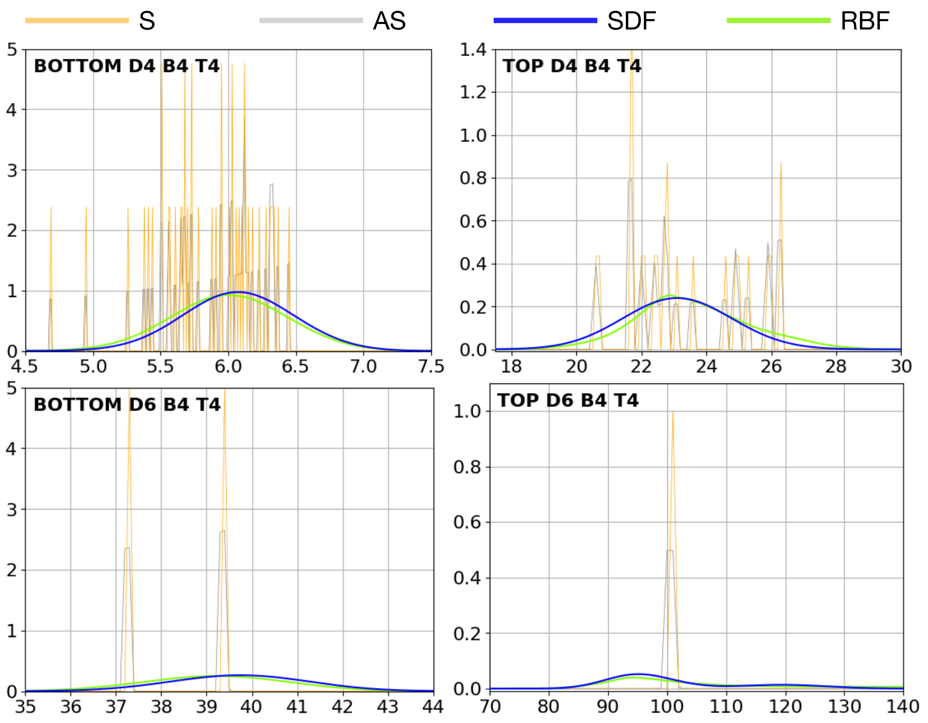


Figure 2.8: Response time PDFs of Split Dependencies First heuristic (SDF), Replace Block First heuristic (RBF), simulation (S), and averaged simulation (AS) for the BOTTOM and TOP models of Section 2.4.3, with structure tree depth $D \in \{4, 6\}$, $B = 4$ concurrent and alternative blocks in AND and XOR blocks, respectively, and $T = 4$ sequential blocks in SEQ blocks.

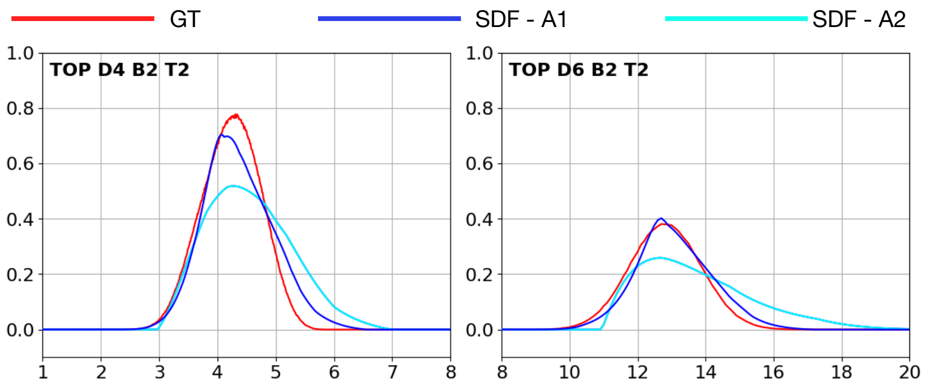


Figure 2.9: Response time PDFs of the ground truth (GT) and Split Dependencies First (SDF) heuristics using the approximant PDF of Section 2.3.2 (A1) and a variant with bounded support of the one proposed in [32] (A2) for the models of Section 2.4.4, with structure tree depth $D \in \{4, 6\}$, $B = 2$ concurrent and alternative blocks in AND and XOR blocks, respectively, $T = 2$ sequential blocks in SEQ blocks, and DAG blocks at the top level.

Chapter 3

The Eulero Library

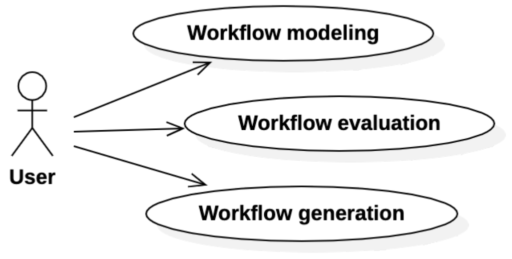
In this chapter, it is presented *Eulero*, a Java library that implements the compositional approach of Chapter 2 for efficient and accurate evaluation of the response time PDF of complex workflows, consisting of activities with GEN duration with bounded support, composed through sequence, choice/merge, and split/join blocks, with unbalanced split and join constructs that break the structure of well-formed nesting. The library (Section 3.1) is organized in three packages, supporting workflow modeling (Section 3.2), workflow evaluation (Section 3.3), and random generation of workflow models (Section 3.4). A workflow is modeled as a hierarchy of sub-workflows using structure trees, providing not only ease of modeling but also efficiency of analysis by facilitating the identification of sub-workflows that can be separately analyzed. Eulero uses the SIRIO Library of the ORIS tool [70] to represent monovariate PDFs, to model sub-workflows as Stochastic Time Petri Nets (STPNs) [33], and to perform their transient analysis to derive the sub-workflow response time PDF. The library is designed with the aim of facilitating usability, maintainability, and extensibility, by exploiting consolidated programming design patterns [71].

3.1 Library overview

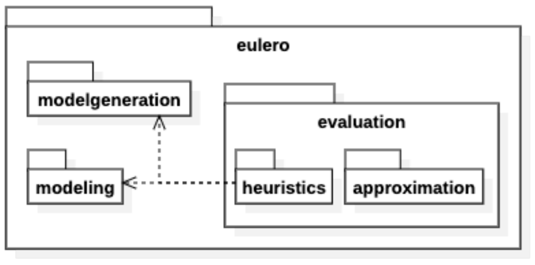
Eulero is a *headless* Java library providing capabilities of modeling and evaluating complex workflow of activities. In particular, the library implements the structure tree representation and the heuristics compositional approach described in Chapter 2. The UML use case diagram of Fig. 3.1a shows the main functionalities of the Library, which are implemented in the packages of Fig. 3.1b. In particular:

- The package `modeling` supports modeling of workflows affording various aspects of complexity (i.e., bounded generally distributed durations, high concurrency degree, unbalanced split and join constructs breaking the structure of well-formed nesting) in terms of structure trees, a hierarchical representation enabling accurate and efficient evaluation of the response time PDF.
- The package `evaluation` includes the sub-package `heuristics`, implementing compositional methods for the evaluation of the workflow response time PDF, and the sub-package `approximation`, supporting the derivation of the analytical form of stochastically ordered approximants of numerical PDFs.
- The package `modelgeneration` provides functionalities to randomly generate models according to the specification of the package `modeling`.

Eulero uses the SIRIO library for the representation of the analytical form of monovariate PDFs, for the derivation of their response time PDF of sub-workflows, and for the estimation of complexity of this evaluation.



(a)



(b)

Figure 3.1: Eulero library: (a) UML use case diagram and (b) package diagram.

Many tools for performance analysis of stochastic models include a Graphical User Interface (GUI) [35, 58, 72], enabling model validation or quick testing. However, Eulero is designed to perform quantitative evaluation of large-scaled models, for which an GUI-driven approach could be time-consuming and error-prone. Programmatically defining and instancing hierarchical models speeds up the modeling process and enables automatic model modifications (e.g., changing activity probability distributions or complexity factors of the workflow generator), facilitating experimentation and increasing its robustness. Despite that, as part of future extension, a GUI could be designed to perform preliminary experiments on basic models or to validate model specifications. The Eulero library is available at `https://github.com/oris-tool/eulero` under the AGPL licence.

3.2 The package *Modeling*

The package `modeling` is responsible for the instancing of workflows specified as structure trees. In this section, a description of the package (Section 3.2.1) and an application example of the package (Section 3.2.2) are provided.

3.2.1 Package description.

The package `modeling` is responsible for building stochastic workflows as Java objects. The related UML class diagram is shown in Fig. 3.2 and a code snippet illustrating the programmatic specification of the model of Fig. 2.2 is provided in Listing 3.1. Workflow modeling is implemented through the design pattern Composite [71], where the abstract class `Activity` defines a common interface, which is implemented by subtype classes `Simple`, `XOR`, `DAG`, `AND`, and `SEQ`. In particular, `Activity` defines block types common attributes, such as the minimum `min` and maximum `max` response time of the block, the degrees of concurrency `C` and `simplifiedC` corresponding to C and $\bar{C}$ respectively, the degrees of sequencing `Q` and `simplifiedQ` corresponding to q and $\bar{q}$ respectively, and the abstract methods `buildSTPN()` and `buildTPN()`, which are overridden by sub-typing classes to define the transformation of the block structure tree into the block STPN and its underlying TPN, respectively. Note that it is precisely through the composite design pattern that we are able to provide the hierarchical representation of the workflow.

The class `Simple` enables the instantiation of simple activities whose piece-wise duration PDF is encoded in the field `features` as a list of `StochasticTransition-Feature`. The latter is a class of the SIRIO library defining the support and analytical form of a PDF. Each feature is assumed to have unit measure, and therefore it is weighted by a value that guarantees unit measure over the whole PDF. These values are encoded by the attribute `weights`.

The class `XOR` enables the instantiation of XOR blocks having different alternative branches. Branches are referenced through the field `alternatives` and associated with a probability value encoded in the field `probabilities`.

The class `DAG` enables the instantiation of DAG blocks. `DAG` objects have three references to class `Activity`. The reference `activities` consists of a list referencing all the activities nested in a DAG block. The references `begin` and `end` refer to the initial

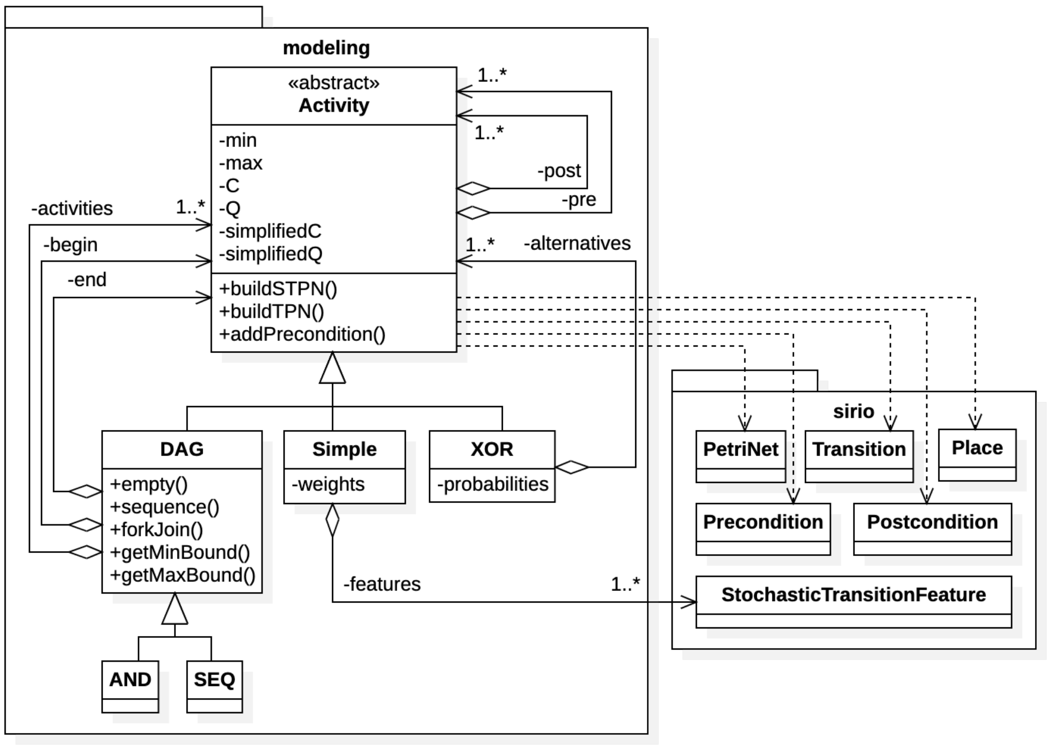


Figure 3.2: UML class diagram of the package `modeling`.

and final fictitious activities, respectively, that are required to define a DAG block as a SESE block. The structure of a DAG is defined by adding preconditions to its internal activities through the method `addPrecondition()`, which updates the list attributes `pre` and `post` of the DAG activities. The `DAG` class also exposes three static methods `empty()`, `sequence()` and `forkjoin()`. The method `empty()` enables the generation of empty DAGs that can then be built by adding activities and preconditions, evaluating the response time bounds, and referencing the added activities. The methods `sequence()` and `forkjoin()` enable the generation of SEQ and AND blocks, respectively, which are implemented as derivation classes of `DAG`. Although `AND` and `SEQ` classes do not extend `DAG` functionalities, they have been treated as separate classes in order to handle them as a specific case in the analysis algorithm.

As mentioned in Sections 2.1.2 and 2.3.1, as soon as the representation formalism and the solution method are extended to deal with different workflow patterns, such as *k-out-n* or *cycle/loops*, also the Eulero Library should be updated to enable for the representation and the analysis of such patterns. This can be achieved by simply extending the `Activity` class through a specialization enabling to fully characterize the new desired pattern. Then, as depicted in Section 3.3, also the solution method requires to be updated, in order to solve the added pattern.

3.2.2 Application example

In Listing 3.1, blocks Q, R, S and T are created using the static methods of `DAG` and the constructors of `XOR` and `Simple` (lines 3 to 52). Then, the top node is created as a DAG block through the static method `empty()`, and populated connecting inner blocks by adding preconditions (lines 54 to 59). In doing so, it is mandatory that final blocks be preconditions of the node `end` (line 59), and that all initial nodes have the node `begin` as precondition (lines 55 and 56). Since DAGs are built step-by-step adding preconditions, then `min` and `max` are estimated and set using the related setter methods (lines 60 and 61), and the added `activities` are also referenced using the related setter method (line 62).

```
 1 StochasticTransitionFeature feature = StochasticTransitionFeature.
       newUniformInstance("0", "1");
 2
 3 Activity P = DAG.sequence("P",
 4     new Simple("P1", feature),
 5     DAG.sequence("P2",
 6         new Simple("P2A", feature),
 7         new Simple("P2B", feature)
 8
 9     ),new Simple("P3", feature)
10 );
11
12 Activity Q = new Simple("Q", feautre);
13
14 Activity R = DAG.forkJoin("R",
15     DAG.forkJoin("R1",
16         DAG.forkJoin("X",
17             new Simple("X1", feature),
18             new Simple("X2", feature)
19         ),
20         DAG.sequence("Y",
21             new Simple("Y1", feature),
22             new Simple("Y2", feature)
23         )
24     ),
25     new XOR("R2",
26         List.of(
27             new Simple("R2A", feature),
28             new Simple("R2B", feature)
29         ),
30         List.of(0.3, 0.7)
31     )
32 );
33
34 Activity S = new Simple("S", feature);
35 Activity T = new Simple("T", feature);
36
37 DAG top = DAG.empty("TOP");
38 P.addPrecondition(top.begin());
39 Q.addPrecondition(top.begin());
40 R.addPrecondition(top.begin());
41 S.addPrecondition(P, Q, R);
42 T.addPrecondition(R);
```

```
top.end().addPrecondition(T, S);
top.setMin(top.getMinBound(top.end()));
top.setMax(top.getMaxBound(top.end()));
top.setActivities(Lists.newArrayList(P, Q, R, S, T));
```

Listing 3.1: Construction of the workflow structure tree of Fig. 2.2.

The method `buildSTPN()` is an abstract method responsible for the construction of the workflow STPN. Construction is realized recursively by exploring the workflow structure tree, and adding `Place`, `Transition`, `Precondition` and `Postcondition` objects to a `PetriNet` object depending on the tree topology (the latter 5 classes belong to the SIRIO library). The method `buildSTPN()` of `DAG`, `AND`, and `XOR` merely adds places and immediate transitions, representing fork, join, choice, and merge structures specifying the nesting of inner blocks, before the method is called recursively by the inner blocks. The method `buildSTPN()` of `SEQ` simply chains the recursive calls made for inner activities belonging to the considered sequence. The method `buildSTPN()` of `Simple` adds a random switch of transitions whose distributions are encoded by the fields `features` and `weights`. The `buildTPN()` method is the abstract method that enables the construction of the underlying TPN. Similarly to `buildSTPN()`, it assigns temporal and complexity features to the workflow activities, so that these information can be exploited during the block complexity analysis.

3.3 The package *Evaluation*

The package `evaluation` implements the compositional technique described in Section 2.3 for the evaluation of the response time PDF of stochastic workflows. In the section, an overview of the analysis heuristics (Section 3.3.1) is recalled before showing how to use of the package (Section 3.3.2).

3.3.1 A recall on the analysis heuristics

The end-to-end response time PDF of a workflow is evaluated by composing the results of separate analyses of sub-workflows. In turn, the sub-workflows are identified by a recursive exploration of the structure tree, which selects the most appropriate action to analyze a block, based on its type or complexity measures and according to the provided heuristics of analysis. Four actions are considered:

- *Numerical analysis* combines the numerical response time PDFs of the components of well-nested blocks, providing the overall response time PDF, as shown in Section 2.1.3.
- *Forward transient analysis* is suitable to evaluate not well-nested blocks with limited complexity, providing the numerical form of their response time PDF.

