IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 September 2022, accepted 7 October 2022, date of publication 10 October 2022, date of current version 17 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3213697

RESEARCH ARTICLE

Stochastic Timed Discrete-Event Systems: Modular Modeling and Performance Evaluation Through Markovian Jumps

CARLOS-ANDREY MAIA
Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Minas Gerais 31270-010, Brazil
e-mail: maia@cpdee.ufmg.br

This work was supported in part by the Brazilian Agency Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and in part by Pró-Reitoria de Pesquisa (PRPq) da Universidade Federal de Minas Gerais (UFMG).

ABSTRACT We are interested in a scalable, flexible, and modular methodology, for modeling and performance analysis of stochastic discrete-event systems (SDES). In this sense, we propose a modular approach for timing non-markovian SDES expressed as a parallel composition of modules that interacts with each other through events. We show how general distribution for event lifetimes can be implemented systematically by coupling timing modules to the system model. As a result, this coupling mechanism preserves modularity, leading to a compact markovian model expressed in terms of flexible modules. Therefore the methodology allows us to write the whole SDES model as a composition of the system model and the timing one, giving flexibility and scalability in modeling design, as we can modify the modules individually according to the designer's interests. In addition, from the whole markovian SDES model, we show how to perform the model analysis through the analytic approach, as well as through Monte Carlo computer simulation. As an application, we present a numerical example of computing the abandonment rate for a service network with general service time employing both analytical and computer-simulation models.

INDEX TERMS Stochastic discrete-event systems, modular models, Markovianization techniques, analytic models, Monte Carlo computer simulation.

I. INTRODUCTION

Deterministic Timed Discrete-Event Systems (DTDES) applies well in situations where the uncertainties associated with the description of the variables involved are negligible, in other words, from a probabilistic point of view, the variance is very small compared to the expected value. Deterministic models can be used, for example, in the modeling and control of automated manufacturing systems [10], [14], [26], [27], transportation networks [17], diagnosing [5], learning [2], [9] and real-time scheduling of tasks on processors [6], [7].

In the present paper, we are concerned with Stochastic Discrete-Event Systems [4], which are used for modeling, controlling, diagnosing, and analyzing the performance of uncertain systems whose dynamics are driven by the occurrence of random events [1], [16], [24]. Applications include industry and intelligent systems, networked systems, service networks, material handling systems [15], [22], [23] and maintenance [21]. In particular, we are interested in taking into account explicitly time in the model construction, leading to what we denote as a Stochastic Timed Discrete-Event Systems (STDES).

An important point to be highlighted is that the models in STDES are in, general, complex due to the discrete nature of the entities, with the number of states that grow in a combinatorial way as more information about the studied process is added to the models. In this sense, the construction of models in a modular way simplifies and systematizes the task of modeling complex systems since the whole model is obtained by composing simpler sub-models.

The utilization of Modular Models is a kind of "divide-and-conquer" strategy used to solve problems in Discrete-Event Systems in different ways. In this context, we can cite, for instance: developments in analysis of untimed models

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Wang.

[11], [18], [28], [12], and [13]; approaches for reducing the computational complexity of synthesizing controllers in the supervisory control theory for timed models [25] and for simulation purposes of complex systems [8].

Unlike previous papers found so far in the literature, we are interested in the study of a modular approach for timing non-markovian STDES expressed as a parallel composition of modules that interacts with each other through events. We show how the general distribution for event lifetimes can be implemented by coupling timing modules to the system model. As a result, this coupling leads to a compact markovian system expressed in terms of flexible modules. This modularity is interesting for modeling purposes since it divides the whole model into system sub-models and the timing sub-models for events that have a non-markovian lifetime, giving flexibility and scalability in modeling design, since the modules can be modified individually according to the designer's interests. In the paper, we show how to use the resulting modular markovian STDES methodology to perform the model analysis through analytic and Monte Carlo computer simulation approaches.

Paper Contributions: The model construction for a Stochastic Timed Discrete Event -System (STDES) is done through the parallel composition of modules based on Stochastic Timed Automaton (STA). STA is a very compact structure with few entities allowing solid mathematical developments. As a result, the approach simplifies and reduces the errors in the modeling task, since the whole model is constructed through the composition of simpler sub-models. Moreover, it allows us to incorporate timing mechanism as modules described in terms of STA's, as well. Therefore, we have a complete modular description for the STDES in terms of STA's. Besides the flexibility in changing the modules, or parameters, according to the designer's interests, the approach gives us flexibility in developing equivalent analytical or Monte Carlo computer simulation models, as we show in the paper.

The sequence of this paper is organized as follows: in Section II, we define the basic elements of STDES that we are interested in, as well as the operation of parallel composition of sub-models. We also discuss how to analyze the dynamics of a STDES using analytic and Monte Carlo computer simulation approaches. In section III, we show how to construct and couple markovian timing modules to markovianize a STDES model, in particular, we develop timing modules for hypoexponential and hyperexponential distributions. In Section IV we discuss the space complexity for the model representation, as well as time complexity for Analytic and Computer Simulation approaches. In Section V, we show a numerical-application example for a service network with customers' abandonment. The conclusion and perspectives for this work are presented in Section VI.

II. STOCHASTIC TIMED AUTOMATA MODELS

It is known that the stochastic automaton is a very general structure to represent STDES, and it can even be used for simulation via discrete computing. In this paper, we are interested in STDES that can be modeled through the stochastic timed automaton defined as follows:

Definition 1: A **Stochastic Timed Automaton (STA)** is a quintuple $\mathcal{G} = (\mathcal{X},\ \mathcal{E},\ f,\ x_0,\ \mathcal{V})$, whose:

- $\mathcal{X}$ is a finite set of states;
- $\mathcal{E}$ is a finite set of events;
- f is a transition function, $f : \mathcal{X} \times \mathcal{E} \to \mathcal{X}$;
- x_0 is an initial state, $x_0 \in \mathcal{X}$;
- $\mathcal{V}$ $= \{\mathcal{V}_i : i \in \mathcal{E}\}$ is a stochastic set of clock structure for generating lifetime of events, being $\mathcal{V}_i$ a non-negative random variable.

In this work, the transition function is not total, that is, quite often some events are not allowed, or not feasible in a given state. This fact leads to the Definition 2.

