Efficient Handling of Phase-type Distributions in Generalized Stochastic Petri Nets

Serge Haddad* Patrice Moreaux* Giovanni Chiola†

March 1997

Keywords: Coxian distributions, Phase-type distributions, Stochastic Petri Nets, Tensor algebra.

Abstract

We study the introduction of transitions with Phase-type distribution firing time in (bounded) generalized stochastic Petri nets. Such transitions produce large increases of both space and time complexity for the computation of the steady state probabilities of the underlying Markov chain. We propose a new approach to limit this phenomenon while keeping full stochastic semantics of previous works. The method is based on a structural decomposition of the net. We establish conditions under which this decomposition leads to a tensor expression of the generator of the chain. The tensor expression is used to solve the chain with an iterative method.

Introduction

Generalized Stochastic Petri Nets (GSPNs) integrate two kinds of transitions, namely immediate and exponential. Although this allows to model a large range of systems, the availability of more general distributions of firing time would be very useful for many real life phenomena. Two general directions have been taken in this area: the first one [5, 9] introduces general distributions of firing time with specific conditions, which allows one to extract so called subordinated Markov chains and compute steady state and transient probabilities of the states. The second approach [16, 12, 7] introduces Phase-type (PH) distribution firing time without further restrictions, and compute the resulting embedded Continous Time Markov Chain (CTMC). This needs to precisely define the stochastic semantics of these new transitions and to cope with the "bi-dimensional" complexity induced by the number of states generated: exponential with respect to the size of the net, exponential under specific hypotheses with respect to the number of stages of the Coxian distributions.

In the framework of this last approach, previous works have taken into account the stochastic semantics problem either at the net level [7], or at the underlying CTMC level directly [12]. Unfortunately, the complexity problem has not been tackled.

Our approach deals not only with the stochastic semantics allowing the most general situations but also with the complexity of the Markov chain. We exploit the concurrency structure

*LAMSADE – URA CNRS 825, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 PARIS Cedex 16, FRANCE, Email: {haddad,moreaux}@lamsade.dauphine.fr

†DISI, Università di Genova, via Dodecaneso 35, 16146 GENOVA, ITALY, Email: chiola@disi.unige.it

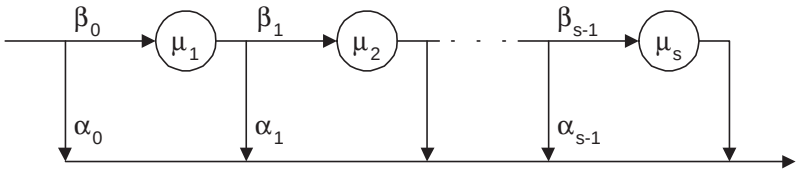


Figure 1: Coxian distribution model

of Petri nets (which are qualitative properties) to obtain a tensor expression of the generator of the underlying CTMC. As long as the impact of PH transitions remains local from the qualitative point of view, the increase of complexity remains achievable: characteristic examples show that we can change from exponential to polynomial complexity.

The outline of the paper is as follows: in Sect. 1 we investigate the problems arising with the introduction of PH and Coxian distribution firing time of transitions (PH and Coxian transitions) in GSPNs. We then formally define the syntax and semantics of bounded GSPNs with such transitions in Sect. 2. In Sect. 3 we explain how to decompose the net to get a tensor expression of the generator of its underlying CTMC. We study the impact of our method on the computational complexity in Sect. 4 and we introduce immediate transitions in Sect. 5 ending with concluding remarks.

1 Dealing with PH transitions in stochastic Petri nets

1.1 PH and Coxian distributions

Coxian distributions [11] generalize Erlang and hyper-exponential distributions which are serial and parallel network of exponential servers. Their model (Fig. 1) is a serial network of s exponential servers (or stages) with service rate μ_i. At the end of service i, clients may leave the global service with probability α_i or enter next stage (if $i < s$) with probability $\beta_i = 1 - \alpha_i$. It is also possible to leave immediately the service with probability α_0. PH distributions [17] are another family of distributions which may be used to approximate many distributions. A PH distribution is the time until absorption of a CTMC with $s + 1$ states and only one absorbing state. It may be also seen as an exponential servers network (Fig. 2), the initial distribution of the chain being $(c_{O,j})_{1 \leq j \leq s}$ and the absorbing state being the state 0. The Laplace transform of a PH distribution is a rational function and it is known that any probability distribution with rational Laplace transform may be (non uniquely) interpreted as a PH distribution, so that many real life phenomena may be modelled with such distributions and their introduction in GSPNs enlarges the usability of this model. In the rest of this paper, we shall only deal with Coxian distributions which are a special class of PH distributions, but the proposed method can be extended straightforward to the general case.