- *Inner block analysis* can be applied to blocks with high complexity measures. It selects an internal block, evaluates its numerical response time PDF with some action and replaces it with a new activity block, whose probability density function is evaluated approximating the numerical response time PDF. The functioning of the action is illustrated in Section 2.3.1.

- *Inner block replication* can be applied to complex not well-nested blocks. It identifies two sub-workflows sharing activities, separates them by replicating shared activities, and recombines them as children of an AND block, guaranteeing that the resulting response time PDF is a stochastic upper bound of the exact one (see Lemma 1). If one or both sub-workflows had become well-nested, they would subsequently be evaluated through *numerical analysis*. The functioning of the action is illustrated in Fig. 2.3a.

Heuristics are defined to explore the structure tree and evaluate the workflow response time. In Section 2.3.1, two heuristics are defined, which both use *numerical analysis* for well-nested blocks and *forward transient analysis* for simple DAGs, but differ in the way complex DAGs are evaluated, by favouring either *inner block replication* or *inner block analysis*, as shown in Algorithms 1 to 3. Specifically, considering a workflow block b:

- If b is an activity block, then its exact response time PDF is its duration PDF for both heuristics (line 2 and 3).

- If b is, or can be reduced to, a well-nested composition of independent sub-workflows, then a numerical analysis is recursively applied to get the exact response time PDF, for both heuristics (lines 4 to 12).

- If b is a DAG block, *inner block analysis*, *inner block replication* or *forward transient analysis* are applied according to the complexity measures of the block, and depending on the considered heuristics (see Algorithms 2 and 3).

Pseudocodes of the analysis heuristics are provided in Section 2.3.1.

3.3.2 Package description

The package `evaluation` implements the recalled compositional evaluation method. The related UML class diagram is provided in Fig. 3.3. The package contains the inner packages `heuristics` and `approximator`, both of which implement the design pattern Strategy [71].

In the package `heuristics`, the abstract class `AnalysisHeuristicsStrategy` implements the methods `numericalXOR()`, `numericalAND()`, `numericalSEQ()`, for numerical analysis operations on well-nested block types `XOR`, `AND`, `SEQ`, respectively, and `forwardAnalysis()`, `innerBlockAnalysis()`, and `innerBlockReplication()`, which implement the remaining actions. Each of these methods makes one or more recursive calls to the abstract method `analyze()`, enabling the top-down evaluation of any structure tree. The order of execution of the actions is specified by overriding the `analyze()` method, using the fields `CThreshold` and `QThreshold` as the thresholds Θ_c and Θ_q for the complexity measures, respectively. The heuristics of Algorithms 2 and 3 are implemented by the classes `SDFHeuristics` and `RBFHeuristics`, respectively. Note that the Strategy Pattern facilitates the addition of new heuristics strategies by extending the abstract class `AnalysisHeuristicsStrategy` with a new concrete class. Moreover, it is trivial to implement new actions as non-abstract methods of the class `AnalysisHeuristicsStrategy`. This enables to perform analysis of additional workflow pattern, as mentioned in Section 3.2.1.

The package `approximation` implements different approximation methods for numerical monovariate PDFs. The abstract class `Approximator` provides the abstract method `getApproximationStochasticTransitionFeatures()`, which processes a numerical monovariate PDF and returns an approximant piecewise PDF represented as a list of weights and `StochasticTransitionFeature` objects. Concrete approximators define the approximation logic by overriding this method. Again, the Strategy pattern enables an easy extension of the package. The class `AnalysisHeuristicsStrategy` uses the class `Approximator` in the method `innerBlockAnalysis()`: when this method is called, the most complex inner block of a DAG is picked up and evaluated by recursive call of `analyze()`, then the result is approximated by the selected `Approximator`, and finally, the obtained features and weights are passed to the constructor of the class `Simple` to generate the simple activity that replaces the complex inner block.

3.3.3 Application Example

Listing 3.2 shows how to evaluate the model built in Listing 3.1. Considering values `tC` and `tQ` for complexity thresholds Θ_c and Θ_q, respectively, analysis horizon `timeLimit`, numerical precision `step`, and a specific `approximator` (lines 1 to 5), the analysis is performed by creating the heuristics (line 6) and calling `analyze()` (line 8). Fig. 3.4 shows the evaluated response time PDF and CDF.

```
1 BigInteger tC = BigInteger.valueOf(3);
2 BigInteger tQ = BigInteger.valueOf(7);
3 BigDecimal timeLimit = model.max();
4 BigDecimal step = BigDecimal.valueOf(0.01);
5 Approximator approximator = new EXPMixtureApproximation();
6 AnalysisHeuristicStrategy strategy = new SDFHeuristics(tC, tQ, approximator);
7 double[] cdf = strategy.analyze(model, timeLimit, step);
```

Listing 3.2: Evaluation of a workflow structure tree.

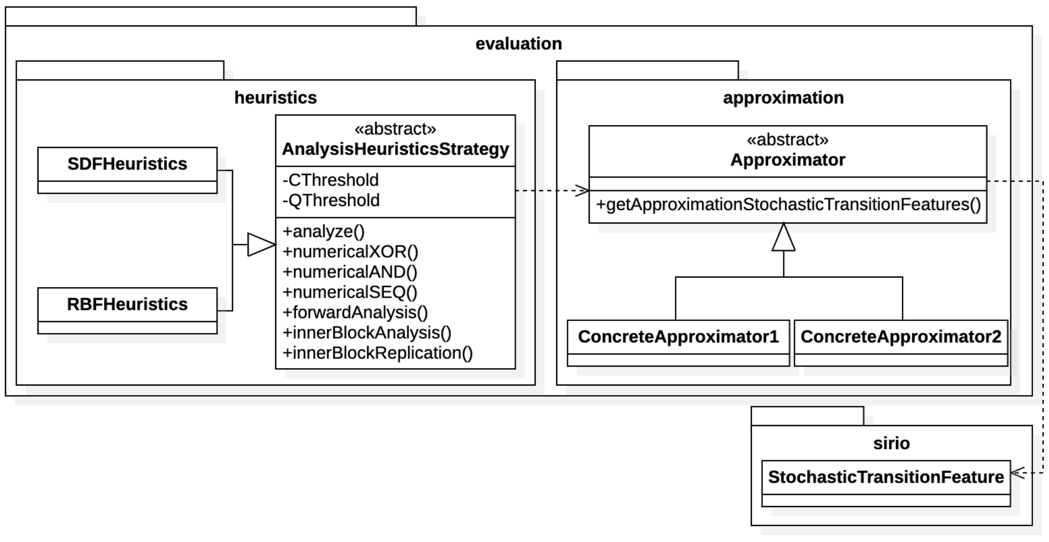


Figure 3.3: UML class diagram of the package `evaluation`.

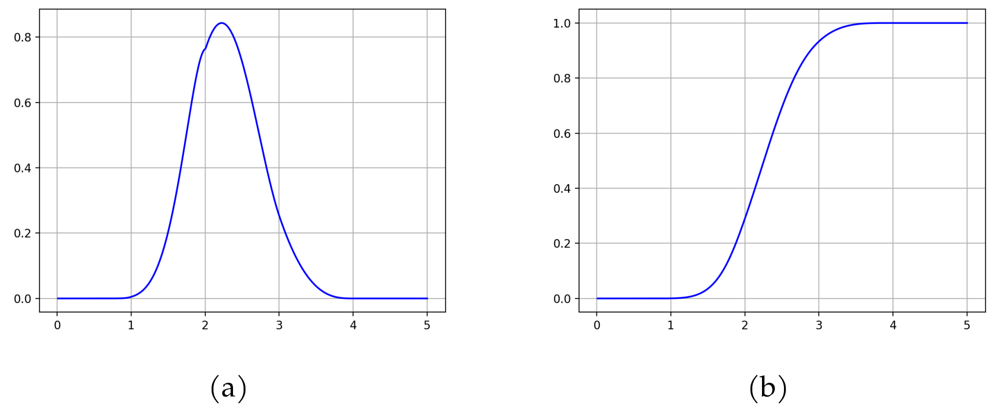


Figure 3.4: PDF (a) and CDF (b) evaluated for the workflow of Fig. 2.2.

The analysis algorithm scales even on more complex models. In fact, analysis heuristics have been tested for structure trees having depth up to 6, concurrency and sequencing degree up to 64, and whose corresponding STPNs contain up to 7000 transitions. In all cases, analysis heuristics always took less than 1 s.

3.4 The package *ModelGeneration*

Many problems can be modeled as workflows, and experimention enabled by tools such as TGFF [73], Apache Airflow [74] or Luigi [75] results to be an added value. However, there is often a lack of models on which to conduct experiments, and when there are, they are often not complex enough to justify the use of compositional methods. In this section, a generation strategy to randomly build workflows is presented. The random generation of a workflow can be exploited to test quantitative evaluation methods, stressing different complexity factors, such as the degree of parallelism and sequencing. In particular, the process of random generation of a workflow is illustrated (Section 3.4.1) and the Java implementation of the generation procedure is described through an illustrative example (Section 3.4.2).

3.4.1 Random Generation

Algorithm 6 implements the proposed strategy for random generation of workflow structure trees. Specifically, given the depth of the structure tree, random generation of a workflow is carried out through a recursive procedure, which builds well-nested or DAG blocks in different ways, depending on the type of block that is drawn (lines 2 and 12). If a well-nested type (i.e., SEQ, AND, XOR) is selected, the number of its children is drawn too, and these are generated by the recursive call of the generation procedure (lines 3 to 5). Then, the block of the drawn type is created, assigning the created children to it (lines 6 to 11).

Algorithm 6: Random generation of a workflow

```
GenerateBlock($d$, $settings$)
    input  : depth level $d$, generation settings $settings$
    output: workflow $b$
 1  if $d > 0$ then
 2      if type is SEQ or AND or XOR then
 3          $children \leftarrow \{\}$
 4          foreach $i \in \{2, ..., \texttt{RandomMaximumChildren()}\}$ do
 5              $children \leftarrow children \cup$ GenerateBlock($d - 1$, $settings$)
 6          if type is SEQ then
 7              return GenerateSEQBlock($children$)
 8          if type is AND then
 9              return GenerateANDBlock($children$)
10          if type is XOR then
11              return GenerateXORBlock($children$)
12      if type is DAG then
13          return GenerateDAGBlock($parameters$)
14  return GenerateSimpleBlock()
```

A DAG block consists of a set of nodes sorted according to a topological order in several consecutive levels, such that every node can have predecessor nodes in previous levels (see Fig. 3.5). In particular, a node belongs to the *i*-th level if, for all paths from the initial fictitious node to the considered one (both not included), the longest paths contain exactly *i* nodes. For instance, node E in Fig. 3.5 belongs to level

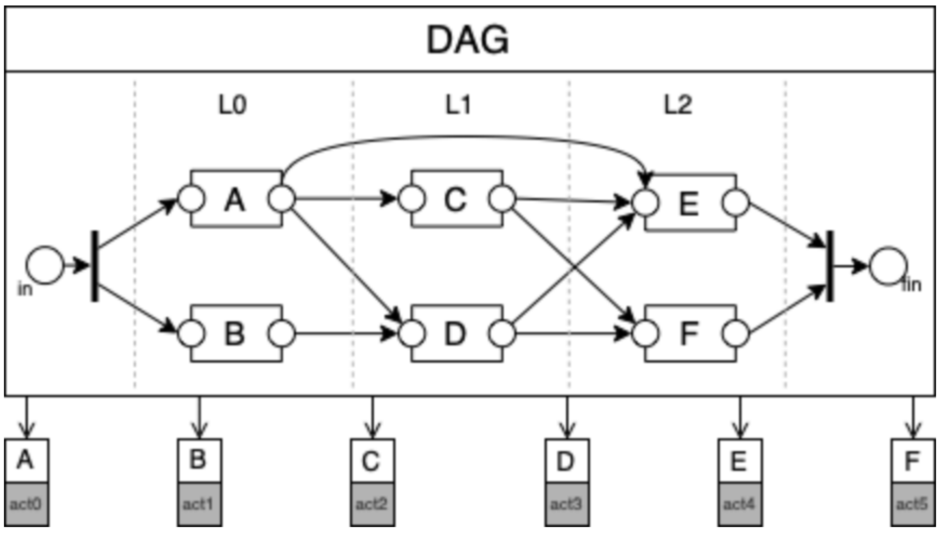


Figure 3.5: A DAG block with nodes sorted according to a topological order.

`L2` because, among all the paths from the initial node `in` to node `E`, the longest paths contain 2 nodes. Since DAG blocks break the well-formed nesting of concurrent blocks, yet ensuring that execution ends w.p.1, both conditions must be fulfilled to generate a not well-nested DAG. The first condition holds when there is at least one level where *i*) at least two nodes share at least one predecessor and *ii*) their predecessor sets do not coincide (e.g., `C` and `D` share A as predecessor, but `D` also has `B`) or coincide but contain at least 2 predecessors (e.g., `E` and `F` share all the predecessors, which are more then 1; otherwise, they would have been well-nested). The second condition holds if every node has at least one predecessor and at least one successor.

Hence, random generation of a DAG (lines 12 and 13 of Algorithm 6) is achieved by drawing the number of levels, the number of nodes for each level, and randomly connecting nodes of different levels, ensuring that the two conditions described above are met. To guarantee not well-formed nesting, it is sufficient to randomly select two nodes from some level $i > 0$, and connect both of them to the same predecessor node, and one of them to a different predecessor node, randomly drawn from the $(i-1)$-th level. To guarantee that a block is a SESE, every node of a level is connected to at least one predecessor and one successor, except for the first-level nodes, which share the same and only predecessor node (the initial fictitious node `in`), and the last-level nodes, which share the same and only successor node (the final fictitious node `fin`). Finally, at the lowest level of the workflow structure tree, a simple block is created (line 14 of Algorithm 6).

3.4.2 Package description

The package `modelgeneration` implements the approach of Section 3.4.1 for random generation of workflow structure trees. The UML class diagram of the package is shown in Fig. 3.6. The class `RandomGenerator` defines the recursive method `generateBlock()`, which implements the generation logic based on the depth `treeDepth` of the structure tree and on some setting parameters referenced by list `settings`. Each item of the list refers to a certain level of depth of the workflow, and collects a set of `BlockTypeSetting` objects, through which generation is driven. In particular, `BlockTypeSetting` is an abstract class that specifies a `type` and a `probability` of being drawn. For every item of the list `settings`, the sum of `BlockTypeSetting-probability` fields must be equal to 1. `BlockTypeSetting` is extended to define block type related parameters, such as the minimum and maximum number of children for class `WellNestedBlockSetting`, or the minimum and maximum number of levels, nodes per level, connections between nodes and distance between nodes belonging to not consecutive levels, for class `DAGBlockSetting`. Note that `XORBlockSet-ting`, `ANDBlockSetting`, and `SEQBlockSetting` do not add functionalities to `WellNestedBlockSetting`, but are required to generate XOR, AND, and SEQ blocks in the method `generateBlock()`, respectively. In addition, it is possible to constrain the structure of a DAG by varying its generation parameters.

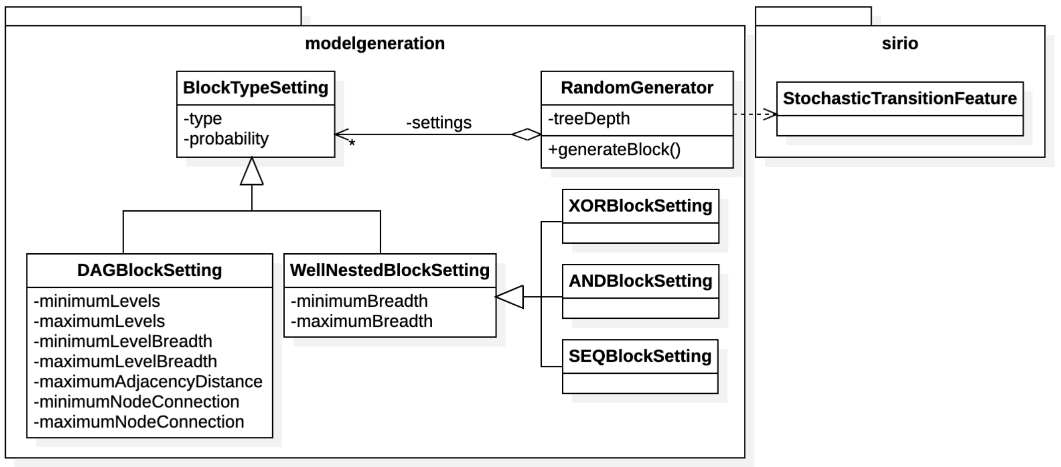


Figure 3.6: UML class diagram of the package `modelgeneration`.