Definition 2: For a STA $\mathcal{G} = (\mathcal{X},\ \mathcal{E},\ f,\ x_0,\ \mathcal{V})$, we define by Γ the function of feasible events:

$$\Gamma(x) = \{e \in \mathcal{E} \,|f(x, e) \neq \emptyset\}. \tag{1}$$

As a result, we can observe that $\Gamma : \mathcal{X} \to 2^{\mathcal{E}}$, being $2^{\mathcal{E}}$ the power set, that is, the set of all sub-set of $\mathcal{E}$.
To computer simulate the behavior of a STA, we need a numerical representation for the transition function f. To this end, the transition functions f of an automaton is represented by matrix according to Definition 3, in a general way, assuming that $\mathcal{E}$ is a subset of larger event sets denoted by $\mathcal{E}^{tot}$.

Definition 3 (Matrix Representation): Given a STA $\mathcal{G} = (\mathcal{X},\ \mathcal{E},\ f,\ x_0,\ \mathcal{V})$, an event set expressed as a sequence $\mathcal{E}^{tot} = \{1, 2, \ldots, L\}$ such that $\mathcal{E} \subseteq \mathcal{E}^{tot}$ and the state set given by $\mathcal{X} = \{1, 2, \ldots, N\}$, the transition function f is presented by the matrix A, $N \times L$ such that:

$$A(x, e) = \begin{cases} f(x, e) & \text{if } e \in \Gamma(x), \\ 0 & \text{if } e \in \mathcal{E} \text{ and } \notin \Gamma(x), \\ x & \text{otherwise.} \end{cases} \tag{2}$$

In addition, we need to define the input-event set for a state, given a precedent state. This is achieved with the input function as defined in Definition 4.

Definition 4: Consider a STA $\mathcal{G} = (\mathcal{X},\ \mathcal{E},\ f,\ x_0,\ \mathcal{V})$, the input-event set of a state $x \in \mathcal{X}$, given a state $y \in \mathcal{X}$, is defined as:

$$\mathcal{I}(x, y) = \{e \in \mathcal{E} \quad | \quad x = f(y, e)\} \tag{3}$$

To deal with huge models, we use a "divide and conquer strategy" that consists of a decomposition of the whole model into sub-models that interact with each other using common events. This can be done systematically by using the parallel-composition operation as presented in Definition 5.

Definition 5 (Paralel Composition of STA): Parallel composition is a binary operation between two STA's $\mathcal{G}_1 = \left(\mathcal{X}_1,\ \mathcal{E}_1,\ f_1,\ x_{1,0},\ \mathcal{V}_1\right)$ and $\mathcal{G}_2 = \left(\mathcal{X}_2,\ \mathcal{E}_2,\ f_2,\ x_{2,0},\ \mathcal{V}_2\right)$ defined by:

$$\mathcal{G}_1 \parallel \mathcal{G}_2 = \left(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{E}_1 \cup \mathcal{E}_2, f_{1\parallel 2}, \left(x_{1,0}, x_{2,0}\right), \mathcal{V}_1 \cup \mathcal{V}_2\right),$$

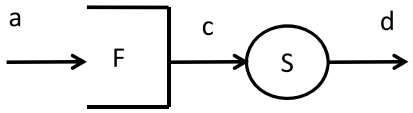

FIGURE 1. Simple manufacturing system with a buffer *F* and a service machine *S*.

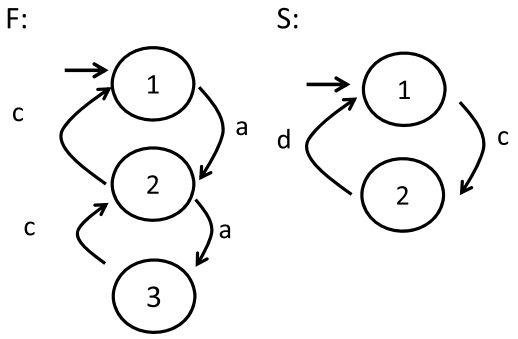

FIGURE 2. Modular automaton representation of the simple manufacturing system depicted in Figure 1.

being

$$f_{1\|2}\left((x_1,x_2),e\right)=\begin{cases}\left(f_1\left(x_1,e\right),f_2\left(x_2,e\right)\right) & \text{if } e\in\Gamma_1\left(x_1\right)\\ & \cap\Gamma_2\left(x_2\right),\\ \left(f_1\left(x_1,e\right),x_2\right) & \text{if } e\in\Gamma_1\left(x_1\right)\\ & \backslash\mathcal{E}_2,\\ \left(x_1,f_2\left(x_2,e\right)\right) & \text{if } e\in\Gamma_2\left(x_2\right)\\ & \backslash\mathcal{E}_1,\\ \emptyset & \text{otherwise.}\end{cases}$$

This operation is associative, that is:

$$\mathcal{G}_1\parallel\mathcal{G}_2\parallel\mathcal{G}_3=(\mathcal{G}_1\parallel\mathcal{G}_2)\parallel\mathcal{G}_3=\mathcal{G}_1\parallel(\mathcal{G}_2\parallel\mathcal{G}_3).\quad(4)$$

Example 1: We consider a simple manufacturing cell, with a waiting space F with capacity for two pieces and a machine S with capacity for processing one piece at a time. We consider the following events a, which represents the arrival of pieces, c, representing piece admission by the machine, and d representing that a piece was processed and has left the system. This system is then divided into two subsystems representing by the modules F and S, whose automaton model can be obtained easily focusing only on such subsystems, as show in Figure 2 As a result, the complete model is represented, in a compact way, by the parallel composition $F||S$. It's important to remark that all states of the system are "embedded" in the automaton resulting from $F||S$ and we do not need to explicit the states. However, if desired, we can explicitly show all the states using the Definition 5. For the present example $F||S$ results in the single automaton shown in Figure 3.

A. ANALYTIC APPROACH

Since exponential distribution is a practical way of generating non-negative random variables, as event lifetime, we can approximate the behavior of an arbitrary STA employing a continuous time Markov chain, as shown in Definition 6.

F||S :

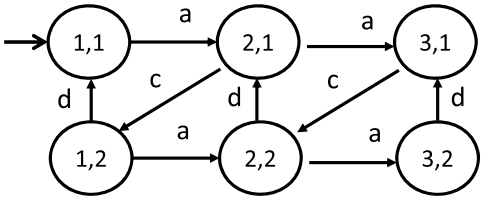

FIGURE 3. Single automaton resulting from *F||S* for the simple manufacturing system.

Definition 6: A continuous time **Markov Chain Approximation (MCA)** $\mathcal{M}_{\mathcal{G}}=\left(\mathcal{X}_{\mathcal{G}},Q,\pi_0\right)$ for a STA, $\mathcal{G}=(\mathcal{X},\mathcal{E},f,x_0,\mathcal{V})$, is given by:

$\mathcal{X}_{\mathcal{G}}$ $=\mathcal{X}$ is the finite set of states;

Q is a square matrix such that[1]

$$\underbrace{Q(x,y)}_{x\neq y}=\begin{cases}\dfrac{1}{\sum E[V_k]} & k\in\mathcal{I}(x,y)(\mathcal{I}(x,y)\neq\emptyset)\\ 0 & \text{otherwise.}\end{cases}$$

$$Q(x,x)=-\sum\frac{1}{E[V_k]},k\in\Gamma(x).$$

π_0 is the vector probability[2] of initial state $x_0\in\mathcal{X}_{\mathcal{G}}$.