1.2 Stochastic semantics of SPNs

The stochastic semantics of SPNs, also called the *execution policy* [1], first defines which is the transition to be fired in a given marking: in this paper we follow the *race policy* [2, 1]: the transition to fire among the enabled transitions is the one with the smallest remaining firing time. The two other main components of the execution policy are the memory policy and

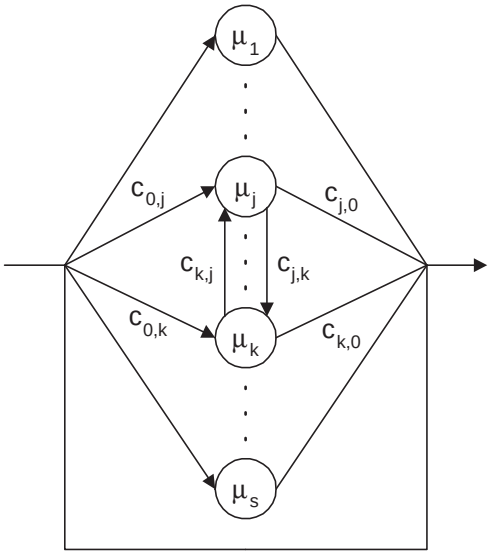


Figure 2: Phase-type distribution model

the service semantics. With Coxian distributions, the interruption/resume policy must also be specified.

1.2.1 Memory policy

As Coxian distributions are no more memoryless, it is necessary to specify how the remaining firing time of a non selected enabled transition is computed. There are three main possibilities.

Resampling: elapsed time is lost and a new sampling of firing delay will be done the next time the transition will be enabled. Resampling allows to model conflicting activities one of which fully disables the others like failures.

Enabling Memory: elapsed time is kept *if the transition stays enabled.* Time-out transition for example may have this memory policy: as long as it is enabled, its timer is decreased. If watch-dog conditions are no more true, the time-out is stopped.

Age Memory: elapsed time is kept whatever the evolution of the system, that is to say: the next time the transition will be enabled, the remaining firing time will be the *residual time* from the last disabling and not a new sampling. Time-sharing service may be modelled by age memory policies.

From a practical point of view, resampling policy has only meaning for conflicting transitions [2], and in this case, it may be modelled with enabling policy and immediate transitions, so that we shall not deal further with resampling in the paper.

1.2.2 Service semantics

The service semantics defines what happens about the service, when the Enabling Degree (ED) of the transition is greater than 1, i.e., when several clients may be served. In standard GSPNs, the firing rate of the transition is used to translate the service policy. If we denote by $\theta(t)$ the rate of t and $\theta(t, \mathbf{m})$ its firing rate in the marking $\mathbf{m}$, there are three usual situations:

Service Policy $\mathrm{sp}(t)$	Memory Policy $\mathrm{mp}(t)$	Interrupt Policy $\mathrm{ip}(t)$
Single Server (S)	Enabling (E)	n.a.
	Age (A)	
K-Servers (K) or Infinite Servers (I)	Enabling (E)	Last engaged (F)
		Least engaged (L)
	Age (A)	Last engaged (F)
		Least engaged (L)

Table 1: Summary of Execution policies

Single Server: only one client may be served at a time. The firing rate $\theta(t, \mathbf{m})$ is $\theta(t)$. This is the default semantics for GSPNs.

Multiple (K) Servers: at most K clients may be served simultaneously. The firing rate is now $\theta(t, \mathbf{m}) = \min(K, \mathrm{ED}(t, \mathbf{m})).\theta(t)$ (the Single Server policy may be seen as the special case $K = 1$ of the Multiple Servers policy).

Infinite Servers: all clients able to be served are served. We have here $\theta(t, \mathbf{m}) = \mathrm{ED}(t, \mathbf{m}).\theta(t)$.

Note the very important fact that, for Coxian transitions with service policy not Single Server, we shall have to *retain in which stages are the clients being served* and to know what stage ends its service to get the firing rate of the transition in a given marking.