3.4.3 Application example

Listing 3.3 shows how to use the package `modelgenerator` to randomly generate workflow structure trees. For each level of the tree, at least one `BlockTypeSetting` object must be created and added to that level setting variable, specifying the occurrence probability of that block type and the related parameters (e.g., lines 4 to 8 and lines 10 to 12 for levels 1 and 2, respectively). Every level setting is added to a global setting variable (lines 14 to 16) which is then passed to a `RandomGenerator` object (line 19). Finally, the model is created by invoking the method `generateBlock()` of the `RandomGenerator` object.

```
 1 int concurrencyDegree, sequenceFactor;
 2 concurrencyDegree = sequenceFactor = 3;
 3 
 4 Set<BlockTypeSetting> level1Settings = new HashSet<>();
 5 BlockTypeSetting AND = new ANDBlockSetting(0.5, concurrencyDegree);
 6 BlockTypeSetting SEQ = new SEQBlockSetting(0.5, sequenceFactor);
 7 level1Settings.add(AND);
 8 level1Settings.add(SEQ);
 9 
10 Set<BlockTypeSetting> level2Settings = new HashSet<>();
11 BlockTypeSetting DAG = new DAGBlockSetting(1.);
12 level2Settings.add(DAG);
13 
14 ArrayList<Set<BlockTypeSetting>> settings = new ArrayList<>();
15 settings.add(level1Settings);
16 settings.add(level2Settings);
17 
18 StochasticTransitionFeature feature = StochasticTransitionFeature.
       newUniformInstance("0", "1");
19 RandomGenerator randomGenerator = new RandomGenerator(feature, settings);
20 Activity model = randomGenerator.generateBlock(settings.size());
```

Listing 3.3: Random generation of a workflow structure tree.

Chapter 4

A game-theoretical approach for workflow service selection

This chapter addresses the case of a hierarchical infrastructure where the association among customers and service providers is inter-mediated by aggregators. Customers demand to resource-constrained aggregators for workflow execution, exhibiting soft deadline SLOs over the workflows e2e times, expressed in terms of Cumulative Distribution Functions (CDFs). In turn, aggregators offer service outsourcing to a set of antagonist providers competing for service delivery, whose bids are represented by CDFs of service completion times. The designed stochastic framework applies matching theory to associate customers and aggregators, and a nested Vickrey-Clarke-Groves (VCG) auction to perform service selection.

The chapter is organized as follows. In Section 4.1, the problem statement is detailed and an overview of the approach is provided. Section 4.2 illustrates the Vickrey-Clarke-Groves (VCG) auction, with prices offered by bidders expressed in terms of the earliness of services to be delivered, quantified as CDF of the service completion time rather than fixed requirements vector of reference literature. The section also investigates the theoretical consequences of the considered auction, mechanism and the impact of cheating strategies on the total welfare, emphasizing the VCG auction ability in discouraging cheating behaviors. Section 4.3 illustrates a matching game with externalities (i.e., with dependencies among preferences of customers) developed to solve the resource-constrained assignment problem between customers and aggregators, integrating the VCG auction mechanism to perform provider selection. In Section 4.4, the variation of total welfare (i.e. the utility gained by all aggregators and providers) is evaluated in terms of the Price of Anarchy (PoA) [45, 46] paid to take decisions that rely on partial information metrics, the impact of delays on service execution and the behaviour of both the aggregator and provider utility, under different assumptions about the competitiveness among providers, are analyzed.

4.1 Problem statement and approach

In this section, it is illustrated the model designed the express the interactions among the entities involved in a system where web services are composed to provide complex functionalities (Section 4.1.1). In particular, the model includes the representation of customers that require the execution of workflows of services, aggregators that aggregates web services provided from different set of service providers, which in turn have different capabilities and so, different time of execution. Then, we provide a formulation of the problem of how to match customers with aggregators who are served by different service providers, with the aim of maximizing both the reward of aggregators and providers to whom service executions are assigned (Section 4.1.2). Finally, an overview of the solution method is provided (Section 4.1.3).

4.1.1 System model

. The system model illustrated in the UML class diagram in Fig. 4.1 is considered. The model includes three types of participants:

- a set $\mathcal{C} = \{1, ..., C\}$ of customers, each requiring repeated execution of a workflow composing services from a common catalog $\mathcal{S} = \{1, ..., S\}$, to be completed with a required distribution F_c of the end-to-end completion time;
- a set $\mathcal{P} = \{1, ..., P\}$ of antagonistic providers, with each provider p offering an implementation for some service, with an offered and an actual distribution of the completion time;
- a set of aggregators $\mathcal{A} = \{1, ..., A\}$ which intermediate the relation between customers and providers; to this end, each aggregator a can recourse to a subset of available providers $\mathcal{P}_a \subseteq \mathcal{P}$, and it assumes the responsibility to select a provider for each service in each workflow, to manage SLAs on both the sides towards customers and providers, and to orchestrate service delivery.

Note that each aggregator a can be selected by multiple customers under the limit of a maximum resource capacity L_a that constrains the total complexity of served workflows, each service can be delivered by multiple providers with equivalent functionality but different quality, and each provider is available-for and can be selected-by multiple aggregators. Without loss of generality, the model assumes that each customer requires a single workflow and each provider delivers a single service. Multiplicity can be accommodated in the framework by introducing multiple virtual customers and providers, as also in [76]. Finally, for simplicity but open to relaxation, each workflow is requested by a single customer.

On the one hand, the SLO in the agreement among an aggregator and the provider of each service s is expressed as a CDF distribution $F_Y^s(\cdot)$ originated from the auction

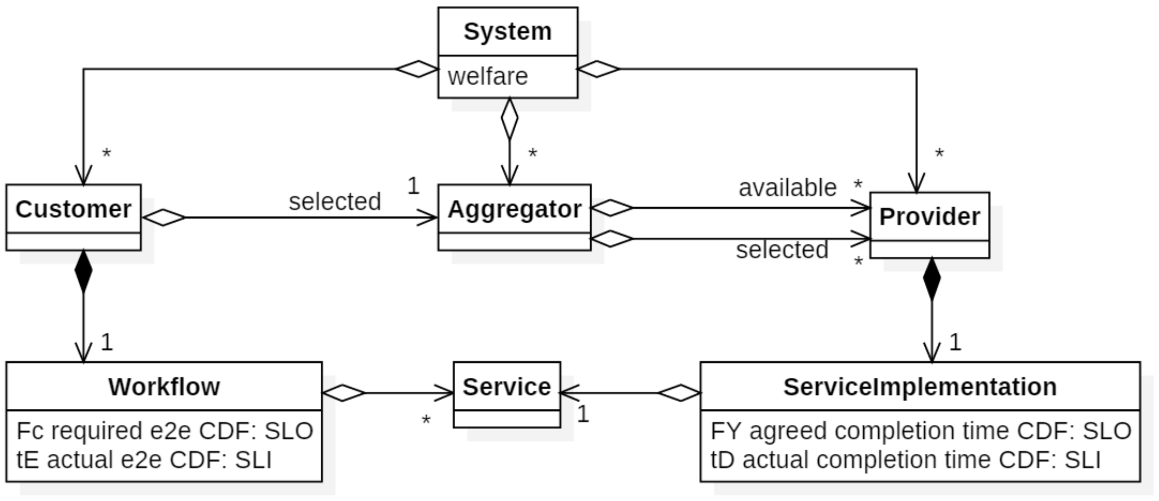


Figure 4.1: Participants in the system model.

mechanism presented in Section 4.2, and the SLI of each service is the statistics of actual completion times t_D. According to this, the aggregator will pay the provider selected for each service S a reward expressed as

$$\mathcal{U}_{a,s} = (1 - F_Y^s(t_D)), \tag{4.1}$$

Note that in so doing, services are assumed to have a uniform cost. This restriction is made for simplicity, without loss of generality, as the treatment can be easily extended to encompass costs of services dependent on their type or position in the topology of the workflow.

On the other hand, each customer expresses its required SLO as a CDF F_c on the e2e workflow completion time t_E, and the SLI supplied by the selected aggregator is the statistics of actual e2e completion times measured in repeated executions of the workflow. For the aggregator, the workflow execution subtends an outlay cost for services orchestration and for providers reward. To cover this cost, here denoted by R_c, the customer pays the aggregator a reward b_c, with $b_c > R_c$, that accounts for the functional complexity of the supplied workflow, multiplied by a factor that accounts for the delivered QoS, here expressed in terms of the e2e time provisioned with respect to the SLO posed by customer c. According to this, the aggregator utility is expressed as

$$\mathcal{U}_{c,a} = (1 - F_c(t_E) + \Theta) \cdot (b_c - R_c), \tag{4.2}$$

where t_E is the actual e2e workflow time, and $\Theta > 0$ is an additive corrective factor to induce positive provider rewards, whose value is investigated in Section 4.4.

4.1.2 Problem formulation

The objective of this paper is the maximization of a measure of system welfare accounting for joint rewards of providers and aggregators. This can be expressed as optimization problem as:

$$\max_{\mathbf{\Gamma},\mathbf{\Delta}} \sum_{c=1}^{C} \sum_{a=1}^{A} \mathcal{U}_{c,a} \gamma_{c,a} + \sum_{s=1}^{S} \sum_{p=1}^{P} \mathcal{R}_{s,p} \delta_{s,p}, \tag{4.3}$$

$$s.t. \quad \sum_{p=1}^{P} \delta_{s,p} \leqslant 1, \quad \forall s \in \mathcal{S} \tag{4.4}$$

$$\sum_{a=1}^{A} \gamma_{c,a} \leqslant 1, \quad \forall c \in \mathcal{C} \tag{4.5}$$

$$\sum_{c=1}^{C} R_c \gamma_{c,a} \leqslant L_a, \quad \forall a \in \mathcal{A} \tag{4.6}$$

where: $\mathbf{\Gamma}$ is the assignment matrix with $\gamma_{a,c}$ equal to 1 when aggregator a is selected to serve customer c, and 0 otherwise. And, similarly, $\mathbf{\Delta}$ is selection matrix with $\delta_{s,p}$ equal to 1 when service s is outsourced to provider p and 0 otherwise. $\mathcal{R}_{s,p}$ is the reward of provider p in completing service s, and is accurately described in Section Section 4.2. Note that, in the formulation, when the same service belongs to more than one workflow, service occurrences are handled as different services, and a number of rows equal to the number of service occurrences is added to $\mathbf{\Delta}$.

Constraint of Eq. (4.4) imposes that each service s can be outsourced to one and only one provider. Similarly, constraint of Eq. (4.5) points out that each customer is assigned to at most one aggregator. Eq. (4.6) requires that resources allocated on the aggregator a cannot exceed its maximum resource capacity L_a.

Eqs. (4.3) and (4.6) formulate an NP-hard problem, as a simplified version can be reduced to the 0-1 knapsack problem [77]. More in detail, a special case of the formulated problem is considered, where the maximization of the providers reward is neglected and $A = 1$. In this case, the objective function is $\max \sum_{c=1}^{C} \mathcal{U}^1 \gamma_{c,1}$, subject to: $\sum_{c=1}^{C} R_c \gamma_{c,1} \leqslant L_1$. Since $\gamma_{c,1} \in \{0,1\}$, by mapping L_1 in the knapsack capacity parameter, $\mathcal{U}^1$ and R_c in the weight and the volume of the generic item, the formulation of the 0-1 knapsack problem [77] is obtained. To avoid the complexity of exact solution, this paper proposes a suboptimal scheme consisting of a matching game with externalities integrated with an auction mechanism to jointly assign customers to aggregators and outsource services to providers, and refers to Section 4.4 for experimental evaluation of suboptimality.

4.1.3 Proposed mechanism

Collaborations among participants, illustrated in Fig. 4.2, involve a twofold mechanism: on the one hand, association among requested workflows and aggregators relies on a *matching game*, presented in Section 4.3, that aims at maximizing a collective utility function that promotes efficient usage of system resources while matching SLOs requested by customers with those that aggregators can deliver. On the other hand, for each aggregator, outsourcing to providers of services requested for the implementation of customers workflows relies on a *stochastic auction* mechanism, detailed in Section 4.2, which aims at maximizing predictability in the ability to fulfill of customers SLOs.

The two mechanisms are combined through the following interaction steps:

1. each customer c requires repeated execution of a workflow c with a SLO specified as a CDF $F_c(\cdot)$ of the expected end-to-end workflow completion time, and it expresses a preference to be assigned to aggregators with higher available capacity, as outlined in Section 4.3;

2. any aggregator a aiming at handling the customer workflow c announces an auction among its providers $\mathcal{P}_a$ to outsource each requested service, as described in Section 4.2;

3. each provider $p \in \mathcal{P}_a$ of a requested service answers with a bid expressing the CDF $F_s(\cdot)$ of the completion time that it will be able to deliver, and each service is assigned to the provider proposing the best CDF, as detailed in Section 4.2;

4. upon completion of the auctions, each aggregator is able to safely predict the end-to-end completion time that it can deliver to each customer, and it can thus express a preference on which customer it can better serve by maximizing the fit to the requested SLO;

5. preferences expressed by customers and aggregators are finally matched through the game detailed in Section 4.3.

4.2 Auction-based service selection

In this section, it is provided a service selection approach based on the Vickrey-Clarke-Groves auction (Section 4.2.1). Here, for a specific service of a workflow, competing providers offer different time of execution. The auction mechanism select the best winner and requires it to finish the execution of the service in a time equal to that of the second best bid. Since providers bids are defined in terms of a PDFs, to provide an order of preference among the offered PDFs, the dominance measure

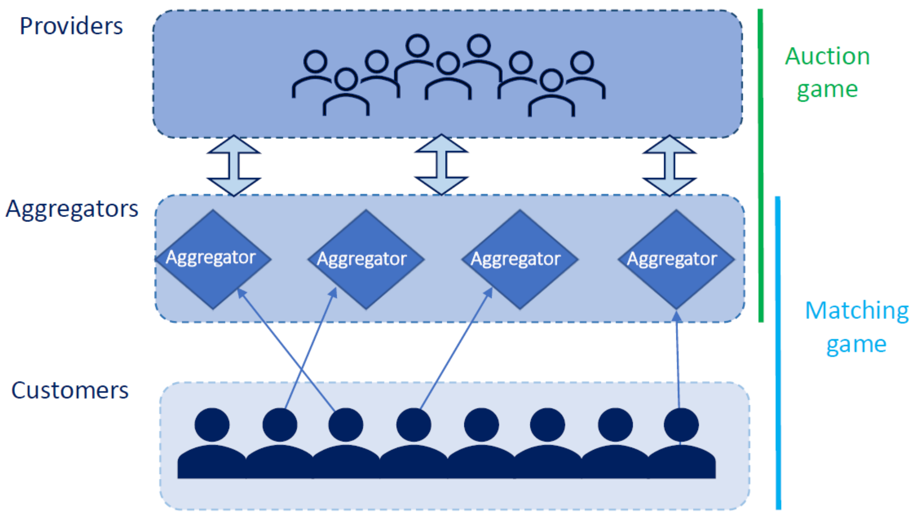


Figure 4.2: An illustration of the proposed framework: a matching game with externalities integrated with an auction mechanism to jointly assign customers to aggregators and outsource services to providers.

between random variables is exploited. In particular, because this measure is not transitive for all classes of distributions, a demonstration of transitivity on the class of uniform distributions is provided. Finally, a proof is provided for the proposed dominance measure that cheating on the times offered does not produce benefits to providers (Section 4.2.3).

4.2.1 Vickrey-Clarke-Groves auction

VCG auction is a Nobel-prize winning framework which provides mathematically tractable solutions to the assignment problem, aiming at simultaneously considering the perspectives of opposite parts involved in the bargaining process. VCG defines the second price auction mechanism, in which the auction winner is the bidder offering the best bid. Then, the winner bidder does not pay its bid, but instead pays the amount bid by the second-best bidder. Considering a generic provider p, if the associated bid is not the best bid, the provider p does not obtain the service outsourcing and does not have to pay anythingprov providers bids correspond to their own CDFs over the service completion time. In fact, each provider offers the CDF of the service time $F_s(\cdot)$ as detailed in Section Section 4.1.3 to compete for service delivery. Then, the auctioneer collects the bids from the antagonist providers and elects the winner on the basis of the CDFs received. The mechanism subtends an order among CDFs, for which the *Pairwise-comparison dominance* relation is considered:

Definition 6. *Let X, Y be random variables with pdf $f_X(t)$, $f_Y(t)$, and CDFs $F_X(t)$, $F_Y(t)$, respectively. X and Y are in relation of* Pairwise-comparison dominance, *denoted by $X \preceq Y$, when $Prob\{X \leqslant Y\} \geqslant \frac{1}{2}$, i.e. when*

$$\int_{t=0}^{\infty} (1 - F_Y(t)) \cdot f_X(t)dt \geqslant \frac{1}{2} \tag{4.7}$$

The relation $\preceq$ is not an order, as it is *not antisimmetric*, i.e. there exist two random variables $X \neq Y$ such that

$$\begin{aligned} &\int_{t=0}^{\infty} (1-F_Y(t)) \cdot f_X(t)dt \\ &\qquad = \int_{t=0}^{\infty} (1 - F_X(t)) \cdot f_Y(t)dt = \frac{1}{2} \end{aligned} \tag{4.8}$$

For example, this occurs for any two random variables X,Y supported over $[0, 1]$ with pdfs $f_X(t) = 1$ and $f_Y(t) = 6 \cdot x \cdot (1 - x)$. For the purposes of our auction mechanism, the relation $\preceq$ should satisfy all the other properties of a *total* order, i.e. *transitivity*, *reflexivity*, and *totality*. The satisfaction of reflexivity and totality does not depends on the stochastic characterization of the considered random variables. On the contrary, transitivity must be demonstrated for different classes of distributions. For the aims of this work, in Section 4.2.2 we demonstrate transitivity of the relation on the class of uniform distributions.

Once the aggregator receives providers CDFs, these are ranked in accordance with Definition Definition 6. Then, each service in the workflow is assigned to a bid attaining the minimum, i.e. any bid $X_\star$ such that $X_\star \preceq X_i$ for any other received bid. Note that multiple bids might attain the same minimum, in this case the tie can be resolved according to any determinization or even in a randomized manner without affecting all the subsequent treatment. A reasonable principle can be that in case of parity, the bid with the lower coefficient of variation be preferred so as to promote predictability, which however can be not sufficient to resolve the tie.