As a result, the dynamical evolution of a MCA if expressed by the following autonomous system:

$$\frac{d\pi(t)}{dt}=Q\pi(t),\quad(5)$$

for which $\pi(t)=[\pi_1(t)\ldots\pi_n(t)]'$ is the state probability vector whose $\pi_1(t)=P[X(t)=i]$, being $X(t)$ a random variable that indicates the state of the system at time t.

Quite often the system converge to a steady-state behavior, whose probabilities are constant, leading to:

$$Q\pi(\infty)=0.\quad(6)$$

In addition, we need to add the probability constraint $\sum_{i=1}^{N}\pi_i(\infty)=1$. Since the rows of Q have linear dependence, we can eliminate, for instance, the first row, resulting in a system of the form:

$$\hat{Q}\pi(\infty)=b,\quad(7)$$

being $\hat{Q}$ identical to Q, except for the first row, whose entries are 1, and b the null vector except for $b(1)=1$.

There are several methods to solve Equation 7, among them the simplest one is based on inverse of $\hat{Q}$.

Example 2: Returning to Example 1, we consider the single automaton depicted in Figure 3.

[1]We denote $E[V]$ the expected value of a random variable V.

[2]In this paper, we assume that $\pi_0=1$.

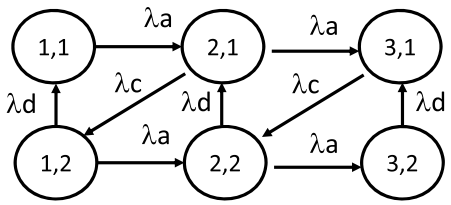

FIGURE 4. Corresponding markov chain for simple manufacturing system whose automaton is depicted in figure 3.

Given the temporal evolution of the system, the state space on the date t is **given by:**

$$X_t = \{(1, 1); (2, 1); (3, 1); (1, 2); (2, 2); (3, 2)\}. \quad (8)$$

If the event lifetimes are random variables, the state of the system at date t is uncertain. Thus, we define the **following probabilities:**

$$\pi_1(t) = P[X_t = (1, 1)], \quad \pi_2(t) = P[X_t = (2, 1)],$$
$$\pi_3(t) = P[X_t = (3, 1)], \quad \pi_4(t) = P[X_t = (1, 2),$$
$$\pi_5(t) = P[X_t = (2, 2)], \quad \pi_6(t) = P[X_t = (3, 2)].$$

We suppose that the lifetimes are generated to respect exponential distributions as follows:

$$P[V_e \leq v_e] = 1 - e^{\lambda_a v_e}, \quad (9)$$

where V_e is the random variables that define the lifetimes for the events e. So in the present example $e \in \{a, c, d\}$.

In this case, we obtain an exact representation of the stochastic automaton as a Markov chain, as shown in Figure 4.

B. MONTE CARLO COMPUTER SIMULATION APPROACH

First of all, we recall that in this case, in general, it is assumed that the system enters into a steady state, more specifically, it is assumed that the associated stochastic process is stationary on average and ergodic. This being the case, the expected value can be estimated by the equation:

$$\theta = \lim_{n\to\infty} \sum_{i=1}^{n} \frac{X_i}{n}, \quad (10)$$

where X_i is the i-th sample of a random variable X. However, in view of the limitation of discrete computing resources, in terms of time and memory, a finite size sample must be considered. In this sense, in general, an estimator $\hat{\Theta}$ is built, that is, a finite approximation for Equation 10 is given by the the following estimator:

$$\hat{\Theta} = \sum_{i=1}^{n} \frac{X_i}{n}. \quad (11)$$

Note that $\hat{\Theta}$ is a random variable that depends on the X_i samples. For this estimator to be an acceptable approximation for $\theta = E[X]$, we need an evaluation of the variance of $\hat{\Theta}$.

Performance evaluation of a STDES through Monte Carlo Computer Simulation demand a lot of human and computational resources, since the models are, in general, quite complex. However the implementation of a computer simulation of a Markov STA is direct due to the memoryless property of the exponential distribution. Indeed, it can be implemented for a given STA $\mathcal{G}$ through a simple Monte Carlo simulation based on generation of uniform random numbers $U \in [0 \quad 1]$ as presented in the Algorithm 1.

Algorithm 1 Markov STDES Simulation

procedure Markov STDES($\mathcal{G}$, N_{max})
  $k = 0$;
  while $k \leq N_{max}$ **do**
    for $i \in \Gamma(x_k)$ **do**
      $U = rand \quad (U \in (0 \quad 1])$;
      $T_i = E[T_i] \times ln(U)$;
    end for
    $T_{min} = Min\{T_i\}$;
    $T_{k+1} = T_k + T_{min}$;
    $e_k = Arg_i\, Min\{T_i\}$;
    $x_{k+1} = f(x_k, e_k)$;
    $k \leftarrow k + 1$;
  end while
  return $\{x_k, e_k, T_k\}$
end procedure

III. MARKOVIANIZATION WITH MARKOV JUMP MODULES

In Section II, we defined the basic elements of STDES that we are interested in, as well as the operation of parallel composition of sub-models. We also show how to analyze the dynamics of STDES using analytic or Monte Carlo computer simulation approaches. Based on those concepts, we show In this section how to construct Markov jump modules in order to transform a non-markovian STDES into a markovian one.

First, in order to have a general automaton representation for the timing mechanism, we construct an automaton T_e as depicted in Figure 5. For this automaton all event lifetime are exponentially generated as:

- the jump rate off state X into I being λ;
- the jump rate from state I to $(k, 1)$ being $p_k\lambda$;
- from state (k, j) given by λ_{jk};
- and from state E given by λ.

So if we have an event that causes a "non-markovian jump", let's say from state "X" to "Y", the operation of parallel composition insert automatically several intermediary states whose transitions operate with markovian jumps in such a way that the " big jump" between "X" and "Y" respects the desired distribution. This operation must be done for each event e that does not respect an exponential distribution. In practical situations, minimal topology can be derived by matching the mean and variance of the collected data as we show in the following.