1.2.3 Interruption/resume policy

If several clients are served in the stages of a Coxian transition, we have to define how to select the clients to be interrupted if necessary. Two choices, at least, correspond to standard solutions: the most recently entered client (*Last engaged* policy, denoted by F for its connection with the FIFO policy) or the client nearest from the first stage (*Least engaged* policy, denoted by L) is interrupted. Furthermore, if the memory policy is Age Memory, the resume policy which tells which interrupted client is resumed has to be defined. It is clear that it must be chosen accordingly with the interrupt policy (e.g., resume the oldest client or the most engaged one) so that we name interrupt policy both interruption and resume (if needed) strategies. We summarize the different execution policies in table 1: each line gives the three parameters needed ($\mathrm{sp}(t), \mathrm{mp}(t)$ and $\mathrm{ip}(t)$) for every Coxian transition.

1.2.4 Complexity of the stochastic model

The size of the Markovian model of GSPNs is roughly the same as the one of the reachability graph of the net, that is to say exponential with respect to cardinalities of the sets of places and transitions. As soon as we introduce Coxian transitions with the different semantics above, the size of the new CTMC will now be the *product* of the previous model size by the cardinalities of *each* of the state space sizes of the Coxian transitions. With Multiple or Infinite Servers policies (and far more with Age Memory policy and FIFO strategy) this increases hugely the size of the model which leads to find efficient method to solve it.

1.3 Previous works

Since the introduction of stochastic Petri nets, several works have tried to overcome the exponential restriction allowing more general distributions. Two main lines were proposed.

1. General distributions are allowed [9]. If two transitions with general distribution are enabled in a marking and if the firing of one of them does not reset the service time of the other, (like with Age Memory policy or Deterministic firing time for instance) the stochastic process is no more Markovian. Under specific hypotheses, it is however possible to define *regeneration points* of the process which is then semi-regenerative [10]. The steady state resolution is based on the extraction of a so called *embedded Markov Chain* and the study of the process between two regeneration points [5].

2. Restrictions about new distributions are such that the underlying stochastic process remains a CTMC.

In the second case, which is the framework of the present paper, two methods have been studied.

The first one *alters* the initial GSPN with Coxian transitions [16, 7]: each Coxian transition is replaced in the net with a sub-net which translates its stochastic semantics. In [7] the authors propose replacement sub-nets of such transitions with a fixed number of places and transitions whatever the number of stages of the Coxian distribution in contrast with [16]: for instance, a Coxian transition with Age Memory and Single Server policies is expanded with a 8 places, 7 transitions sub-net. This expansion may be automated and integrated in a tool. Note however that only the Single Server policy is studied and that moreover, each Coxian transition must have only one input and one output place.

The second method expandes the embedded Markov chain [12] (the author deals with PH distributions but the principle is the same for Coxian distributions): the net is not modified; the user simply gives the properties of each Coxian transition. These distributions are taken into account only during the reachability graph computation: on one hand, each "marking" holds standard marking information and all information on the states of each Coxian transition (clients in service, their stages); on the other hand, during the computation of firings from a given "marking", these new information are updated accordingly to the stochastic semantics of the Coxian transitions. This method allows the management of any combination of the three elements of the execution policy and may be also integrated in a tool, but at the solver level whereas the first method must be included at a step preceding the solver in the analysis procedure.

The main goal of the present work is to deal with the intrinsic increase in complexity of the Markov chain induced by the introduction of PH or Coxian transitions, a problem which has not been studied in the contributions mentioned above.

2 Definition of generalized stochastic Petri nets with Coxian transitions

In this section we give the formal definition of a generalized stochastic Petri net with Coxian transitions and its stochastic interpretation, that is to say its underlying Markov chain.

2.1 Coxian GSPNs

Definition 2.1 *A Coxian generalized stochastic Petri net is a tuple* $\mathcal{N} = (\mathcal{N}', \theta, \theta_C)$ *where:*

- $\mathcal{N}' = (P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{Inh}, \mathbf{pri})$ *is a Petri net:*
  - P *is the set of places;*
  - T *is the set of transitions;*
  - **Pre**, **Post** *and* **Inh**: $P \times T \to \mathrm{Bag}(P)$ *are the incidence functions*[1]*:* **Pre** *is the input function,* **Post** *the output function and* **Inh** *the inhibition function;*
  - **pri**: $T \to \mathbb{N}$ *is the priority function.*

  and[2] $T = T_E \biguplus T_I \biguplus T_C$ *where* T_E, T_I *and* T_C *are the sets of exponential, immediate and Coxian transitions.*