The algorithm behavior can be summarized through the following steps:

1. The aggregator performs service auctioning to outsource service exploitation;

2. Each provider competes to win service delivery by proposing the CDF $F_s(\cdot)$ of the service completion time as bid;

3. The aggregator receives bids from providers, then selects the provider $p_\star$ with the best bid, i.e., the bid $X_\star$ such that $X_\star \preceq X_i$ for any other received bid, on the basis of Definition 6.

4. The winner bidder provides service to the customer and the auctioneer computes the corresponding utilities considering the actual duration of the service

and the second best bid, on the basis of the pairwise-comparison dominance relation. In fact, when the service is completed by the winner $p_\star$, say at time t_D, the winner receives a reward $\mathcal{R}^{p_\star}$ equal to the probability that t_D is better (i.e. earlier) than the time that would have been provided by the second best bid, i.e. $\mathcal{R}^{p_\star}(t_D) := 1 - F_Y(t_D)$, where $F_Y(t)$ is the CDF of the second best bid.

5. The aggregator computes the CDF of the actual e2e time t_E, i.e., $F(t_E)$, on the basis of the service CDFs received by providers.

For each aggregator a and customer c, the proposed VCG auction is repeated for a number of times equal to the number of services composing the workflow of c.

4.2.2 Pairwise-comparison dominance transitivity for uniform distributions

In this subsection, it is firstly provided a lemma showing that two uniform random variables X and Y are in pairwise-comparison dominance relation when $E[X] \leqslant E[Y]$. Then, the lemma is exploited to demonstrate transitivity on the pairwise-comparison dominance relation for uniform random variables.

Lemma 5 (Pairwise-comparison dominance among uniform random variables)**.** *Let X and Y be two random variables with support $[a, b]$ and $[c, d]$ respectively. X (or Y) is in pairwise-comparison dominance with Y (or X), denoted by $X \preceq Y$ (or $Y \preceq X$), i.e. $Prob\{X \leqslant Y\} \geqslant \frac{1}{2}$ (or $Prob\{Y \leqslant X\} \geqslant \frac{1}{2}$), if and only if $a + b \leqslant c + d$ (or $c + d \leqslant a + b$).*

Proof of Lemma 5. There are 6 possible orders of the support bounds of the considered random variables: *abcd* and *cdab* (disjoint supports), *acbd* and *cadb* (overlapping supports), and *acdb* and *cabd* (one support includes the other).

- In the case of *abcd*, since supports are disjoint and X support precedes Y support, then $Prob\{X \leqslant Y\} = 1$, i.e. $X \preceq Y$. Then, $a + b < c + d$ by construction, and thesis is proved. The case of *cdab* is analogous but specular.

- In the case of *acbd*, the vector $< X, Y >$ determines a rectangle on the coordinate plane, representing the joint support of the two random variables. The bisector $x = y$ cut the rectangle in two parts: one identifies pairs (x, y) for which $x < y$ (above the bisector) and the other pairs for which $y < x$ (below the bisector). As shown in Fig. 4.3a, given the order of the support bounds, the area of the rectangle above the bisector is grater than the area below. In fact, the area above the bisector contains completely the area below the bisector (the pink square in the figure, is divided in two equal parts by the bisector, and above the bisector the area of the joint support contains completely the half square, which coincides with the area of the support below the bisector). Hence, $X \preceq Y$. Since $a < c$ and $b < d$, it follows that $a + b < c + d$, from which the thesis is proved. The case of *cadb* is analogous but specular.

- In the case of *acdb*, the vector $< X, Y >$ determines a rectangle on the coordinate plane, representing the joint support of the two random variables. The bisector $x = y$ cut the rectangle in two parts: one identifies pairs (x, y) for which $x < y$ (above the bisector) and the other pairs for which $y < x$ (below the bisector). As shown in Fig. 4.3b, given the order of the support bounds, the area of the rectangle above the bisector is grater than the area below, only if $c - a > d - b$. In fact, the areas identified by the bisector both contains the halves of the pink square cut by the bisector; then, since the height of the support is the same for the considered areas, the remaining part of the support above the bisector is greater then the part below the bisector, only if $c - a > d - b$. In this case, $X \preceq Y$. From $c - a > d - b$, it follows that $a + b < c + d$, from which the thesis is proved. The case of *cabd* is analogous but specular.

The thesis is then proved for all the orders of support bounds. □

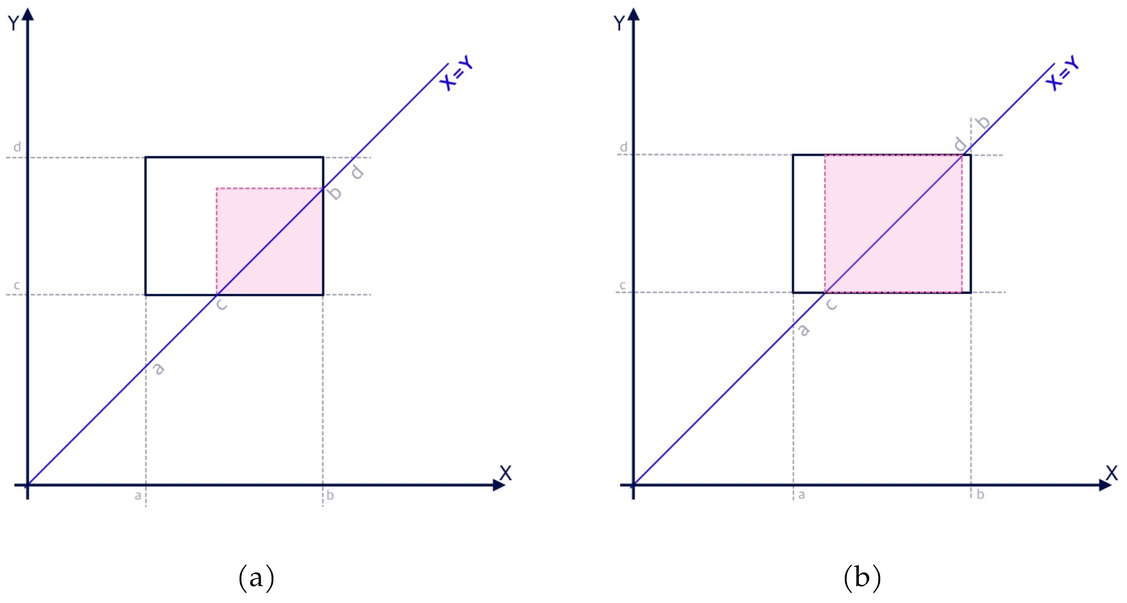


Figure 4.3: Graphical representation of the joint support of two uniform random variables, when support bounds order are *acbd* (a) and *acdb* (b).

Corollary 1. *Let X and Y be two random variables with support $[a, b]$ and $[c, d]$ respectively. X (or Y) is in pairwise-comparison dominance with Y (or X), denoted by $X \preceq Y$ (or*

$Y \preceq X$), i.e. $Prob\{X \leqslant Y\} \geqslant \frac{1}{2}$ (or $Prob\{Y \leqslant X\} \geqslant \frac{1}{2}$), if and only if $E[X] \leqslant E[Y]$ (or $E[Y] \leqslant E[X]$).

Proof of Corollary 1. The condition of the corollary $E[X] \leqslant E[Y]$ is obtained by the condition of the Lemma 5, $a + b < c + d$. In fact $a + b < c + d \rightarrow 2E[X] \leqslant 2E[Y] \rightarrow E[X] \leqslant E[Y]$. □

Theorem 1 (Pairwise-comparison dominance transitivity for uniform random variables)**.** *Let X, Y and Z be three uniform random variables with supports $[a, b]$, $[c, d]$ and $[e, r]$ respectively, so that $X \preceq Y$ and $Y \preceq Z$, then $X \preceq Z$.*

Proof of Theorem 1. From Corollary 1, the relations $X \preceq Y$ and $Y \preceq Z$ imply that $E[X] < E[Y]$ and $E[Y] < E[Z]$, respectively. Hence, $E[X] < E[Y] < E[Z]$, and from Corollary 1, $X \preceq Z$. □

4.2.3 VCG consequences and cheating strategy

In this subsection, the consequences of this mechanism when the assigned service is repeated for a number of times sufficiently large so as to let emerge a stable statistics of the observed QoS provided by the winning bidder $p_\star$ are evaluated. In this perspective, the expected reward for the winner is:

$$E[\mathcal{R}^{p_\star}] = \int_{t=0}^{\infty} (1 - F_Y(t)) \cdot f_{\tilde{X}_\star}(t) dt, \tag{4.9}$$

where $F_Y(t)$ is the CDF of the second best bid and $f_{\tilde{X}_\star}$ is the actual pdf provided by the winning bidder, i. e., the QoS impemented. Based on the order relation existing between $\tilde{X}_\star$ and $X_\star$, the following cases can occur:

- If the QoS provided by the winning bidder actually satisfies the bid, i.e. $f_{\tilde{X}_\star}(t) = f_{X_\star}(t)$, then $E[\mathcal{R}^{p_\star}] \in [\frac{1}{2}, 1]$, with a value that is as higher as the probability that the second bid is lower than the best one. Note that by definition the reward $\mathcal{R}^{p_\star}$ is the complement to 1 of a CDF and thus belongs to $[0, 1]$. Moreover, by design of the auction mechanism, $X \preceq Y$, and thus $E[\mathcal{R}^{p_\star}] \geqslant \frac{1}{2}$.

- On the one hand, the minimum reward $\frac{1}{2}$ occurs if the second best bid Y is equivalent to the best bid $X_\star$, i.e. if $X_\star \preceq Y \wedge Y \preceq X_\star$ as $\int_{t=0}^{\infty} (1 - F_Y(t)) \cdot f_Y(t) \cdot dt = \frac{1}{2}$, which implies

$$\begin{aligned} &\int_{t=0}^{\infty} (1 - F_Y(t)) \cdot f_{X_\star}(t) dt \\ &\quad = \int_{t=0}^{\infty} (1 - F_{X_\star}(t)) \cdot f_Y(t) \cdot dt = \frac{1}{2}. \end{aligned} \tag{4.10}$$

- On the other hand, the maximum expected reward 1 occurs when the second best bid Y is deterministically later than the best bid X, which occurs when $F_Y(t) > 0 \rightarrow F_{X_\star}(t) = 1$, i.e. the support of $X_\star$ is *before* [78] the support of Y.

- If the actual QoS provided by the winner is better than its bid, i.e. if $f_{\tilde{X}_\star}(t) \preceq f_{X_\star}(t)$, then the winner improves its advantage with respect to the second best bid Y, which will result in a higher value of $E[\mathcal{R}^{p_\star}]$, but always under the maximum limit of 1.

- Conversely, the expected reward $E[\mathcal{R}^{p_\star}]$ falls under $\frac{1}{2}$ if the actual QoS $f_{\tilde{X}_\star}(t)$ degrades up not to dominate the second best bid Y, i.e.

$$\int_{t=0}^{\infty}(1 - F_Y(t)) \cdot f_{X_\star}(t)dt \leqslant \int_{t=0}^{\infty}(1 - F_Y(t)) \cdot f_Y(t) \cdot dt. \tag{4.11}$$

- In the limit case that the actual SLI $\tilde{X}$ of the winner is deterministically later than that of the second best bid Y, i.e. if Y is *before* $\tilde{X}$, then the reward of the winner becomes 0. In any case, the reward cannot become negative, being defined as a probability.

The VCG mechanism rules out the case of a rational provider pursuing a cheating strategy. In fact, let assume that a provider may attempt to lie about the CDF, proposing an improved CDF which describes an unrealistic best-case service completion scenario. Let $p_\star$ and $F_Y(t)$ be the winner provider and the CDF of the second best bid, respectively. Supposing that provider $p_\star$ completes service delivery with a deterministic delay Δ, the corresponding revenue is expressed by:

$$\begin{aligned} \mathcal{R}_\Delta^{p_\star}(t) &= \int_{x=0}^{t} \big(1 - F_Y(x)\big) f_{\tilde{X}_\star}(x - \Delta)dx \\ &= \int_{x=0}^{t} f_{\tilde{X}_\star}(x - \Delta)dx - \int_{x=0}^{t} F_Y(x) f_{\tilde{X}_\star}(x - \Delta)dx. \end{aligned} \tag{4.12}$$

After some algebraic manipulation, the following equation is obtained:

$$\begin{aligned} \mathcal{R}_\Delta^{p_\star}(t) &= \int_{x=0}^{t} \big(1 - F_Y(x)\big) f_{\tilde{X}_\star}(x - \Delta)dx \\ &= \int_{x=0}^{t-\Delta} \big(1 - F_Y(x + \Delta)\big) f_{\tilde{X}_\star}(x)dx. \end{aligned} \tag{4.13}$$

From Eq. (4.14), due to the fact that F_Y is monotonic increasing, it is evident that the provider i does not improve the revenue, since the difference between integrand functions is greater than zero, i.e.,

$$\big(1 - F_Y(x)\big) - \big(1 - F_Y(x + \Delta)\big) = F_Y(x + \Delta) - F_Y(x) \geqslant 0. \tag{4.14}$$

Differently, when all the winning providers realize a SLI that satisfies the bid, i.e., $f_{\tilde{X}_*}(t) = f_{X_\star}(t)$, $E[\mathcal{R}^p] \in [\frac{1}{2}, 1]$, and the aggregator receives a non-negative reward. This is a direct consequence of Eq. (4.2), where $b_c - R_c > 0$ and $\Theta > 0$.

By combination of the VCG mechanism with Eq. (4.2), parameter Θ rules the degree of compliance requested to the aggregator, which in turn determines the degree of compliance that the aggregator must request to its providers:

Behavioural consequence (C1): *The value of Θ represents the critical value of the strategy proposed. Necessarily, $\Theta \in [\frac{1}{2}, 1]$, but an over-pessimistic choice about Θ, i.e., $\Theta = \frac{1}{2}$, may penalize the aggregator reward, since it may not cover the cost employed by aggregator to pay providers when service is completed later than the SLO agreed. Conversely, $\Theta = 1$ represents an over-optimistic case that takes away responsibility in selecting reliable providers from aggregators. At a second level of analysis, the value assigned to Θ rules the level of competitiveness imposed by aggregators to providers, controlling race conditions among providers imposed by aggregators, and consequently the profit margin of aggregators. When $\Theta = \frac{1}{2}$, the aggregator is required to maintain a high level of competitiveness among its providers for each service so that the second bid CDF be close to the best one, whereas $\Theta = 1$ implies soft competition conditions.*

4.3 Matching-based customers-aggregators assignment

Matching Theory (MT) represents a powerful technique to match together elements belonging to two opposite sets, considering the satisfaction of each element in being associated to each element of the opposite set and vice-versa, reaching a valuable trade-off between the preferences drafted by elements. Moreover, the MT has a naturally distributed nature, since it involves exclusively local utility metrics to build the individual preferences. Consequently, MT algorithms represent a suitable strategy in distributed scenarios, such as the environment proposed in this paper. In our case the matching game is formulated between the set of customers $\mathcal{C}$, and that of aggregators $\mathcal{A}$, in order to establish relations reciprocally advantageous for all the players belonging to $\mathcal{C}$ and $\mathcal{A}$ [79], taking into account their preferences. Preference relations describe the level of interest of each element of a set in being matched with each element of the opposite set. Note that, on the basis of the workflow required, each customer c can be supported by a set of aggregators $\mathcal{A}_c$. Furthermore, defining with $\mathcal{L}_a$ the amount of resources available on aggregator a, i.e.,

$$\mathcal{L}_a = L_a - \sum_{c \in \mathcal{C} \setminus \{c\}} R_c \gamma_{c,a}, \tag{4.15}$$

each customer c can be supported by aggregators belonging to $\mathcal{A}_c$ having $\mathcal{L}_a \geqslant R_c$. As a consequence, the size of preference lists built by customers may change over algorithm execution, producing incomplete lists and lying in the class of the matching games with incomplete preference lists [80]. In the following, the utility functions involved in the preference lists construction process are defined.

4.3.1 Customers preference list

For each customer $c \in \mathcal{C}$, the utility function $H_c(a)$ of c in being matched with the aggregator $a \in \mathcal{A}_c$ is given by

$$H_c(a) = \mathcal{L}_a - R_c. \tag{4.16}$$

From Eq. (4.16) follows that the most preferred aggregator $a^\star$ is given by

$$a_c^\star = \arg\max_{a \in \mathcal{A}_c} H_c(a). \tag{4.17}$$

Eq. (4.17) expresses that the most preferred $a_c^\star$ is the aggregator having the greatest number of available resources. In this reference, it is important to note that as the algorithm execution proceeds, the number of allocated customers on aggregators grows. Hence, after each algorithm iteration, the unassigned customers need to re-build their preference lists, to properly catch the actual resource availability of the aggregators, which decreases with the progress of the algorithm, due to the presence of customers previously assigned to the same aggregator.