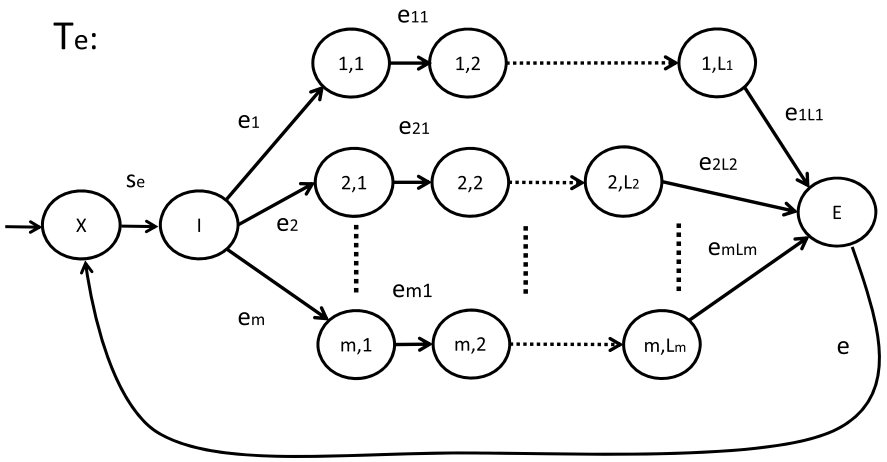

FIGURE 5. Automaton T_e that implements general distribution as a timing mechanism, in which states X and E represent respectively the origin and the destination of an event.

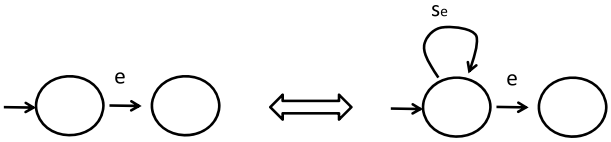

FIGURE 6. Adding the event s_e that indicates the starting of the jumping process.

As a result, if we denote the corresponding timing STA for each event e_i by T_i, the complete timing mechanism is therefore given by the parallel composition:

$$T_m = T_1 \parallel \ldots \parallel T_p. \tag{12}$$

In addition, if the starting of the jumping process is indicated with an event, let's say s_e, we must include this event in the alphabet of the system by making it feasible whenever event e is. This can be done by inserting an appropriate trigger mechanism as shown in Figure 6.

Therefore given an STA, denoted as $\mathcal{G}$, with general lifetime distribution for some events, lets say $e \in \mathcal{E}_t = \{1, \ldots p\}$, we create and associate starting events $s_e \in \mathcal{E}_s$ for those events, resulting the set of pairs:

$$\mathcal{P} = \{(e, s_e) | e \in \mathcal{E} \text{ and } s_e \in \mathcal{E}_s\} \tag{13}$$

Then we construct a modified STA by adding self-loops of these starting event e_s in all states x of $\mathcal{G}$ for which event "e" is feasible, *i.e.* $e \in \Gamma(x)$. This procedure of insertion self-loops is illustrated in in Figure 6 and results in a new automaton, which we denote as $\mathcal{T}(\mathcal{G}, \mathcal{P})$. This transformed STA is presented formally in Definition 7

Definition 7: Given $\mathcal{G} = (\mathcal{X}, \mathcal{E}, f, x_0, \mathcal{V})$ and the set of pairs $\mathcal{P}$, we define the **Triggered STA** as:

$$\mathcal{T}(\mathcal{G}, \mathcal{P}) = (\mathcal{X}, \mathcal{E} \cup \mathcal{E}_t, f_t, x_0, \mathcal{V}_t) \tag{14}$$

for which:

- $f_t(x, e') = \begin{cases} f(x, e') & \text{if } e' \in \Gamma(x), \\ x & \text{if } e' = s_e \text{ and } e \in \Gamma(x). \end{cases}$
- $\mathcal{V}_t = \mathcal{V} \cup \mathcal{V}_s$;

being $\mathcal{V}_s$ the lifetime structure for the starting events and $\Gamma(x)$ the set of feasible events for a given state x of the automaton $\mathcal{G}$.

As a result the the complete STA, with markovian jumps, denoted as $\mathcal{G}_{Markov}$, is given by the parallel composition of triggered STA and the corresponding STA for time mechanism as:

$$\mathcal{G}_{Markov} = \mathcal{T}(\mathcal{G}, \mathcal{P}) \parallel T_m. \tag{15}$$

It is important to remark that the corresponding Markov chain is systematically obtained directly from G_{Markov} by replacing events with their rates. An important advantage of the compact representation of the Markov Chain as given by the Equation 15 is the reduced space complexity, since the operation of parallel composition allows us to represent the whole system through modules. Another interesting feature of this modularity is the flexibility to change the distributions by replacing modules.

A. TIMING MODULES FOR HYPOEXPONENTIAL AND HYPEREXPONENTIAL DISTRIBUTIONS

In a practical situation, hypoexponential and hyperexponential are quite a general way of representing unimodal distribution for time random variables [3]. Important specifications to fit those distributions are the well-known mean (μ) and variance (σ^2) of the input data. In this sense, we will discuss in sequence how can we deal with a practical situation with data described as a hypoexponential distribution, that is $\sigma < \mu$ or a hyperexponential one, that is $\sigma > \mu$. In particular, we will see how to obtain minimal structures ensuring mean and variance matching for those distributions

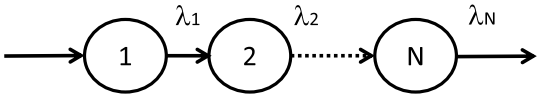

FIGURE 7. Sequence of markov jumps for hypoexponential with N steps. We assume that λ_i the jump rate from step i to step $i+1$.

1) CONVENIENT MINIMAL STRUCTURES FOR MEAN AND VARIANCE MATCHING

The simplified diagram for the states of an Hypoexponential distribution is depicted as shown in Figure 7.

Therefore, our matching problem is achieved by solving the following pair of equation for a minimum N:

$$\begin{aligned}\sum_{i=1}^{N} m_i &= \mu \\ \sum_{i=1}^{N} m_i^2 &= \sigma^2\end{aligned} \tag{16}$$

whose $m_i = 1/\lambda_i$, being λ_i the jump rate from state i to state $i+1$. First, we must observe that triangular and internal product inequalities (Cauchy-Schwartz Inequality) lead to:

$$\frac{(\sum_{i=1}^{N} m_i)^2}{N} \leq \sum_{i=1}^{N} m_i^2 \leq (\sum_{i=1}^{N} m_i)^2 \tag{17}$$

As a result, provided that a solutions exists, they are all such that:

$$\frac{\mu^2}{N} \leq \sum_{i=1}^{N} m_i^2 \leq \mu^2 \tag{18}$$

So the number of states N must satisfy:

$$\frac{\mu^2}{N} \leq \sigma^2 \rightarrow N \geq \frac{\mu^2}{\sigma^2}. \tag{19}$$

Since N is an integer, its smallest possible value is[3] $N = \lceil \frac{\mu^2}{\sigma^2} \rceil$.