- θ *and* θ_C *give the stochastic properties of the transitions:*
  - $\theta : T \times \mathrm{Bag}(P) \to \mathbb{R}^+$. $\theta(t, \mathbf{m})$ *is the firing rate (if* t *is exponential) or the weight (if* t *is immediate or Coxian) of* t *in the marking* $\mathbf{m}$.
  - $\theta_C = (s, \mu, \alpha, mp, sp, ip)$ *is only defined for Coxian transitions. It gives for each Coxian transition* t *all specific information needed to unambiguously describe its stochastic semantics:*

    $s(t)$: *the number of stages of* t

    $\alpha_0(t)$: *the probability of immediate end of service of* t

    $\forall 1 \leq i \leq s(t)$

    $\mu_i(t)$: *the rate of the* i*th stage of* t

    $\alpha_i(t)$: *the probability of the end of service after the end of service of the* i*th stage* ($\alpha_{s(t)} = 1$).

    $mp(t)$: *the memory policy of* t (E *or* A)

    $sp(t)$: *the service policy of* t (S, K *or* I)

    $ip(t)$: *the interrupt policy of* t*, if needed* (F *or* L)

The syntax and semantics for standard GSPNs components are left unchanged. For $0 \leq i \leq s(t)$, we denote by $\beta_i(t) = 1 - \alpha_i(t)$ the probability of reaching the $(i+1)$th stage after the end of service of the ith stage of t ($\beta_{s(t)} = 0$).

The choice of an initial marking $\mathbf{m_0}$, defines a marked Coxian generalized stochastic Petri net called also a *Coxian GSPN system* $\mathcal{S} = (\mathcal{N}, \mathbf{m_0})$.

As we are dealing here with the *introduction* of Coxian distributions, we restrict ourselves in this section to Coxian GSPNs without immediate transitions and to Coxian transitions without immediate firing (i.e., $\alpha_0 = 0$). The general case will be studied in Sect. 5 where the exact meaning of **pri** and θ will also be discussed for Coxian transitions.

2.2 Reachability graph of a Coxian GSPN

The reachability graph of a Coxian GSPN is the standard reachability graph of GSPNs where a Coxian transition is seen as an exponential one. During its building, the values of the Enabling Degree with Service constraint

$$\mathrm{EDS}(t, \mathbf{m}) = \begin{cases} \min(K, \mathrm{ED}(t, \mathbf{m})) & \text{if } \mathrm{sp}(t) = K \text{ or } S \\ \mathrm{ED}(t, \mathbf{m}) & \text{if } \mathrm{sp}(t) = I \end{cases}$$

[1] $\mathrm{Bag}(P)$ is the set of multi-sets on P: $\mathrm{Bag}(P) = \{x = (x(p))_{p \in P} \mid \forall p \in P,\ x(p) \in \mathbb{N}\}$

[2] $A \biguplus B$ means $A \bigcup B$ with $A \bigcap B = \emptyset$

are used to compute the greatest total number $e(t)$ of clients which may effectively be in all the stages of a Coxian transition in any marking of the reachability set RS of the net:

$$e(t) = \max_{\mathbf{m} \in RS} \{\mathrm{EDS}(t, \mathbf{m})\}$$

We also assume that $\mathrm{EDS}(t, \mathbf{m})$ may always be accessed during the next steps of the qualitative as well as quantitative analysis of the net, without worrying about this problem in this paper.

2.3 Markov chain of a Coxian GSPN

A state of the chain is an *enlarged* marking which in addition to standard marking, gives for each Coxian transition, its internal state, that is to say in which exponential stages are the clients being served.

We call *descriptor* of the transition the information to memorize which depends on the properties of its stochastic semantics.

Given the descriptors $d(t)$ of the Coxian transitions, a state of the Markov chain is the *extended marking* compatible with a marking $\mathbf{m}$ of the GSPN:

$$\overline{\mathbf{m}} = (\mathbf{m}, (d(t))_{t \in T_C})$$

From RS, we deduce a set of extended markings, adding all possible values of the descriptors. This may generate unreachable states if $\mathrm{mp}(t) = A$ for some transitions.

Before describing the state transitions of the Markov chain, let us give three examples of descriptors adapted to classical execution policies.

2.3.1 Descriptors for policies with identified clients

These are the policies with multiple servers ($\mathrm{sp}(t) = K$ or I) such that the interrupt policy requires to distinguish the clients, e.g., the Last engaged (FIFO) policy ($\mathrm{ip}(t) = F$). We may have $\mathrm{mp}(t) = E$ or A.