4.3.2 Aggregators preference list

From the other hand, the preference list of each aggregator a is built preferring the customer $c^\star$ which maximizes the mean reward of a. In order to select the proper customer, the auction mechanism presented in Section 4.2 is run for each service required by customer. Defining the set of customers proposing to the aggregator a as $\mathcal{C}_a$, the corresponding aggregator preference list $E_a(c)$ is built ranking in descending order the following metric

$$E_a(c) = (b_c - R_c) \int_{x=0}^{\infty} f_{\tilde{X}_\star}(x)(1 - F_c(x))dx, \tag{4.18}$$

representing the mean utility where the term Θ is neglected since a constant value. Therefore, the most preferred $c_a^\star$ is given by

$$c_a^\star = \arg\max_{c \in \mathcal{C}_a} E_a(c). \tag{4.19}$$

In this paper a modified version of the Gale-Shapley algorithm (GSA) [81, 79] is applied.
Summarizing, the algorithm acts as follows

1. each unassigned customer $c \in \mathcal{C}$ builds its preferences list on aggregators accordingly to Eq. (4.16);

2. each aggregator a, receiving a set of proposals $\mathcal{C}_a \subseteq \mathcal{C}$, builds its preference list performing, for each customer, the VCG auction;

3. each aggregator that receives more than one proposal, i.e., $|\mathcal{C}_a| > 0$, selects the most favorite customer $c_a^\star$ in accordance with Eq. (4.19), and accepts to serve $c_a^\star$ among those received, rejecting the others;

4. each unassigned customer $c \in \mathcal{C}$ deletes from the possible aggregators those having an available resource capacity lower than R_c;

5. repeat $1) - 4)$ until all the customers have been matched with one aggregator.

Note that, in our problem, the preferences list of each customer depends on the assignments of the other players. This corresponds to the case of a game belonging to the class of matching problems in which the preference lists change based on the choices performed during the algorithm execution, typically referred as matching games with *externalities* [80]. Therefore, due to the existing interdependencies and relations among the players' preferences lists, these lists have to be updated after each algorithm assignment.

The matching algorithm developed is that it terminates in a finite number of iterations. In order to perform the termination analysis, the worst case scenario is considered, and the following assumptions are made: i) there is only one aggregator a; ii) during each iteration only one customer is allocated on the aggregator a; iii) $L_a = \infty$. In reference to steps 1)-5) of the algorithm previously introduced, considering the working hypothesis i)-iii), the algorithm terminates in a number of iteration μ equal to the number of customers, i.e., in $\mu = |\mathcal{C}|$ steps. Removing hypothesis i)-iii), the resulting scenario is not worst case, and the algorithm terminates in a number of steps $\mu \leqslant |\mathcal{C}|$.

Algorithm design has the following implications, in terms of customer fully-rational perspective, and provider alternation in winning auction:

Behavioural consequence (C2): *Selfish customer behaviors are allowed. This is the practical consequence for not having bounded above the term b_c. As a matter of principle, each Costumer may select an arbitrary large $b_c > R_c$ in order to be assigned to its most preferred aggregator.*

Behavioural consequence (C3): *The market proposed does not safeguard alternation in service adjudication among providers. The possibility of having providers with potentially winning bids linked to aggregators unavailable to host customers is not excluded. This case stems from the condition in which the available resources on the aggregator, i. e., Eq.* (4.15), *are insufficient to accept the customer requiring the minimum cost. Furthermore, the case where a provider offers valuable bids, but it is never the best, may occur.*

Future work may consider mechanisms to mitigate the non-alternation due to free competition in provider market. For example, metrics focusing on both bids age and quality may be introduced to favor alternation among providers. Similarly, to overcome the resource unavailability issue, *gentrification*-like mechanisms [82, 83], i.e., the process whereby providers with high-quality bids leaves the aggregator with unavailable resources to freely compete with providers linked to other aggregators, may be investigated.

4.3.3 Stability analysis

As opposed to classical matching problems, for matching games with externalities there not exists any algorithm that guarantees a stable outcome. This challenging aspect is due to the presence of dependencies among the players' preferences [79]. In order to discuss the stability of the matching stemmed by the proposed framework, the following S2ES stability definition is given and represents a modified version of the one originally proposed in [84].

Definition 7. *Let $\mathcal{Z}$ be the outcome matching of the algorithm developed. Let $\mathcal{Z}(c)$ be the aggregator matched with the customer c in the matching $\mathcal{Z}$. The outcome matching $\mathcal{Z}$ is a S2ES matching if there not exists a pair of customers (c_1, c_2) s.t.:*

1. $H_{c_1}(\mathcal{Z}(c_2)) \geqslant H_{c_1}(\mathcal{Z}(c_1))$ *and*
2. $H_{c_2}(\mathcal{Z}(c_1)) \geqslant H_{c_2}(\mathcal{Z}(c_2))$ *and*
3. $E_{\mathcal{Z}(c_1)}(c_2) \geqslant E_{\mathcal{Z}(c_1)}(c_1)$ *and*
4. $E_{\mathcal{Z}(c_2)}(c_1) \geqslant E_{\mathcal{Z}(c_2)}(c_2)$ *and*
5. $\exists \psi \in \{c_1, c_2\}$ *s.t. at least one of the conditions* $1) - 2)$ *is strictly verified and*
6. $\exists \phi \in \{\mathcal{Z}(c_1), \mathcal{Z}(c_2)\}$ *s.t. at least one of the conditions* $3) - 4)$ *is strictly verified.*

Definition 7 means that a swap is allowed if an improvement to at least one customer and one aggregator involved is provided, and the rest of players participating in the swap do not worsen. Aiming at proving the stability of the proposed matching algorithm, the assumption of the existence of a pair of customers (c_1, c_2), for which the conditions $1) - 2)$ of Definition 7 results to be true, is done. Furthermore, let c_1 and c_2 be s.t. $\mathcal{Z}(c_1) = a_1$ and $\mathcal{Z}(c_2) = a_2$, respectively. This necessarily means that

$$H_{c_1}(a_2) \geqslant H_{c_1}(a_1), \tag{4.20}$$

$$H_{c_2}(a_1) \geqslant H_{c_2}(a_2). \tag{4.21}$$

In reference to the satisfaction of condition 5) of Definition 7 by Eqs. (4.20) and (4.21), since the proposed assignment policy does not provide any discard strategy, the amount of available resources on aggregators cannot increases during the assignment process. Therefore, the preference list of each matched customer cannot change after its assignment. Since, after the assignment of the generic customer on the aggregator, the amount of available resources on that aggregator cannot increase, then at the most $H_{c_1}(a_1) = H_{c_1}(a_2)$ and $H_{c_2}(a_2) = H_{c_2}(a_1)$. As a consequence, condition 5) is not verified. Therefore, even if condition 6) is verified, the proposed matching game reaches a configuration satisfying the S2ES property.

Behavioural consequence (C4): *The stability of the game formulated underlines that the objective pursued by the social welfare-oriented maximization coincides with the objective of an aggregator that arbitrarily deviates its partner selection to pursue its individual interests. The result holds under the assumption that the aggregator is modeled as altruistic, in the sense presented in [85]. Accordingly to [85], an aggregator is altruistic if it keeps its partner to avoid customer worsening, even if the rational choice is to deviate the match. Differently, a rational aggregator [85] obeys to a selfish behavioural model, changing partner if doing so it increases its reward. The stability discussion of this case can be addressed relaxing Definition 7, allowing that only one between conditions 5) and 6) is satisfied. Nevertheless, the theoretical analysis of this behavioural model is out of the scope of the paper in its current form, which investigates this case from an experimental perspective in Section 4.4.*

Algorithm 7: Matching assignment strategy

```
input : set of customers $C$, set of aggregators $A$
output: assignment matrix $\mathbf{\Gamma}$
1 while $\exists a \in A$ with available resources and unmatched customers do
2     compute customers preference list;
3     compute aggregator preference lists invoking the VCG auction;
4     foreach unmatched customer $c$ do
5         propose assignment to the most favorite aggregator;
6     foreach aggregator $a$ receiving proposals do
7         select the most favorite customer $c^\star$;
8         $\mathbf{\Gamma}(c^\star, a) = 1$
9 return $\mathbf{\Gamma}$
```

4.4 Experimentation

In this section, the applicability and the effectiveness of the proposed framework are tested with the aim of answering the following research questions:

Algorithm 8: VCG service selection strategy

```
input  : aggregator $a$, set of provider $P$, set of customers allocated on $a$
output: selection matrix $\mathbf{\Delta}$
while $\exists s$ unselected do
    foreach provider $p \in P$ do
        propose the CDF $F_s$;
    foreach received CDF do
        select the best CDF on the basis of the Pairwise-comparison dominance;
        select the corresponding provider $p_\star$;
        $\mathbf{\Delta}(s, p_\star) = 1$
return $\mathbf{\Delta}$
```

- **Q1.** How does the approach perform in terms of system welfare (i.e., welfare of aggregators and providers) in comparison to a strategy assuming full knowledge about the system?
- **Q2.** How does the system welfare vary when at least one provider is not able to satisfy the agreed SLO? How do the aggregator reward and the provider reward change?
- **Q3.** How does the aggregator reward vary when the critical value Θ and the level of competitiveness among the providers change?

4.4.1 Experimental setup

The approach is implemented in Java, using the SIRIO Library [57] of the ORIS tool [86] to represent and manage CDFs. The implementation exploits also the Eulero library [87] to compute the PDF f_{t_E} of the e2e time t_E of the workflow of each customer c (based on the CDF $F_{\tilde{X}_{\star,s}}$ of the actual completion time of each service s of the workflow), and consequently, based on Eq. (4.2), to derive the expected reward of aggregator a serving customer c as $E[\mathcal{U}_{c,a}] = \int_{t=0}^{\infty}(1 - F_c(t) + \Theta) \cdot (b_c - R_c) \cdot f_{t_E}(t)\, dt$. Similarly, the expected reward gained by provider p in supplying service s can be computed through Eq. (4.9) as $E[\mathcal{R}^{p_\star,s}] = \int_{t=0}^{\infty}(1 - F_Y(t)) \cdot f_{\tilde{X}_{\star,s}}(t)\, dt$, where $F_Y(t)$ is the CDF of the second best bid (according to the VCG auction) and $f_{\tilde{X}_{\star,s}}$ is the PDF of the actual completion time of service s (i.e., the actual pdf of the provider winning the auction banned to supply service s).

The considered 3 workflow topologies are made of 8 services composed through the sequence operator (*sequential* topology), the split-join operator (*parallel* topology), or a well-nested combination thereof (*mixed* topology). Since transitivity of the Pairwise-comparison dominance relation is proved on the class of uniform distributions, the CDF F_c representing the SLO expressed by each customer c, the CDF representing the bid expressed by each provider for each service (including the CDF F_s modeling the SLO agreed for each service s according to the VCG auction), and the CDF $\tilde{F}_s$ of the actual completion time of each service s are assumed to be Uniform CDFs, which also enables to easily control different working scenarios, facilitating the interpretability of results. The competitiveness among providers is controlled by choosing different support bounds, and different expected values, as illustrated in Section 4.2.2.

Each service of each workflow belongs to a different type. For each aggregator and each service (of each workflow) that the aggregator proposes to supply, two providers are generated as follows, where not otherwise specified. For the first provider, a uniform CDF is taken with support bounds $[a, b]$. In particular, to reduce the correlation between aggregator available resources and the performance of its providers, a is randomly drawn between 0 and 10, favoring lower or higher values depending on whether the aggregator has few or many resources available, respectively; b is then obtained as a plus a value randomly taken between 0 and 5. For the second provider, a uniform CDF is taken with support bounds $[c, d]$, where c is randomly drawn between a and $a + 2.5$, while d is randomly selected to guarantee that the bid is dominated by the bid of the first provider. Moreover, the SLO F_c expressed by each customer c is a Uniform CDF having support $[e, f]$ with e evaluated as the difference between expected value and the variance of the completion time of the workflow whose activities are distributed as Uniform distributions having support $[2.5, 7.5]$, and f evaluated to fit that expected value. In turn, $E[t_E]$ is obtained from the PDF f_{t_E}. Where not otherwise specified, f_{t_E} is computed assuming that the CDF $\tilde{F}_s$ of the actual completion time of each service s supplied by each provider p is equal to the bid expressed by p. Furthermore, to bound the dynamic of the problem and improve the readability of the results, the cost R_c of the request of each customer c is set to 1, the price b_c offered by each customer c is set to 20, and the capacity L_a of each aggregator a is uniformly selected over $[5, 20]$.

The experiments illustrated in the following subsections are performed using a single core of an Intel Xeon Gold 5120 CPU (2.20 GHz) equipped with 32 GB of RAM.

4.4.2 Analysis of the Price of Anarchy

A variant of the proposed approach is implemented, where full knowledge about the system is exploited to build the preference lists of customers, defining the utility of customer c in being matched with aggregator a as $H'_c(a) = \mathcal{L}_a(1 - F_c(t_E))$ rather than as in Eq. (4.16). To measure the performance degradation in adopting local metrics to build preference lists, the Price of Anarchy (PoA) [46, 45] is computed.

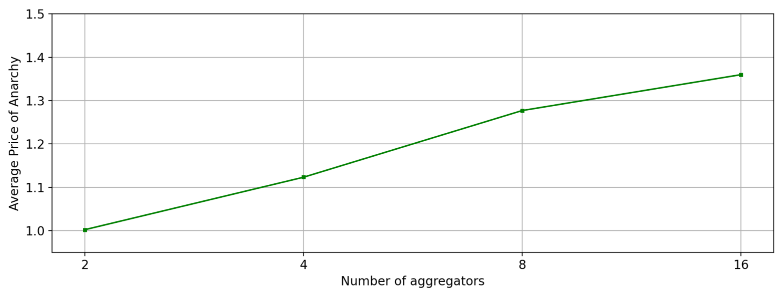


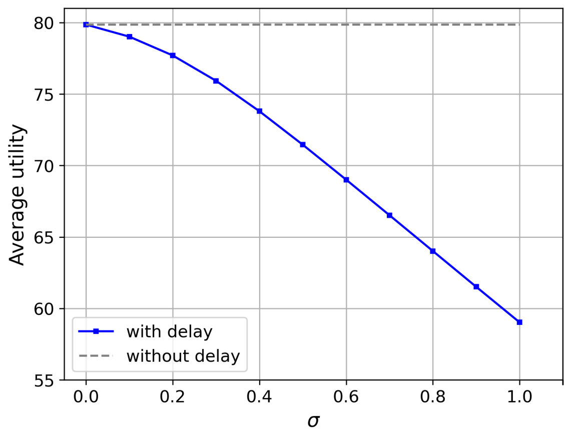
Figure 4.4: Average (over 200 runs) Price of Anarchy (i.e., ratio between the expected system welfare achieved by the full-knowledge approach and that achieved by our approach) as the number of aggregators increases.

In particular, the PoA is defined as the ratio between the expected system welfare achieved by the full-knowledge approach and that achieved by the proposed approach:

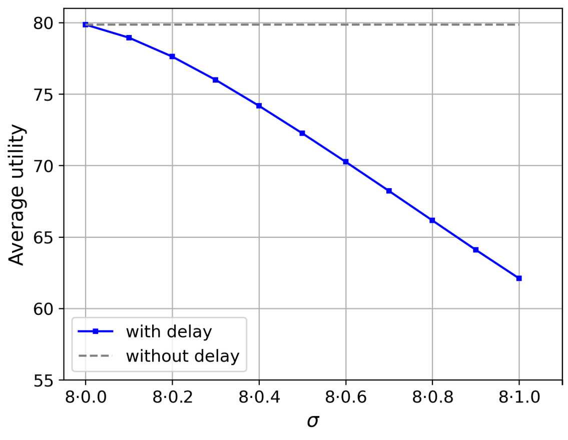
$$\mathrm{PoA} = \frac{\sum_{c=1}^{C}\sum_{a=1}^{A} E[\mathcal{U}'_{c,a}]\gamma'_{c,a} + \sum_{s=1}^{S}\sum_{p=1}^{P} E[\mathcal{R}'_{s,p}]\delta'_{s,p}}{\sum_{c=1}^{C}\sum_{a=1}^{A} E[\mathcal{U}_{c,a}]\gamma_{c,a} + \sum_{s=1}^{S}\sum_{p=1}^{P} E[\mathcal{R}_{s,p}]\delta_{s,p}} \tag{4.22}$$

where the quantities with prime denote those of the full-knowledge approach. Specifically, $C = 20$ customers, $A \in \{2, 4, 8, 16\}$ aggregators, and 1 workflow with random topology (among sequential, parallel, and mixed) for each customer are considered, while the remaining parameters are selected as illustrated in Section 4.4.1. For each number of aggregators, 200 instances of the problem are generated, deriving the average PoA, as shown in Fig. 4.4. Results show that the average PoA increases as the dynamics of the problem grows. This implies that, for a system that includes a low number of aggregators, the possible matches are reduced and almost equivalent with respect to the chosen matching strategy. This is also corroborated by the value 1 of the PoA for the case of 2 aggregators. As soon as the problem grows in complexity, the additional information introduced by the full-knowledge algorithm, i.e., $1 - F_c(t_E)$, is a recessive metrics for our problem compared with $\mathcal{L}_a$. This fact means that preference lists defined in Eq. (4.16), based on local and partial information about the system, are not able to properly catch the dynamics of the problem, when the combinatorial space grows in dimension, i.e., when the number of aggregators increases, making the assignment more challenging. However this is a consequence of having reduced correlation among aggregator resources and provider capabilities. In fact, for configurations built without driving the random generation, the dynamic of the system is complex to be caught and, despite the two matching strategies identify different matches, the average utilities converge to same value.

Hence, when it is known that different aggregators are served by differently per-



(a)



(b)

Figure 4.5: Average (over 200 runs) aggregator utility as the value of the slowing factor σ increases in the cases of: (a) balanced delay (i.e., support bounds are delayed by factor σ) and (b) unbalanced delay (i.e., support bounds of the actual completion time of a single service is delayed by σ).

forming providers, the full-knowledge strategy should be preferred. Otherwise, the proposed matching strategy can be exploited (and even preferred, since ordering lists with respect to a value, i.e. the available resources, is more efficient than ordering lists with respect to a function, i.e., $1 - F_c(t_E)$).