Keeping in mind those observations, we present a solution for the system of equations 16. First we observe that if $\frac{\mu^2}{\sigma^2} = 1$, the solution is trivial, with only one state, that is $N = 1$. So let us concentrate our attention in situations whose $\lceil \frac{\mu^2}{\sigma^2} \rceil \geq 2$.

Proposition 1 (Minimal Hypoexponential Distribution): If $\lceil \frac{\mu^2}{\sigma^2} \rceil \geq 2$, a solution for the system of equations 16 is given by:

$$m_j = \frac{\mu}{N} - \frac{\alpha_N}{\sqrt{(N-1)}} \quad (1 \leq j \leq N-1), \tag{20}$$

and

$$m_N = \frac{\mu}{N} + \sqrt{(N-1)}\alpha_N. \tag{21}$$

being $N = \lceil \frac{\mu^2}{\sigma^2} \rceil$, $\alpha_N = \frac{\sqrt{N\sigma^2 - \mu^2}}{N}$.

[3] $\lceil x \rceil$ stands for the ceil of x, *i.e.* smallest integer greater than or equal to x.

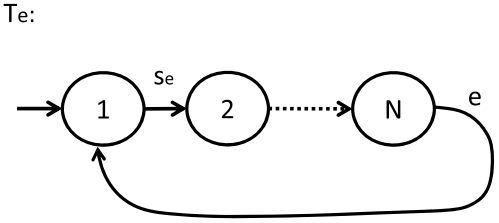

FIGURE 8. STA that implements a hypoexponential distribution as an event timing mechanism for $\sigma < \mu$, whose $\frac{\mu^2}{\sigma^2} \leq N < \frac{\mu^2}{\sigma^2} + 1$.

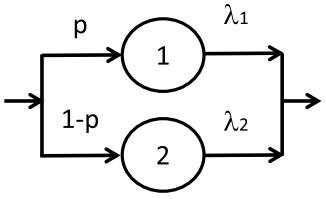

FIGURE 9. Sequence of markov jumps possibilities for hyper-exponential distributions. We denote by p the probability of initially routing to state "1", and by λ_i the jump rate to get out from state i.

Proof: By denoting $m_j = y\,(1 \leq j \leq N-1)$ and $x_N = z$, The system 16 is written as:

$$\begin{cases}(N-1)y + z = \mu, \\ (N-1)y^2 + z^2 = \sigma^2,\end{cases} \tag{22}$$

being $N = \lceil \frac{\mu^2}{\sigma^2} \rceil$. As a result, we can check that:

$$x = \frac{\mu}{N} - \frac{\alpha_N}{\sqrt{(N-1)}} \quad (1 \leq j \leq N-1),$$

and

$$y = \frac{\mu}{N} + \sqrt{(N-1)}\alpha_N$$

are indeed solutions for the pair of Equations 22.

As a result, we obtain the minimum STA as shown in Figure 8.

To couple the time module into the system for a given event e, we do the following:

- Create a Triggered STA according to Definition 7 using the procedure illustrated previously in Figure 6;
- The time scheme is presented by the STA T_e depicted in Figure 8, with jump rates given off state $j \in \{1, \ldots, N\}$ by $\lambda_j = 1/m_j$, being m_j given by Equations 20 and 21.

So far, we have established that if $\sigma^2 \leq \mu^2$, the minimum number of states is given by $N = \lceil \frac{\mu^2}{\sigma^2} \rceil$, being the means between states provided by Proposition 1. It remains to solve the cases for which σ is strictly greater than μ, that is $\sigma > \mu$. We solve this problem by considering two states and two routing probabilities, as depicted in Figure 9. This configuration leads us to an Hyperexponential distribution.

Denoting $m_i = \frac{1}{\lambda_i}$, we can write that:

$$E[T_g] = pm_1 + (1-p)m_2 = \mu, \tag{23}$$

$$E[T_g^2] = 2pm_1^2 + 2(1-p)m_2^2 = \sigma^2 + \mu^2. \tag{24}$$

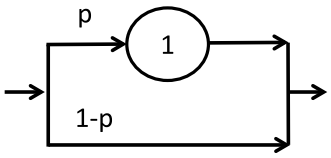

FIGURE 10. Visualization of the structure that implements a hyper-exponential distribution.

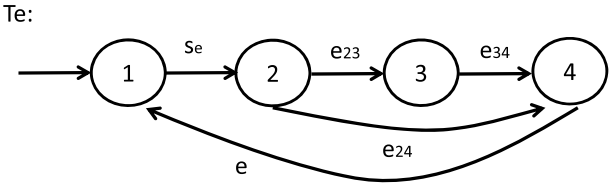

FIGURE 11. Minimum automaton that implements a hyperexponential distribution: Events e_{23} and e_{24} are implements the routing mechanism while e_{34} is the end of the service time.

Considering, without loss of generality that $m_1 \geq m_2$, we solve the system of equation, whose solutions are given by:

$$m_1 = \mu(1 + \frac{\alpha}{p}), \tag{25}$$

$$m_2 = \mu(1 - \frac{\alpha}{1-p}). \tag{26}$$

being $0 < p \leq \frac{2}{1+C_v^2}$, $\alpha = \sqrt{\frac{p(1-p)(C_v^2-1)}{2}}$ and C_v the coefficient of variation, *i.e* $C_v = \frac{\sigma}{\mu}$. So the simplest topology to ensure the desired results is given by choosing $p = p_{max} = \frac{2}{1+C_v^2}$, leading to $m_1 = \mu\frac{C_v^2+1}{2}$ and $m_2 = 0$. This topology, which has only one state, is shown in Figure 10.

Remark 1: An interpretation of the resulting distribution for this topology, in terms of service time, is the following: with probability p, some clients are served with exponentially distributed service time with mean m_1, while others are served with negligible service mean with probability $(1 - p)$.
As a result, we obtain the corresponding minimal automaton implementation for this distribution as depicted respectively in Figure 11.

In order to couple the time mechanism into the system, given event e, we follow a similar procedure as we did for hypoexponential distributions:

- Create a Triggered STA according to Definition 7 using the procedure illustrated previously in Figure 6;
- The time scheme is presented by the STA T_e depicted in Figure 8, with jump rates given off state $j \in \{1, \ldots, N\}$ by $\lambda_j = 1/m_j$, being m_j given by Equations 20 and 21.

IV. COMPLEXITY ANALYSIS OF THE MODULAR MARKOV REPRESENTATION

First, we analyze the memory requirements complexity of the modular model representation, then we analyze the time complexity for performance evaluation using analytic approach as well as computer simulation one.

A. MODEL REPRESENTATION

The complexity Analysis is performed for a model given by a parallel composition of sub-models expressed by Equation 15. Explicitly:

$$\mathcal{G}_{Markov} = \mathcal{T}(\mathcal{G}_s, \mathcal{P}) \parallel T_m. \tag{27}$$

being $\mathcal{G}_s = \mathcal{G}_1 \parallel \ldots \parallel \mathcal{G}_L$ the modular STA model for the system, $\mathcal{P}$ the set of pairs for the non-markovian events, as indicated in Equation 13, and $T_m = T_1 \parallel \ldots \parallel T_p$ the modular STA representation for the timing mechanism.

If we denote $|\mathcal{H}|$ the number of states of an arbitrary STA $\mathcal{H}$, the space complexity for the implicit model expressed by the parallel composition is linear in terms of the number of the sub-models:

$$\mathcal{O}(\sum_{i=1}^{L} |\mathcal{G}_i| + \sum_{i=1}^{p} |T_i|). \tag{28}$$

On the other hand, for the explicit (monolithic) model, for which we explicitly represent all states, the space complexity is exponential:

$$\mathcal{O}(\prod_{i=1}^{L} |\mathcal{G}_i| \times \prod_{i=1}^{p} |T_i|). \tag{29}$$

Therefore, comparing Equations 28 and 29, we can see that parallel representation saves much more memory than the explicit one.

Regarding time complexity for performance evaluation, in the following we compare the worst-case computational complexity for analytic as well as Monte Carlo computer simulation approaches.

1) ANALYTIC APPROACH

In this case the method is base on matrix inversion. So the worst-case complexity is given by:

$$\mathcal{O}((\prod_{i=1}^{L} |\mathcal{G}_i| \times \prod_{i=1}^{p} |T_i|)^3), \tag{30}$$

However, in practice, the matrices are very sparse for large instance problems, since the number of events is quite small in comparison with the number of states. Therefore the computational complexity can be much more smaller depending on the numerical method.

Numerical and Symbolic solution methods are presented in [3]. Numerical Methods for solving large sparse linear equation systems can be found in [19]. Symbolic representations and analysis of large system can be found in [20].

2) COMPUTER SIMULATION APPROACH

This method requires intensive computer iteration as we can observe in Algorithm 1. The number of iterations depends

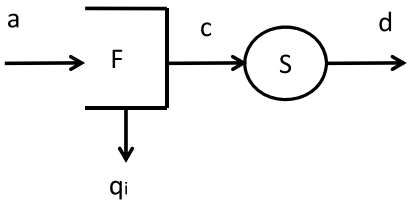

FIGURE 12. Service network with customers abandonment: Event "a" represents that a customer arrives to the queue F; "c" customer admission by the server S; "d" departure of customer from the server after receiving the service. The customer abandonment when the queue F has length i is represented by event "q_i".

on the desired precision. For instance, if we are interested in estimating a given expected value, we observe in Equation 10 and 11 that the large the number of iterations the better is the approximation. In practice, given a desired confidence interval, the worst-case complexity is written as:

$$\mathcal{O}((\prod_{i=1}^{L}|\mathcal{G}_i| \times \prod_{i=1}^{p}|T_i|) \times N_{ev} \times N_{sp} \times N_{rn}). \quad (31)$$

being N_{ev}, N_{sp} $\times N_{rn}$ respectively the number of events, computer iteration and the number of runs.

We can observe that the complexity depends on sample number N_{sp}, as well as on the number of runs N_{rn}. These parameters depend on a previously established confidence interval. The sample number is in general very large (thousands and even millions of samples) and the number of runs is usually small (some units). Therefore, the approach requires intensive use of computer time, but, for large systems, it is simpler to implement than analytic approach.

V. NUMERICAL RESULTS: SERVICE NETWORK WITH CUSTOMERS ABANDONMENT

We evaluate the methodology with a study of a Service Network with Customers Abandonment whose service time is non-markovian but specified in terms of mean and variance. For the representation of this system, we consider two main subsystems: a queue, which is a space where the customers wait for the service, and a server that can process one customer at a time. In this system, customers conditions are sensitive to queue waiting time, in the sense that the longer the queue length the higher the abandonment rate. A block diagram of this system with the event descriptions is depicted in Figure 12.

The correspondent modular automaton model for the system is shown in Figure 13.

To numerically evaluate the performance of this system in terms of the overall abandonment rate of parts, we consider that all event lifetimes follow an exponential distribution, except for the service time, which is specified in terms of a hypoexponential distribution. We consider that the abandonment rates that are proportional to the queue length.

In the present example, we consider that the maximum queue length is $M = 10$ and the mean lifetime for

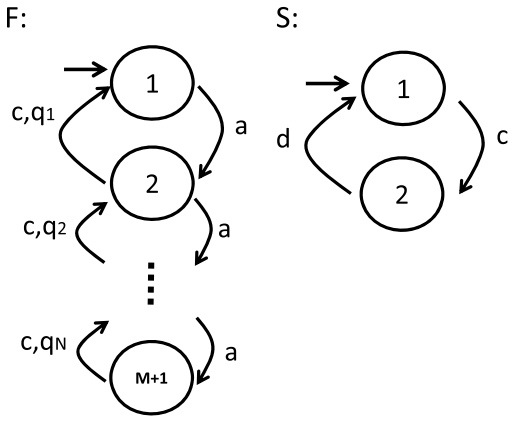

FIGURE 13. Automaton model for service network with customers abandonment for a queue with length (capacity) M.

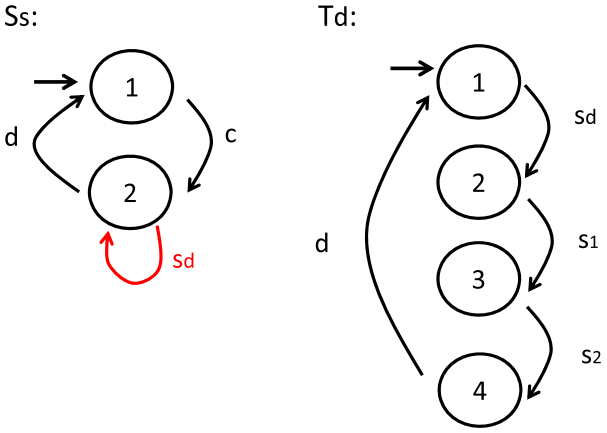

FIGURE 14. Triggered automaton, given by $S_s = \mathcal{T}(\mathcal{G}, \mathcal{P})$, and automaton T_d that implements the time mechanism for event "d".

exponentially distributed events are: a, $E[V_a] = 1.40$; q_i, $E[V_{q_i}] = 14/i$; c, $E[V_c] = 0.10$.