4.4.3 Analysis under unexpected delays

The impact of unexpected delays in service supply on the aggregator utility is evaluated. To this end, $A = 1$ aggregator and $C = 1$ customer requesting 1 workflow with sequential topology are considered, and two different kind of delays are tested: in the first, all services of the workflow are subject to a delay (*balanced delay*); in the second, only one of the services is subject to a delay (*unbalanced delay*). In the nominal case without delay, for each service s supplied by provider p, the CDF $\tilde{F}_s$ of the actual completion time is equal to the bid expressed by p. In the case of balanced delay, each service s supplied by provider p is delayed by increasing the support bounds (and so the expected value) by a factor $\sigma \in \{0.1, 0.2, \dots, 1\}$, e.g., if $\sigma = 0.1$, then the support bounds and the expected service time is increased by 10% with respect to the nominal case. Conversely, in the case of unbalanced delay, the variation is applied only to a single service s of the workflow, by increasing the support bounds and the expected value of a factor of $\tilde{F}_s$ by σ with $\sigma \in \{8 \cdot 0.1, 8 \cdot 0.2, \dots, 8 \cdot 1\}$, where 8 is the number of services of the workflow. The remaining parameters are selected as illustrated in Section 4.4.1.

For each value of σ, 200 instances of the problem are generated, deriving the average aggregator utility, as illustrated in Figs. 4.5a and 4.5b for the cases of balanced

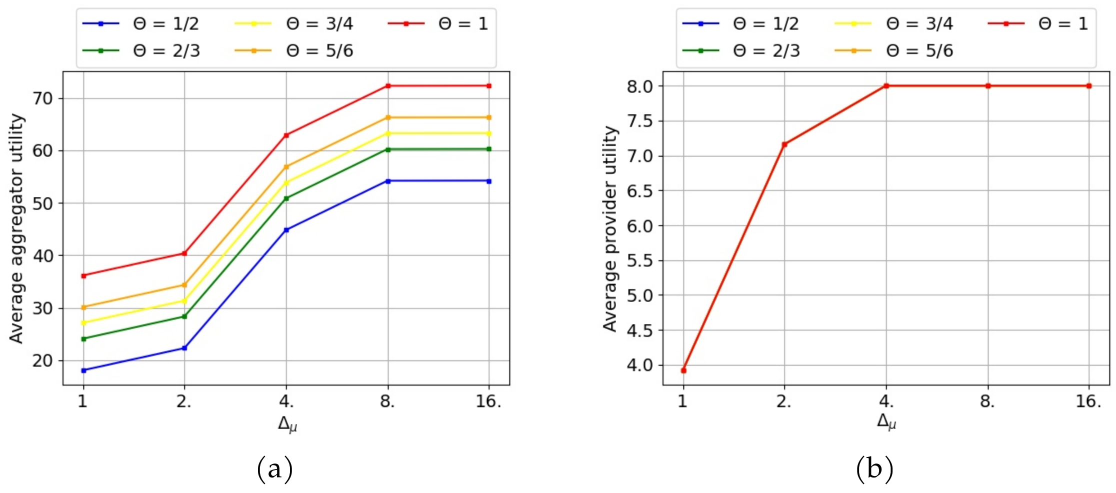


Figure 4.6: (a) Average (over 200 runs) aggregator utility as a function of Δ_μ (i.e., which rules the earliness of support bounds and expected value of the best bids with respect to the second best bids), for different values of the critical parameter Θ of the proposed approach. (b) Average (over 200 runs) provider utility as a function of Δ_μ.

and unbalanced delay, respectively. As expected, the average aggregator utility decreases as the slowing factor σ increases, since the expected actual completion time of the workflow increases. Note that, as σ increases, the reduction in the average utility is larger in the balanced case, when an expected delay of $100 \cdot \sigma\,\%$ affects each of the 8 services, rather than in the unbalanced case, when an expected delay of $8 \cdot 100 \cdot \sigma\,\%$ affects a single service, given that the expected delay in the e2e workflow time is larger in the first case. It is possible to show that for both the considered cases, the workflow completion time is equally delayed, as a consequence of using uniform distributions to characterize activities of the workflow. Since the completion time is identically delayed for the tested cases, the average utility of the aggregator (first addend in Eq. (4.3)) is identical same in both cases. Instead, as the delay amount increase, the cumulative utility of providers (second addend in Eq. (4.3)) is more penalized when many aggregators results to be delayed Consequently, the utility of the system tends to decrease more slowly in the unbalanced delay case than in the balanced case. This suggests that the presence of very reliable providers succeeds in shielding the presence of an unreliable provider, more than many equally reliable providers are able to.

4.4.4 Analysis under variable competitiveness

The impact of the competitiveness among providers on the aggregator utility and the provider utility is evaluated. To this end, $A = 1$ aggregator and $C = 1$ customer requesting one workflow with mixed topology are considered. Then, different scaling factor Δ_μ values are considered. In particular, values of Δ_μ rules the earliness of the expected value and the width of the support of the uniform CDF of the best bid with respect to expected value and the width of the support of the uniform CDF of the second best bid, identified by the VCG auction banned for each service s. In particular, the second best bid is a fixed uniform distribution with expected value $\mu_{\text{2nd}} = 10$ and support $[7.5, 12.5]$ and, while the best bid is a uniform distribution having expected value $\mu_{\text{1st}} = \frac{3\mu_{\text{2nd}}}{2\Delta_\mu}$ and support $\left[\frac{2\mu_{\text{1st}}+\log_2 \Delta_\mu - \mu_{\text{2nd}}/2}{2}, \frac{2\mu_{\text{1st}}-\log_2 \Delta_\mu + \mu_{\text{2nd}}/2}{2}\right]$, with $\Delta_\mu \in \{1, 2, 4, 8, 16\}$.

The remaining parameters are selected as illustrated in Section 4.4.1, respectively.

For each value of Δ_μ, 200 instances of the problem are generated, deriving the average aggregator utility and the average provider utility, as illustrated in Figs. 4.6a and 4.6b. As Δ_μ increases, the average aggregator utility in Fig. 4.6a increases due to the fact that $(1 - F_c(t_E))$ increases. From a certain value of Δ_μ on, $(1 - F_c(t_E))$ becomes equal to 1 and thus the average aggregator utility remains almost constant. To analyze average aggregator utility under different market settings, experiments are repeated with $\Theta \in \{\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{5}{6}, 1\}$. For each value of Δ_μ, the greatest value of the average aggregator utility is obtained with $\Theta = 1$, which in fact represents the best-case condition where aggregators receive the highest possible payment from customers according to Eq. (4.2). According to this, as the value of Θ decreases, also the average aggregator utility decreases.

Similarly, the average provider utility in Fig. 4.6b increases with Δ_μ, up to a certain value after which it remains constant, which comprises a direct consequence of the VCG mechanism. In fact, according to Eq. (4.9), as the first best bid outperforms more and more the second best bid, the average provider utility increases, until the service completion time corresponding to the first best bid is deterministically better than that of the second best, in which case the average utility of each provider tends to 1, i.e., the maximum value achievable. In Fig. 4.6b, given that the workflow consists of 8 services, the average utility of providers tends to 8.

Chapter 5

Model-Driven Engineering for manufactures predictive analysis

This chapter illustrates a Model-Driven Engineering (MDE) approach the application of the research methods proposed in Chapters 2 and 4 to concrete models coming from the real application contexts of a manufacturing district. In particular, after the characterization of a meta-model that describes entities of a manufacturing district and their relationships, the approach shows how to implement Model-to-Model (M2M) transformations [47] to map manufacturing domain-specific models of production to the environment presented in Fig. 4.1, comprising the production workflow specification as structure tree, enabling their quantitative evaluation.

The chapter is organized as follows. In Section 5.1, a metamodel of a manufacturing district is provided, with particular attention to the entities that can be mapped to a workflow and to the competition environment of Chapter 4. Section 5.2 illustrates how M2M transformations among models conforming to the different considered metamodels are accomplished. Then the approach is demonstrated on the model of a production from the context of a textile manufacturing district. The described process shows how MDE can be exploited in different application domains to provide added value features for planning or taking process design decisions.

5.1 A metamodel of a manufacturing district

A manufacturing district is a complex environment in which many parties cooperate and compete to produce material goods and products for a specific type of industry (e.g., garments for the fashion industry, electronic components for high-tech industry). Typically, the considered parties are specialized in solving different types of processings, which are subject to precedence constraints that determine their order of execution. The result is a complex puzzle, which is difficult to manage and plan, and that could benefit from the implementation of efficient and accurate quantitative valuation methods. In this section, a general description of manufacturing district is provided (Section 5.1.1), and a metamodel of this contexts is illustrated (Section 5.1.2).

5.1.1 Description of the context

In a manufacturing district, there are three typologies of parties that interact with each other.

- The *customer* (e.g. a fashion brand) is the party that requires the production of some good or product to be used for its business.
- The *converter* (e.g. a woolen mill) is the party who acquires orders from customers, taking charge of production autonomously, or partially involving other parties.
- The *contractor* (i.e. a dyeing factory, yarning factory) is a party that has specialized manufacturing resources that can be deployed to accomplish processings of a larger production acquired from a converter.

When a converter acquires an order from a customer, a supply contract that specifies a product from those available in the converter *catalog* is concluded. Specifically, the contract details the required quantities and the expected delivery dates of the good. A product is realized through a series of processings. A *processing* is a transformation that takes *raw materials* or *subassemblied products* as input and produces subassemblied products or *finished products*. Each processing has its own specific characterization, which determines the type of absorbed production resources and the time required to complete it.

Product-specific information are encoded by the converter in the *Bill of Materials* (BOM), which is a document that specifies the hierarchy of components that constitute a product, comprising the quantities required for every type of raw material or subassemblied product. The BOM determines the *Bill of Processes* (BOP), which is a document that specifies the order of execution among the processings. The BOM and the BOP become primary characteristics of each type of product available in the catalog of a converter. The processings defined in a BOP can be allocated by the converter to different contractors. The choice may depend on several criteria, such as the contractor resources, reliability, or productivity.

Contracts between customers, converters and contractors, specify the expected completion time to complete the production or to end a specific processing. This time can be specified as a date or a time slot, and can be characterized as a deterministic growth (e.g., a specific date, a range of days) or as a sample CDF (e.g. within the day *x* production completes at 90%, within the day *y* at 95%, and so on). Expected times are subject to delays that are induced by internal events (e.g. reallocation of resources to higher priority productions, unexpected machine failures or stuff absence) or external events (e.g. fashion brand requires an invasive change to a production, logistical problems or delays in raw material deliveries). The occurrence of a delay results in a decrease in the profit of the parties affected by the delay. In this case, mechanisms can be provided to penalize the utility or reputation of the delayer, with the intent of to minimize the loss of profit caused by the delay, or to exclude untrustworthy parties from future assignments.

5.1.2 The proposed metamodel

In Section 5.1.2, it is shown a UML Class Diagram that specifies a metamodel for the described context.

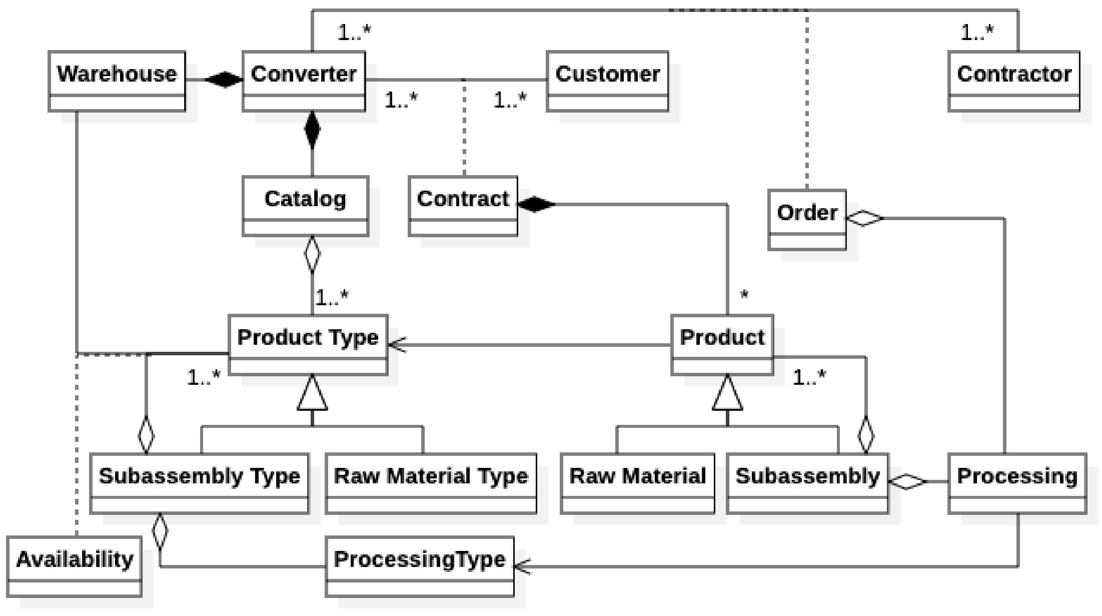


Figure 5.1: The proposed UML Class Diagram of the metamodel of a manufacturing district.

In the metamodel, the *Customer* and *Converter* entities are in a many-to-many association through the entity *Contract*. This entity specifies the *Product* entities that are required by a customer. The desired product can be chosen among those that are collected in the *Catalog* entity of every converter. The catalog contains the BOMs of the available types of product, which specify the recipe to realized a single concrete unit of that product type. The BOM of a product type is characterized as a Composite Pattern [88], where the interface is the entity *Product Type*, the basic element is the *Raw Material Type* entity, and the compositional elements are the *Subassembly Type* entities. When customer and converter close a contract that specifies the required product types, concrete production of that product type can be started by implementing a *Reflection* mechanism. The *Reflection* pattern [89] separates an abstract layer of representation from an operational layer. The abstract layer enables the specification of relationships, attributes, and features that characterize an entity in an absolute sense, while the operational layer enables the concretization of those relationships, attributes, and features, by specifying their concrete values. Therefore, the Composite pattern encoded in entities *Product Type*, *Subassembly Type* and *Raw Material Type* will correspond to the Composite pattern implemented by the entities *Product*, *Subassembly* and *Raw Material*, which define the nominal quantities that are expressed by the related types. This mechanism enables to handle an order based on the available stocks in the warehouse. In fact, the quantity of units required for a product type to fulfill an order requested by a customer is calculated net of stocks of that product type, which are encoded in the association entity *Availability*. Also the representation of processing exploits the Reflection Pattern. Specifically, while entity *Processing Type* characterizes the type of processing in terms of some quality attribute, the entity *Processing* represents the processing in terms of its nominal parameters such as the machine allocated for that processing or its operating period. Each process is assigned to an entity *Contractor*. The many-to-many relationship between *Converter* and *Contractor* is represented by the entity *Order*, which contains the reference to the *Processing* entities that are assigned by the converter to the contractor included in the association. The choice among different Contractors for a specific Processing is delegated to external strategies.

The proposed metamodel is effective in characterizing the context of a manufacturing district, from the perspective of the converter. In fact, the metamodel includes all the entities and relationships that are necessary to characterize the macro-planning of a product supply chain. Specifically, the metamodel enables to represent BOMs, involved cooperating/competing parties, required processings and warehouse stockings. All these information can be used by a Converter to manage and plan the production of a good, and could be valued by the application of the scientific methods presented in this thesis work, since they would enable a smart support for supply chain design choices. This is pursued by applying MDE approaches, in which some transformations that can map instances of this metamodel to instances of the metamodels described in Chapters 3 and 4 are identified, enabling the application of the presented scientific methods to a concrete case of application.

5.2 Demonstration on a case of study

Model-to-model (M2M) transformations are used to move from one domain to another one much closer to the solution domain [47]. In this section, a description of the transformations required to map manufacturing district production models to the scientific application models defined for Chapters 2 and 4 is provided (Section 5.2.1); then, an application example is illustrated (Section 5.2.2).

5.2.1 Description of the transformations

In our case study, the transformation aims at providing a model of the competition environment described by model of Fig. 4.1, comprising the structure tree workflow representation of the BOP of the production. This involves two types of M2M transformations. The first concerns the parties in the model of Section 5.1.2 and how these are mapped to the participants in the model of Fig. 4.1. The second determines how the concrete BOP of a product can be mapped to the structure tree of a workflow that use each processing of the BOP as a workflow activitiy. Note that since the Customer of model Fig. 4.1 requires a workflow to be provided, then it is possible to embed the mapping between BOP and structure tree workflow in the mapping between the parties of Section 5.1.2 and the participants of Fig. 4.1.

Through the definition of these transformation, it is possible to exploit scientific methods in a manufacturing district to determine the optimal assignment among contractors and processings, and to predict the completion time of a production. Following the classification in [90], this M2M process can be seen as a bidirectional transformation, where both domains can be used as the source and target domain of the transformation. Recalling that the MDE approach is aimed at determining the best contractors for the production processings, and the best matching among customer and converter, exploiting the structure tree workflow specification of the production BOP to estimate the completion time of the production, the bidirectionality of the transformations can be observed in the steps required to get an optimal matching between the entity of the manufacturing district:

- Initially, some instances of *Customer*, *Converter* and *Contractors* are provided, in the manufacturing district domain. At the beginning of the process, any *Contract* nor *Order* entity associates the district parties among them.

- Then, *Contractors* of the manufacturing district domain are mapped to *Provider* of the scientific application domain, *Converter* to *Aggregator* and *Customer* to *Customer*. For each of these maps, simple attributes (e.g. name of the parties, available resources) are directly mapped to the analogous attributes in the target domain.