Unlike the other events, event "d" follow a hypoexponential, with mean $\mu = 2.50$ and standard deviation $\sigma = 1.40$. Therefore Proposition 1 helps us with the construction of Markov STA for timing mechanism with 4 states and events $\{s_d, s_1, s_2, d\}$, whose lifetimes are given by $E[V_{s_d}] = E[V_{s_1}] = E[V_{s_2}] = 0.443$ and $E[V_d] = 1.171$. In the sequence of the procedures, we need to obtain the Triggered STA $S_s = \mathcal{T}(\mathcal{G}, \mathcal{P})$, being $\mathcal{P} = \{(d, s_d)\}$, as explained in Definition 7.

The corresponding STA's that implement $S_s = \mathcal{T}(\mathcal{G}, \mathcal{P})$ and T_d are depicted in Figure 14 The complete STA model is the result of the parallel composition of the modules F, S_d, and Td presented in the Figure 15.

In the sequence, we are interested in the performance evaluation of the system. We show how this can be achieved through analytic approach, as well, as through

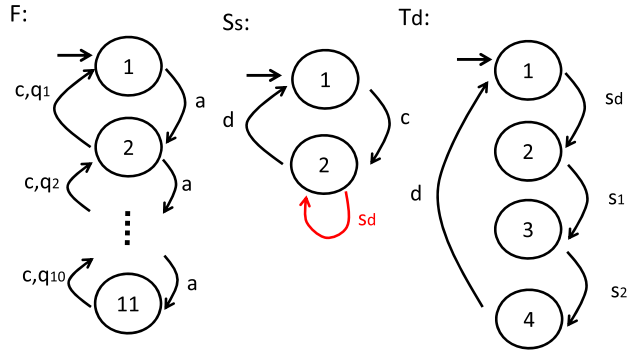

FIGURE 15. Automaton model for service network with customer abandon coupled with a markovian jump module for event "d".

compute simulation, using programs developed in ScicosLab (http://scicoslab.org/) a variant of Scilab.[4]

3) ANALYTIC APPROACH

First, from Figure15, we check that the resulting STA model has $11 \times 2 \times 4 = 88$ states, leading to an equivalent Markov Chain with the same number of states.

In order to evaluate the abandonment rate for the system, we need to compute the throughput rate, that is, the rate of the number of customers that receive a service from the server per time unit. Since we can say that a customer is served whenever an event "d" happens, we compute the number of customers served by the number of times the event "d" occurs. In this sense, we observe that the event of departure, "d" is feasible whenever the server is on state "2" and the Markov module is on state "4" no matter the queue state. As result, the throughput rate Tr is computed using the conditional probability formula:

$$Tr = \sum_{k=1}^{N+1} P[X_F = k, X_S = 2, X_{Td} = 4] \times \frac{1}{E[V_d]}. \quad (32)$$

So the overall abandon rate, Tq is computed by the difference between arrival and throughput rates:

$$Tq = \lambda_a - Tr \quad (33)$$

whose $\lambda_a = \frac{1}{E[V_a]}$ is the arrival rate.

As a consequence, we obtain an abandonment percentage rate $100 \times \frac{Tq}{\lambda_a} = 33,50\%$. The computation time was 0.4516 s in a DELL Computer Intel I7, 3.10Ghz, 8 GB RAM, Windows 8 Pro with 64 bits.

For the sake of comparison, in the sequence, we do the same analysis through Computer Simulation Approach, using ScicosLab scicoslab.org, a variant of Scilab, to develop the programs.

[4]Simulator's code is available on GitHub in https://github.com/AndreyM14/Marvianization-Stochastic-TDES.git

4) COMPUTER SIMULATION APPROACH

Using the Algorithm1, with the same STA used previously in analytic approach, as depicted in Figure15, we do the computer simulation.

We performed the simulation until the conclusion of the service for 10000 pieces. To eliminate the transient behavior, the throughput was computed only after the 400th piece departure. We ran the simulation 10 times, to compute a confidence interval (IC) of 95% using t-Student distribution.

As a result, we obtain the IC for the abandonment percentage rate as [33.37% 33.60%], being the computation time 104.49 s in the same DELL Computer Intel I7, 3.10Ghz, 8 GB RAM, Windows 8.1 Pro with 64 bits.

Remark 2: Comparing the results obtained through the two approaches, that is, the abandonment percentage rate of 33, 50% for analytic and an IC of [33.37% 33.60%] for the computer simulation, we observe that they are quite coherent. However, in terms of computation time, we obtained 0.4516 s for analytic approach versus 104.49 s for computer simulation one, that is computer simulation is more than 200 times slower.

A. DISCUSSION

The model used in the numerical example had a moderate complexity, but we think that it was enough to show the potential of the methodology since we could observe the flexibility in the model construction by changing the whole model by simply replacing modules. For instance, changing the queue capacity, or timing mechanism, is a matter of replacing the respective module for the system. Moreover, despite its apparent simplicity, the total number of states for the example is 88, that is, it is difficult to visualize (or to even draw) the whole model resulting from the compositions of the modules. On the other hand, modular models for the queue, server and timing mechanism had respectively only 11, 2 and 4 states. We remark that the analytic approach gives us real values instead of interval estimates. For systems, with a small number of modules, it is faster, since the number of samples required in the computer simulation approach to generate a small confidence interval is usually very large (thousands or even millions of samples). However, computer simulation algorithms are simpler to implement.

VI. CONCLUSION

In this paper, we presented a complete methodology to analyze the behavior of stochastic discrete-event systems represented using STA, being the whole system expressed as a parallel composition of sub-systems or modules. We have shown how to incorporate timing modules to describe general non-exponential distributions for event lifetimes. In particular, we have shown how to design timing modules for hypo-exponential and hyperexponential distributions. Therefore, we obtained a complete description of the system dynamics by the parallel composition of individual modules that interacts with each other through events, being dynamics operating through markovian jumps among states. With the

aid of a numerical example we show how the task of obtaining the whole model of a system could be drastically simplified through the utilization of modular models for the subsystems, as well as for the timing mechanism. We can observe the versatility of the representation as we could analyze the dynamic behavior of the system by analytic or by Monte Carlo computer simulation methods. Finally, we remark that other application perspectives of the methodology, in terms of performance evaluation of STDES, include computing probabilities of accessing states, blocking, failures, or executing a given sequence of events, which is useful in the analysis and decision of system issues in areas such as control, diagnosing, maintenance and security. Another interesting works to be exploited in future works are an automated tool to convert from STDES to STA or formal system verification.

REFERENCES

[1] R. Ammour, E. Leclercq, E. Sanlaville, and D. Lefebvre, "Fault prognosis of timed stochastic discrete event systems with bounded estimation error," *Automatica*, vol. 82, pp. 35–41, Aug. 2017.

[2] I. W. Bates, A. Karimoddini, and M. Karimadini, "Learning a partially-known discrete event system," *IEEE Access*, vol. 8, pp. 61806–61816, 2020.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Triverdi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2006.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Cham, Switzerland: Springer, 2008.

[5] Q. Chen, L. Yin, N. Wu, M. A. El-Meligy, M. A. F. Sharaf, and Z. Li, "Diagnosability of vector discrete-event systems using predicates," *IEEE Access*, pp. 2169–3536, 2019.

[6] R. Devaraj, A. Sarkar, and S. Biswas, "Real-time scheduling of non-preemptive sporadic tasks on uniprocessor systems using supervisory control of timed DES," in *Proc. Amer. Control Conf. (ACC)*, Seattle, WA, USA, May 2017, pp. 3212–3217.

[7] R. Devaraj, A. Sarkar, and S. Biswas, "Optimal work-conserving scheduler synthesis for real-time sporadic tasks using supervisory control of timed discrete-event systems," *J. Scheduling*, vol. 24, pp. 8–69, Feb. 2021.

[8] M. Farsi, J. A. Erkoyuncu, D. Steenstra, and R. Roy, "A modular hybrid simulation framework for complex manufacturing system design," *Simul. Model. Pract. Theory*, vol. 94, pp. 14–30, Jul. 2019.

[9] V. M. Gonçalves, "Max-plus approximation for reinforcement learning," *Automatica*, vol. 129, pp. 1–9, Jul. 2021.

[10] V. M. Gonçalves, C. A. Maia, and L. Hardouin, "On max-plus linear dynamical system theory: The regulation problem," *Automatica*, vol. 75, pp. 202–209, Jan. 2017.

[11] A. G. C. Gonzalez, M. V. S. Alves, G. S. Viana, L. K. Carvalho, and J. C. Basilio, "Supervisory control-based navigation architecture: A new framework for autonomous robots in Industry 4.0 environments," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1732–1743, Apr. 2018.

[12] M. Goorden, J. V. D. Mortel-Fronczak, M. Reniers, W. Fokkink, and J. Rooda, "Structuring multilevel discrete-event systems with dependence structure matrices," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1625–1639, Apr. 2020.

[13] Z. Jakovljevic, V. Lesi, and M. Pajic, "Attacks on distributed sequential control in manufacturing automation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 775–786, Feb. 2021.

[14] J. Komenda, S. Lahaye, J.-L. Boimond, and T. van den Boom, "Max-plus algebra in the history of discrete event systems," *Annu. Rev. Control*, vol. 45, pp. 240–249, Jan. 2018.

[15] A. C. Lisboa, F. H. B. De Souza, C. M. Ribeiro, C. A. Maia, R. R. Saldanha, F. L. B. Castro, and D. A. G. Vieira, "On modelling and simulating open pit mine through stochastic timed Petri nets," *IEEE Access*, vol. 7, pp. 112821–112835, 2019.

[16] X. Liu, H. Zhang, J. Lin, X. Chen, Q. Chen, and N. Mao, "A queuing network model for solving facility layout problem in multifloor flow shop," *IEEE Access*, vol. 10, pp. 61326–61341, 2022.

[17] A. M. Marrota, V. M. Gonçalves, and C. A. Maia, "Tropical lexicographic optimization: Synchronizing timed event graphs," *Symmetry*, vol. 12, pp. 1–18, Sep. 2020.

[18] T. Masopust and X. Yin, "Complexity of detectability, opacity and A-diagnosability for modular discrete event systems," *Automatica*, vol. 101, pp. 290–295, Mar. 2019.

[19] R. Mehmood and J. Crowcroft, "Parallel iterative solution method for large sparse linear equation systems," Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-650, Oct. 2005.

[20] A. Miner and D. Parker, "Symbolic representations and analysis of large probabilistic systems," in *Validation of Stochastic Systems*. Cham, Switzerland: Springer, pp. 296–338.

[21] M. L. Neves, L. P. Santiago, and C. A. Maia, "A condition-based maintenance policy and input parameters estimation for deteriorating systems under periodic inspection," *Comput. Ind. Eng.*, vol. 61, no. 3, pp. 503–511, 2011.

[22] R. G. Ribeiro, R. R. Saldanha, and C. A. Maia, "Modeling and portfolio optimization of stochastic discrete-event system through Markovian approximation: An open-pit mine study," in *Proc. Linköping Electron. Conf.*, Dec. 2018, pp. 1–6.

[23] R. G. Ribeiro, R. R. Saldanha, and C. A. Maia, "Analysis of decision stochastic discrete-event systems aggregating max-plus algebra and Markov chain," *J. Control, Autom. Electr. Syst.*, vol. 29, pp. 1–10, Oct. 2018.

[24] R. S. Mendes, L. Hardouin, and M. Lhommeau, "Stochastic filtering of max-plus linear systems with bounded disturbances," *IEEE Trans. Autom. Control*, vol. 64, no. 9, pp. 3706–3715, Sep. 2019.

[25] G. Schafaschek, M. H. de Queiroz, and J. E. R. Cury, "Local modular supervisory control of timed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 934–940, Feb. 2017.

[26] B. De Schutter, T. van den Boom, J. Xu, and S. S. Farahani, "Analysis and control of max-plus linear discrete-event systems: An introduction," *Discrete Event Dyn. Syst.*, vol. 30, no. 1, pp. 25–54, Mar. 2020.

[27] Y. Shang, L. Hardouin, M. Lhommeau, and C. A. Maia, "An integrated control strategy to solve the disturbance decoupling problem for max-plus linear systems with applications to a high throughput screening system," *Automatica*, vol. 63, pp. 338–348, Jan. 2016.

[28] X. Yin and S. Lafortune, "Verification complexity of a class of observational properties for modular discrete events systems," *Automatica*, vol. 83, pp. 199–205, Sep. 2017.

CARLOS-ANDREY MAIA received the B.S. and M.S. degrees in electrical engineering from the Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil, and the Ph.D. degree from Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil, and Université d'Angers, Angers, France. Since 1997, he has been with the Escola de Engenharia, UFMG, where he is a Professor of electrical engineering. His current research interests include modeling, simulation, analysis, control of time, and event-driven dynamic systems.