- In the *Contractor*-to-*Provider* transformation, an information is added in the target domain. In particular, the stochastic service time of the provider is specified by passing a parameter to the method that implements the transformation. This can also be easily implemented as a Rest API, that can be called by contractors that intend to compete in the auction of a certain processing.

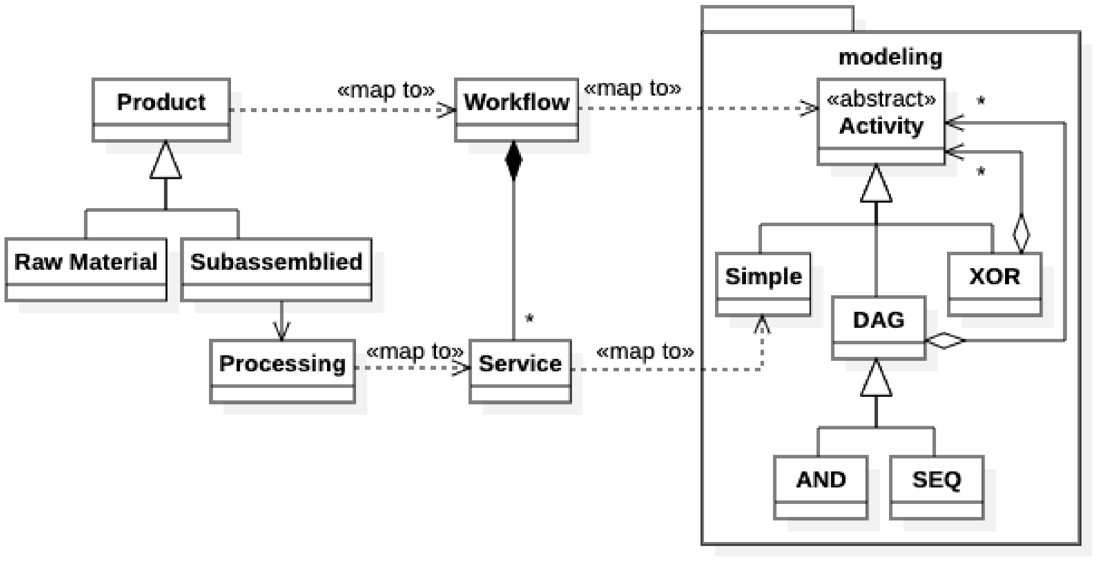
- In the *Customer*-to-*Customer* transformation, the desired stochastic service time is specified as done in the previous step.

- The *Customer*-to-*Customer* transformation also embeds the specification of the structure tree workflow representation of the desired BOP. In particular, chosen the desired product BOM from the Catalog, since every *Subassembly Type* of the BOM has a reference to a *Processing Type* entity, it is possible to build up the workflow structure tree by simply visiting the Composite of the product BOM.

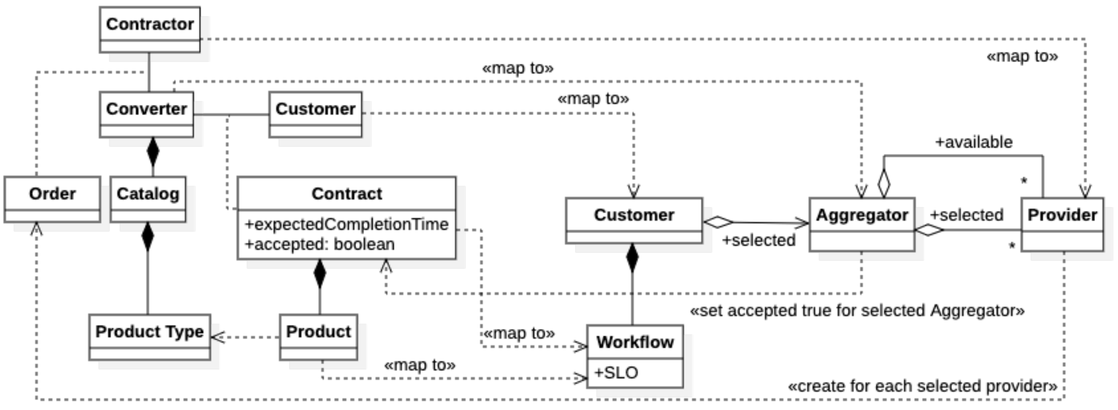
- In the auction-game domain, it is now possible to play the algorithm described in Section 4.2 for every processings constituting the activities of the BOP workflow. This also comprises the evaluation of the workflow through the compositional method of Algorithm 1.

- After the auction, a transformation to the manufacturing district domain is performed. back. In particular, winning providers are mapped back to *Contractor* instances, associating them to the instance of *Converter* through the entity *Order*. The entity *Order* includes the timing details corresponding to the SLO and the SLA of the target domain, and the references to the *Processing* entities for which the contractor has won the auction.

A schema of the proposed transformations is provided in the UML Class Diagram of Figs. 5.2a and 5.2b, where the main classes of the considered domains are recalled, and the perfomed mapping are illustrated. In particular, Fig. 5.2a shows the map the enables a BOP to be transformed into its structure tree workflow of activities, and Fig. 5.2b shows how the parties of a manufacturing district are transformed into the participants of Fig. 4.1, also showing the bidirectionality of the mapping which is encoded in the dependencies from entities *Contractor* and *Provider*, *Customer* and *Customer*, *Product* and *Workflow*, on one direction, and *Aggregator* and *Contract*, *Provider* and *Order*, on the other direction.



(a)



(b)

Figure 5.2: The UML Class Diagram that illustrates the proposed M2M transformations. In particular, (a) demonstrates how a BOM (BOP) is mapped to a workflow; (b) shows how parties from different domains are mapped from a domain to the other.

By means of this set of rules, it is easy to switch from one of the considered domain to the other and to apply the scientific methods presented in Algorithm 1 and Section 4.2 to the concrete context of a manufacturing district. The application of these methods bring a higher level of knowledge that enables to perform more reliable business choices based on sound information.

5.2.2 Application of the transformations

To demonstrate the scientific method, the model of a textile manufacturing production, having the BOM illustrated in the UML Object diagram of Section 5.2.2, is considered. The model represents an order in which two fabrics of different colors are requested and released after a finishing processing. The first fabric (*Waved Fabrics A*) is obtained by applying processes of *spooling* and *weaving* to a yarn (A); the second fabric is obtained from two different yarns (B and C), one of which (C) is before subjected to the process of *washing*, in order to eliminate impurities from the early stages. Part of the *Weaved Fabrics B* is coloured using the raw material *Dye B*, producing the subassemblied product *Colored Fabrics B*; the remaining *Weaved Fabrics B* and *Weaved Fabrics A* are subjected to another dyeing process, which consumes the raw material *Dye A* to produce the product *Colored Fabrics A*. Finally, the obtained fabrics are finished and ready to be delivered. From the illustrated model, it is possible to evince that the types of processes involved in the production are 5, *washing*, *spooling*, *weaving*, *dyeing* and *finishing*, some of which are performed multiple times, for a total of 9 concrete processings. Through the transformation rules illustrated in Fig. 5.2a, the considered model is mapped to the workflow structure tree shown in Section 5.2.2.

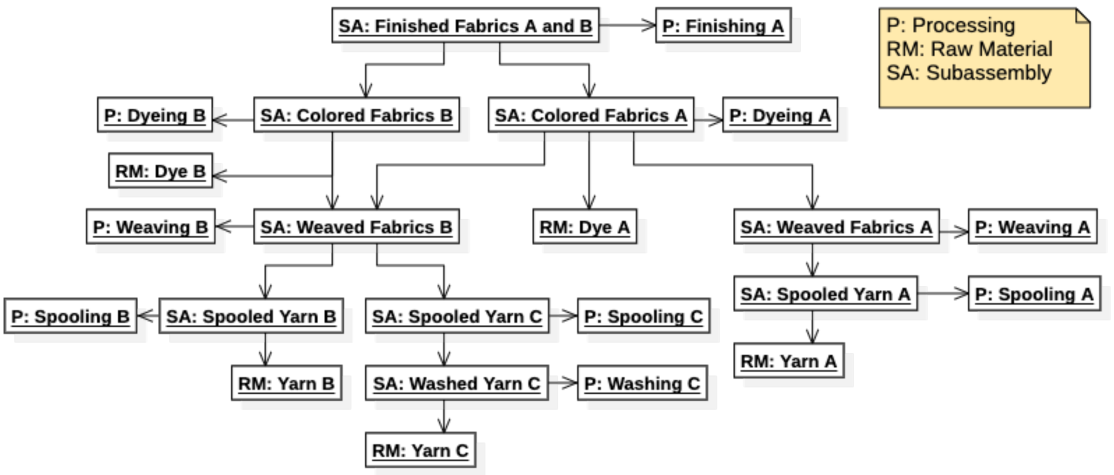


Figure 5.3: The UML Object Diagram of the BOM and BOP of the production described in Section 5.2.2.

For the sake of simplicity, in the perspective of the involved parties, only one customer and converter are considered, making trivial the matching problem described in Section 4.3, as only one pair can be formed. For each of the processing, 3 different contractors are considered. Each of these contractors participates in the VCG auction described in Section 4.2 offering a uniform distribution as a bid. Parameters

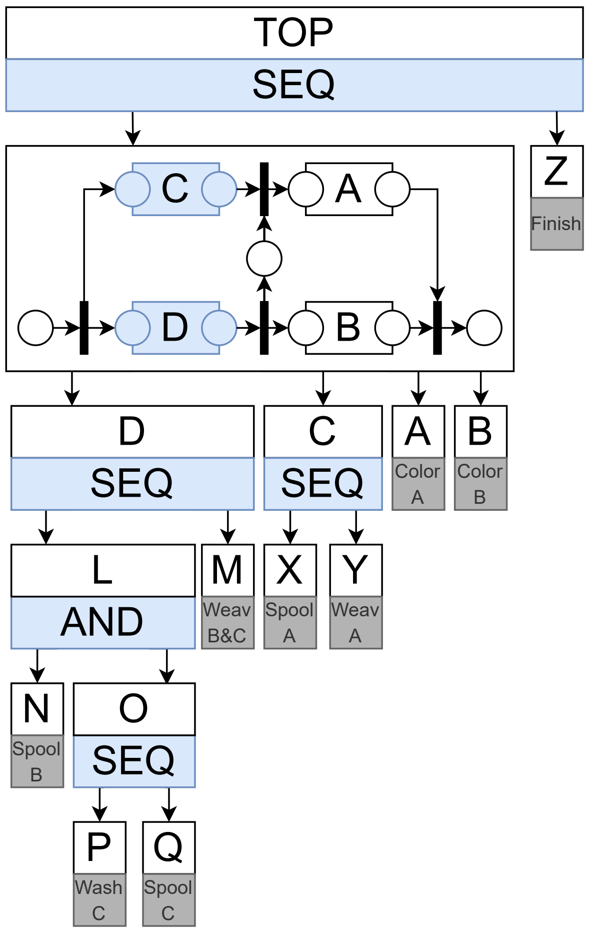


Figure 5.4: The structure tree of the workflow obtained applying the proposed M2M transformation to the model illustrated in Section 5.2.2.

a of the distributions is randomly drawn in the interval $[0, 2]$, while parameters b in the interval $[3, 5]$, so that every process is guaranteed to last at least 1 time unit. Table 5.1 shows the drawn stochastic parameters that characterize the uniform distribution provided by each contractor for a specific processing, ordered by the first, the second and the third best bids, showing the winners and the time constraints to which they are subjected.

After contractors are assigned to activities, the completion time of the workflow can be evaluated through the compositional method described in Section 2.3. In particular, given the workflow structure, the comparison conducted between the two analysis heuristics in Section 2.4.2 leads to exploit heuristic 1 to evaluate the completion time both during the auction game and after assigning contractors to tasks. For each analysis, the time horizon is set to the workflow support and the time tick to 0.1 time unit. The obtained PDF and CDF completion time are shown in Fig. 5.5. The figures show how different contractors impact on the final results, highlighting the effectiveness of the game-theoretical matching-based approach. The red lines, the blue lines and the green lines show the completion times when activities are distributed according to the bid of the best, the second best and the worst contractors, respectively. As expected the obtained CDFs are stochastically ordered, confirming that choosing more efficient contractors determines a faster completion time of a supply chain workflow. Result are obtained on a single core of an Intel Xeon Gold 5120 CPU (2.20 GHz) equipped with 32 GB of RAM.

Processing	**Contractor**	**a**	**b**	**Processing**	**Contractor**	**a**	**b**
Spooling A	1st	0.92s	3.56s	**Spooling B**	1st	0.15s	3.45s
	2nd	1.04s	3.91s		2nd	1.24s	3.13s
	3rd	0.14s	4.89s		3rd	1.00s	4.34s
Washing C	1st	0.56s	4.01s	**Spooling C**	1st	0.00s	4.13s
	2nd	0.35s	4.41s		2nd	0.50s	3.75s
	3rd	1.25s	4.52s		3rd	0.58s	4.58s
Weaving A	1st	0.44s	4.20s	**Weaving B**	1st	1.00s	3.86s
	2nd	0.21s	4.91s		2nd	0.03s	4.85s
	3rd	1.79s	3.56s		3rd	1.85s	3.88s
Dyeing A	1st	0.00s	3.48s	**Dyeing B**	1st	1.36s	3.20s
	2nd	0.84s	3.93s		2nd	1.45s	3.35s
	3rd	1.17s	4.01s		3rd	1.60s	4.87s
Finishing	1st	0.08s	3.89s				
	2nd	0.99s	4.00s				
	3rd	0.34s	4.72s				

Table 5.1: Parameters of uniform distributions [a, b] bid by contractors for each processing. For each processing, the distributions are ordered from the best contractor (1st) to the worst (3rd), according to the Pairwise-comparison dominance defined in Definition 6.

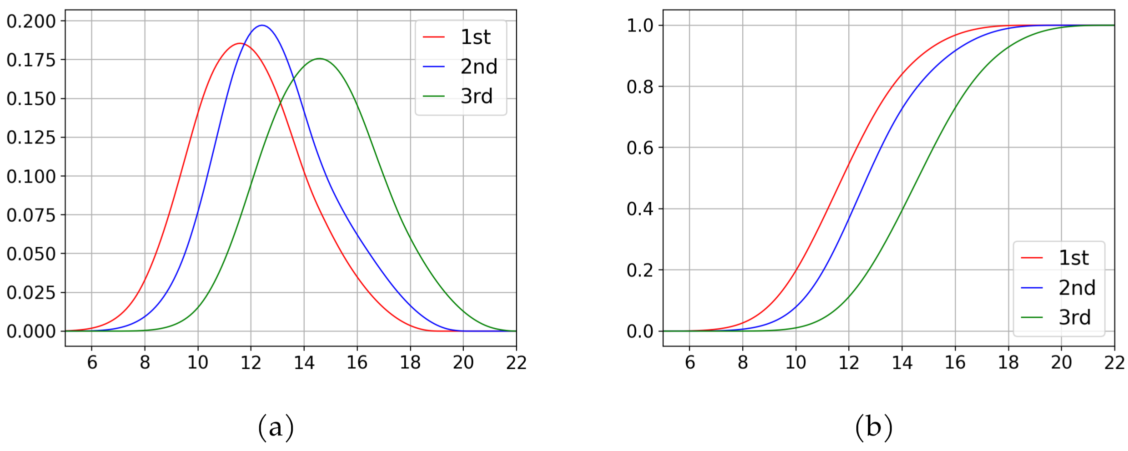


Figure 5.5: PDF (a) and CDF (b) evaluated for the workflow of Section 5.2.2 with different stochastic characterization of the activities.

Chapter 6

Conclusions

This thesis work develops around the concept of workflow, which is explored in multiple aspects. On one hand, a hierarchical workflow representation formalism was identified, enabling the development of some compositional heuristics methods that efficiently provide an accurate stochastic upper bound of the response time PDF of a workflow. The conducted experimentation shows that the proposed compositional method scales easily on workflows with a high degree of complexity, i.e. workflows having a high number of concurrent and generally distributed over bounded support durations. In particular, experiments on a suite of synthetic 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. The conducted research and the intent of implementing a robust experimentation to support it, leads to the realization of a Java library that provides features to headlessly model and evaluate workflows. The library is designed with the intent of ensuring easy extendibility, opening to future integrations or extensions of the researched methods. In particular, workflow modeling could be extended with other constructs, possibly affecting well-formed nesting (e.g. introducing loops) or coming from different formalism (e.g. Business Process Model and Notation (BPMN) or Business Process Execution Language (BPEL)), provided that positive correlation is guaranteed among the response times of different subworkflows. In turn, workflow evaluation is open to the definition of other heuristics to differently explore the structure tree and decompose the workflow, to the exploitation of other analytical approximations in the class of exponomial functions or piecewise CPHs over bounded supports [91] to fit numerical PDFs, and to the integration of other solution techniques to evaluate the response time PDF of a block.

On the other hand, workflows are addressed from the perspective of the definition of their stochastic parameters, in a context of competition between antagonistic parties on different layers of competition. In particular, a two-layer scenario is considered where customers request repeated execution of complex workflows to resource-constrained aggregators under specific SLOs, and aggregators offer service outsourcing to providers competing for service delivery. Customer SLOs are expressed as CDFs of the required e2e workflow completion time, and provider bids are expressed as CDFs of the offered service completion time. The proposed stochastic framework solves the assignment problem between customers and aggregators through a matching game with externalities and incomplete information, during which aggregators select providers through a VCG auction mechanism, which characterizes the stochastic parameters of the workflow. Experiments have been performed to investigate the feasibility, the effectiveness and the robustness of the approach. Results shows that the local and partial information used to build the preference lists of customers is still able to properly solve the assignment of customers to aggregators. Moreover, results also show that the approach can be effectively used to estimate how the aggregator utility is impacted by unexpected delays in service supply and by the level of competitiveness among providers. It is worth noting that this kind of experimental evaluation is enabled by the fact that both customer SLOs and provider bids are expressed in terms of CDFs (of workflow e2e time and individual service completion time, respectively), and viceversa is substantially ruled out when they are expressed as fixed values. The approach is open to various extensions. In particular, a hierarchical framework with more than two levels could be designed, investigating the performance of combinations of solution methods other than the one presented in this paper, e.g., the VCG auction mechanism could be exploited to solve the assignment problem at each level of the hierarchy. Notably, the possibility to represent customer SLOs and provider bids through non-Markovian CDFs (which are supported by the SIRIO and Eulero libraries, used to implement the proposed approach) facilitates fitting of data (either observed or synthetically produced) and thus application to a real application scenario.

Finally, this thesis work shows the use of the presented methods on real application cases from a manufacturing districts, following an MDE approach. In fact, by implementing M2M transformations that maps manufacturing district parties and Bill of Materials to a resource constrained environment and to structure tree workflow, respectively, the proposed compositional evaluation method and the VCG auction are used to determine the stochastic parameters and the completion time of a supply chain of from textile manufacturing district. These applications demonstrate that the topics covered in the thesis have both a relevant scientific impact, and a centrality for processes belonging to industrial and digital business sectors, as they solve salient problems that are shared among different application areas.

Appendix A

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

Peer reviewed conference papers

1. **L. Carnevali, R. Reali, E. Vicario**, "Compositional evaluation of stochastic workflows for response time analysis of composite web services", *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (*ICPE21*), pages:177–188, 2021. (**Best paper award**)
   Candidate's contributions: theoretical analyses, design and implementation of experimentation.

2. **L. Carnevali, M. Paolieri, R. Reali, E.Vicario**, "Compositional Safe Approximation of Response Time Distribution of Complex Workflows", in *International Conference on Quantitative Evaluation of Systems* (*QEST21*), pages:83–104, 2021.
   Candidate's contributions: theoretical analyses, design and implementation of experimentation.

3. **L. Carnevali, R. Reali, E. Vicario**, "Eulero: a tool for quantitative modeling and evaluation of complex workflows", in *International Conference on Quantitative Evaluation of Systems* (*QEST22*), pages:xx–xx, 2022.
   Candidate's contributions: design and implementation of the library and the packages.

Workshop papers

1. **P. Cappanera, L. Carnevali, L. Paroli, R. Reali, E. Vicario**, "Resource allocation for complex DAG tasks with probabilistic execution times", in *11th Inter-*

[1]The author's bibliometric indices are the following: H-index = 2, total number of citations = 5 (source: Google Scholar on Month March, 2023).

national Real-Time Scheduling Open Problems Seminar (RTSOPS22), 2022.
Candidate's contributions: problem formulation, literature analysis, identification of open issues.

2. **L. Carnevali, M. Paolieri, R. Reali, L. Scommegna, F. Tammaro, and E. Vicario**, "Using the ORIS tool and the SIRIO library for model-driven engineering of quantitative analytics", in *18th European Performance Engineering Workshop (EPEW22)*, September 2022.
Candidate's contributions: problem formulation, design and implementation of the experimentation.

Papers under review

1. **L. Carnevali, M. Paolieri, R. Reali, E. Vicario.** "Compositional safe approximation of response time probability density function of complex workflows", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*.
Candidate's contributions: theoretical analyses, design and implementation of experimentation.

Bibliography

[1] N. Russell, A. H. Ter Hofstede, W. M. Van Der Aalst, and N. Mulyar, "Workflow control-flow patterns: A revised view," *BPM Center Report BPM-06-22, BPMcenter. org*, pp. 06–22, 2006.

[2] T. G. de Kok and J. C. Fransoo, "Planning supply chain operations: definition and comparison of planning concepts," *Handbooks in operations research and management science*, vol. 11, pp. 597–675, 2003.

[3] W. M. Van der Aalst, "The application of petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

[4] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia, "Modeling resources to simulate business process reliability," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 30, no. 3, pp. 1–25, 2020.

[5] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business process execution language for web services," 2002.

[6] E. Van Eyk, A. Iosup, C. L. Abad, J. Grohmann, and S. Eismann, "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.

[7] A. Rogge-Solti and M. Weske, "Prediction of business process durations using non-markovian stochastic petri nets," *Information Systems*, vol. 54, pp. 1–14, 2015.

[8] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "Qos-aware replanning of composite web services," in *Proc. IEEE Int. Conf. on Web Services*, pp. 121–129, IEEE, 2005.

[9] G. Casale, M. Artač, W.-J. Van Den Heuvel, A. van Hoorn, P. Jakovits, F. Leymann, M. Long, V. Papanikolaou, D. Presenza, A. Russo, *et al.*, "Radon: rational decomposition and orchestration for serverless computing," *SICS Software-Intensive Cyber-Physical Systems*, vol. 35, no. 1, pp. 77–87, 2020.

[10] A. U. Gias, A. van Hoorn, L. Zhu, G. Casale, T. F. Düllmann, and M. Wurster, “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.

[11] D. Bruneo, S. Distefano, F. Longo, and M. Scarpa, “Qos assessment of ws-bpel processes through non-markovian stochastic petri nets,” in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pp. 1–12, IEEE, 2010.

[12] E. D. Jensen, C. D. Locke, and H. Tokuda, “A time-driven scheduling model for real-time operating systems.,” in *Rtss*, vol. 85, pp. 112–122, 1985.

[13] J. Rahman and P. Lama, “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.

[14] G. Ciardo, R. German, and C. Lindemann, “A characterization of the stochastic process underlying a stochastic petri net,” *IEEE Transactions on Software Engineering*, vol. 20, no. 7, pp. 506–515, 1994.

[15] S. Distefano and K. S. Trivedi, “Non-markovian state-space models in dependability evaluation,” *Quality and Reliability Engineering International*, vol. 29, no. 2, pp. 225–239, 2013.

[16] R. German and C. Lindemann, “Analysis of stochastic petri nets by the method of supplementary variables,” *Perf. Eval.*, vol. 20, no. 1-3, pp. 317–335, 1994.

[17] M. Telek and A. Horváth, “Transient analysis of age-mrspns by the method of supplementary variables,” *Perf. Eval.*, vol. 45, no. 4, pp. 205–221, 2001.

[18] H. Choi, V. G. Kulkarni, and K. S. Trivedi, “Markov regenerative stochastic petri nets,” *Performance evaluation*, vol. 20, no. 1-3, pp. 337–357, 1994.

[19] A. Bobbio, A. Horváth, and M. Telek, “Matching three moments with minimal acyclic phase type distributions,” *Stochastic models*, vol. 21, no. 2-3, pp. 303–326, 2005.

[20] P. Reinecke, T. Krauß, and K. Wolter, “Cluster-based fitting of phase-type distributions to empirical data,” *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3840–3851, 2012.

[21] A. Horváth and M. Telek, “PhFit: A General Phase-Type Fitting Tool,” in *Proc. Int. Conf. on Comput. Perf. Eval., Modelling Tech. and Tools*, pp. 82–91, 2002.

[22] P. Reinecke, T. Krauß, and K. Wolter, "Hyperstar: Phase-type fitting made easy," in *2012 Ninth International Conference on Quantitative Evaluation of Systems*, pp. 201–202, IEEE, 2012.

[23] P. Reinecke, T. Krauß, and K. Wolter, "Phase-Type Fitting Using HyperStar," in *Proc. Europ. Perf. Eng. Workshop*, pp. 164–175, 2013.

[24] R. Fricks, M. Telek, A. Puliafito, and K. S. Trivedi, "Markov renewal theory applied to performability evaluation," tech. rep., North Carolina State University. Center for Advanced Computing and Communication, 1996.

[25] A. Horváth, M. Paolieri, L. Ridi, and E. Vicario, "Transient analysis of non-markovian models using stochastic state classes," *Performance Evaluation*, vol. 69, no. 7-8, pp. 315–335, 2012.

[26] R. German, D. Logothetis, and K. S. Trivedi, "Transient analysis of markov regenerative stochastic petri nets: A comparison of approaches," in *Proceedings 6th International Workshop on Petri Nets and Performance Models*, pp. 103–112, IEEE, 1995.

[27] A. Bobbio and M. Telek, "Markov regenerative spn with non-overlapping activity cycles," in *Proc. Int. Comput. Perf. and Depend. Symp.*, pp. 124–133, 1995.

[28] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *IEEE Int. Symp. on Software Reliability Engineering*, pp. 210–219, IEEE, 2011.

[29] Y. Liu, Z. Zheng, and J. Zhang, "Markov model of web services for their performance based on phase-type expansion," in *Proc. DASC-PICOM-CBDCOM-CYBERSCITECH*, pp. 699–704, IEEE, 2019.

[30] F. Arnold, H. Hermanns, R. Pulungan, and M. Stoelinga, "Time-dependent analysis of attacks," in *Proc. Int. Conf. on Principles of Security and Trust*, pp. 285–305, Springer, 2014.

[31] L. Carnevali, R. Reali, and E. Vicario, "Compositional evaluation of stochastic workflows for response time analysis of composite web services," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pp. 177–188, 2021.

[32] L. Carnevali, M. Paolieri, R. Reali, and E. Vicario, "Compositional safe approximation of response time distribution of complex workflows," in *Proceedings of QEST 2021*, vol. 12846 of *Lecture Notes in Computer Science*, pp. 83–104, Springer, 2021.

[33] E. Vicario, L. Sassoli, and L. Carnevali, "Using stochastic state classes in quantitative evaluation of dense-time reactive systems," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 703–719, 2009.

[34] R. A. Sahner and K. S. Trivedi, "Performance and reliability analysis using directed acyclic graphs," *IEEE Transactions on Software Engineering*, no. 10, pp. 1105–1114, 1987.

[35] K. S. Trivedi and R. Sahner, "SHARPE at the Age of Twenty Two," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 52–57, Mar. 2009.

[36] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," in *2009 IEEE 25th International Conference on Data Engineering*, pp. 1733–1740, 2009.

[37] F. Sheikholeslami and N. Jafari Navimipour, "Auction-based resource allocation mechanisms in the cloud environments: A review of the literature and reflection on future challenges," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 16, p. e4456, 2018.

[38] G. Baranwal and D. P. Vidyarthi, "A truthful and fair multi-attribute combinatorial reverse auction for resource procurement in cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 6, pp. 851–864, 2019.

[39] S. Sebastio, G. Gnecco, and A. Bemporad, "Optimal distributed task scheduling in volunteer clouds," *Computers & Operations Research*, vol. 81, pp. 231–246, 2017.

[40] A. N. Gullhav and B. Nygreen, "A branch and price approach for deployment of multi-tier software services in clouds," *Computers & Operations Research*, vol. 75, pp. 12–27, 2016.

[41] C.-H. Chen-Ritzo, T. P. Harrison, A. M. Kwasnica, and D. J. Thomas, "Better , faster , cheaper : A multi-attribute supply chain auction mechanism," 2003.

[42] C. Xu, L. Song, Z. Han, Q. Zhao, X. Wang, X. Cheng, and B. Jiao, "Efficiency resource allocation for device-to-device underlay communication systems: A reverse iterative combinatorial auction based approach," *CoRR*, vol. abs/1211.2065, 2012.

[43] J. Huang, Z. Han, M. Chiang, and H. V. Poor, "Auction-based resource allocation for cooperative communications," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 7, pp. 1226–1237, 2008.

[44] V. Krishna, *Auction Theory*. Elsevier Science, 2009.

[45] G. Christodoulou, *Price of Anarchy*. Boston, MA: Springer US, 2008.

[46] S. Singhal and V. Kavitha, "Coalition formation resource sharing games in networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 49, p. 57–58, mar 2022.

[47] A. R. Da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Computer Languages, Systems & Structures*, vol. 43, pp. 139–155, 2015.

[48] D. C. Schmidt, "Model-driven engineering," *Computer-IEEE Computer Society-*, vol. 39, no. 2, p. 25, 2006.

[49] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE software*, vol. 31, no. 3, pp. 79–85, 2013.

[50] Z. Zheng, K. S. Trivedi, K. Qiu, and R. Xia, "Semi-markov models of composite web services for their performance, reliability and bottlenecks," *IEEE Transactions on services computing*, vol. 10, no. 3, pp. 448–460, 2015.

[51] K. S. Trivedi, *Probability and statistics with reliability, queuing, and computer science applications*. John Wiley & Sons, 2001.

[52] V. S. Sharma and K. S. Trivedi, "Reliability and performance of component based software systems with restarts, retries, reboots and repairs," in *2006 17th International Symposium on Software Reliability Engineering*, pp. 299–310, IEEE, 2006.

[53] N. Sato and K. S. Trivedi, "Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks," in *International Conference on Service-Oriented Computing*, pp. 107–118, Springer, 2007.

[54] R. Johnson, D. Pearson, and K. Pingali, "The program structure tree: Computing control regions in linear time," in *ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI)*, pp. 171–185, ACM, 1994.

[55] J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," *Data Knowl. Eng.*, vol. 68, no. 9, pp. 793–818, 2009.

[56] D. L. Dill, "Timing assumptions and verification of finite-state concurrent systems," in *AVMFSS'89*, vol. 407 of *LNCS*, pp. 197–212, Springer, 1990.

[57] SIRIO Library. `https://github.com/oris-tool/sirio`, 2022.

[58] M. Paolieri, M. Biagi, L. Carnevali, and E. Vicario, "The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems," *IEEE Transactions on Software Engineering*, vol. 47, pp. 1211–1225, June 2021.

[59] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Transactions on Software Engineering*, vol. 17, no. 3, pp. 259–273, 1991.

[60] E. Vicario, "Static analysis and dynamic steering of time-dependent systems," *IEEE Trans. Softw. Eng.*, vol. 27, pp. 728–748, Aug. 2001.

[61] L. Sassoli and E. Vicario, "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.

[62] F. Baccelli and A. M. Makowski, "Multidimensional stochastic ordering and associated random variables," *Operations Research*, vol. 37, no. 3, pp. 478–487, 1989.

[63] L. Carnevali, R. Reali, and E. Vicario, "Eulero: a tool for quantitative modeling and evaluation of complex workflows," in *Proceedings of QEST 2022*.

[64] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[65] F. Nielsen, "On a generalization of the jensen-shannon divergence and the js-symmetrization of distances relying on abstract means," *arXiv preprint arXiv:1904.04017*, 2019.

[66] G. Rubino and B. Tuffin, *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

[67] J. Bucklew, *Introduction to rare event simulation*. Springer Science & Business Media, 2013.

[68] M. Villén-Altamirano and J. Villén-Altamirano, "The rare event simulation method RESTART: efficiency analysis and guidelines for its application," in *Net. Perf. Eng.*, pp. 509–547, Springer, 2011.

[69] C. E. Budde, M. Biagi, R. E. Monti, P. R. D'Argenio, and M. Stoelinga, "Rare event simulation for non-Markovian repairable fault trees," in *TACAS*, pp. 463–482, Springer, 2020.

[70] M. Paolieri, M. Biagi, L. Carnevali, and E. Vicario, "The ORIS tool: quantitative evaluation of non-Markovian systems," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1211–1225, 2019.

[71] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 ed., 1994.

[72] E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis, "30 years of greatspn," in *Principles of Performance and Reliability Modeling and Evaluation*, pp. 227–254, Springer, 2016.

[73] R. P. Dick, D. L. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Proceedings of the Sixth International Workshop on Hardware/Software Codesign.(CODES/CASHE'98)*, pp. 97–101, IEEE, 1998.

[74] A. S. Foundation, "Apache airflow."

[75] S. T. SA, "Luigi."

[76] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based vm allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1702–1716, 2021.

[77] V. Vazirani, *Approximation Algorithms*. Springer Berlin Heidelberg, 2013.

[78] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[79] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Processing Magazine*, vol. 33, pp. 103–122, Nov 2016.

[80] D. Manlove, *Algorithmics of matching under preferences*, vol. 2. World Scientific, 2013.

[81] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[82] I. Francis A. Pearman, "Gentrification and academic achievement: A review of recent research," *Review of Educational Research*, vol. 89, no. 1, pp. 125–165, 2019.

[83] G. Bridge, "Bourdieu, rational action and the time-space strategy of gentrification," *Transactions of the Institute of British Geographers*, vol. 26, no. 2, pp. 205–216, 2001.

[84] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer effects and stability in matching markets," vol. 6982, 03 2011.

[85] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services," *SIGOPS Oper. Syst. Rev.*, vol. 39, p. 45–58, oct 2005.

[86] M. Paolieri, M. Biagi, L. Carnevali, and E. Vicario, "The oris tool: Quantitative evaluation of non-markovian systems," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1211–1225, 2021.

[87] L. Carnevali, R. Reali, and E. Vicario, "Eulero: A tool for quantitative modeling and evaluation of complex workflows," in *International Conference on Quantitative Evaluation of Systems*, pp. 255–272, Springer, 2022.

[88] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[89] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-oriented software architecture, patterns for concurrent and networked objects*. John Wiley & Sons, 2013.

[90] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, vol. 45, pp. 1–17, USA, 2003.

[91] L. Korenčiak, J. Krčál, and V. Řehák, "Dealing with zero density using piecewise phase-type approximation," in *European Workshop on Performance Engineering*, pp. 119–134, Springer, 2014.