The descriptor of these policies (C-descriptor) gives for each client, interrupted or not (if $\mathrm{mp}(t) = A$), its service stage. As we know that $\mathrm{EDS}(t, \mathbf{m}) = 0$ iff no client is being served, we have:

$$d(t) = (d_1, \ldots, d_i, \ldots, d_{e(t)}) \in \mathrm{CD}(t) = \{1, \ldots, s(t)\}^{e(t)}$$

Clients are in their arrival order, from left to right. d_i is the current service stage of the ith client if $i \leq \mathrm{EDS}(t, \mathbf{m})$ or the stage where it is interrupted (and from which it will be resumed) if $i > \mathrm{EDS}(t, \mathbf{m})$; Note that in this way, there is no need to distinguish in the descriptor, the situation where a client i is interrupted in stage 1 and the situation where it is absent: in both cases $d_i = 1$. For example, we have:

$d(t)$	$\mathrm{EDS}(t, \mathbf{m})$	situation
$(3, 1, 2, 7, 1, 1, \ldots, 1)$	2	clients 1 and 2 in service; clients 3 and 4 interrupted
$(3, 1, 2, 7, 1, 1, \ldots, 1)$	5	clients 1 to 5 in service
$(3, 1, 2, 7, 1, 1, \ldots, 1)$	0	clients 1 to 4 interrupted
$(1, 1, \ldots, 1)$	0	no client

These descriptors are the finest ones and may be used with any execution policy.

2.3.2 Descriptors for un-identified clients policies

If there is no need to distinguish clients in the stages of the Coxian distribution, they are only known by the stage in which they are served, as for example with $\mathrm{mp}(t) = E$. We then define the descriptor (S-descriptor) as

$$d(t) = (d_1, \ldots, d_{s(t)}) \in \mathrm{SD}(t) = \{0, 1, \ldots, e(t)\}^{s(t)}$$

If $\mathrm{EDS}(t, \mathbf{m}) > 0$, d_i is the number of clients in service in the stage i. If $\mathrm{EDS}(t, \mathbf{m}) = 0$, $d_i = 0$, $\forall i > 1$ and $d_1 = 1$ (here also we do not distinguish in the descriptor, between no client at all and one client in the first stage). Some examples of descriptor values are:

$d(t)$	$\mathrm{EDS}(t, \mathbf{m})$	situation
$(1, 0, \ldots, 0, 2, 0, 3, 0, 0)$	6	6 clients in service in stages 1, $s-4$ and $s-2$
$(1, 0, \ldots, 0, 2, 0, 3, 0, 0)$	5	5 clients in service in stages $s-4$ and $s-2$
$(1, 0, \ldots, 0)$	1	1 client in service in stage 1
$(1, 0, \ldots, 0)$	0	no client

2.3.3 Descriptors for Single Server policies

In this case, whatever the memory policy, we simply store in the descriptor (SS-descriptor), the fact that some client is or was in service in the stage r:

$$d(t) = r \in \mathrm{SSD}(t) = \{1, \ldots, s(t)\}$$

If $\mathrm{EDS}(t, \mathbf{m}) = 1$, $d(t)$ is the stage of the client in service and if $\mathrm{EDS}(t, \mathbf{m}) = 0$, it is the stage of the interrupted client. For example we have:

$d(t)$	$\mathrm{EDS}(t, \mathbf{m})$	situation
6	1	client in service in the stage 6
6	0	client interrupted in the stage 6
1	1	client in service in the stage 1
1	0	no client

Each of these descriptors may be modified (simplified) when the execution policy allows it: for example, with enabling memory, it is useless to keep information about states of the interrupted clients. Moreover, each Coxian transition has of course, a descriptor adapted to its own stochastic semantics.

It is clear that the C-descriptors need the most memory since $|\mathrm{CD}(t)| = (s(t))^{e(t)}$. The S-descriptors use less memory ($|\mathrm{SD}(t)| = (e(t)+1)^{s(t)}$) and the SS-descriptors catch the minimal information (to be able to manage the Age Memory policy). We see that, for a fixed number of stages of t, the memory used by $\mathrm{CD}(t)$ may grow exponentially with the cardinality of the reachability set of the net, whereas the growing is polynomial for the S-descriptors.

2.4 State transitions of the Markov chain

The transitions (with their rates) between states of the Markov chain may be deduced as for standard GSPNs, from a graph we call the *Extended Reachability Graph* (denoted by $\overline{RG}$): its nodes are the extended markings, and an edge between $\overline{\mathbf{m}}$ and $\overline{\mathbf{m}}'$ corresponds to the firing of a transition of the net. To describe the changes induced on the descriptor of a given Coxian transition t' by the firing of another transition t, we need to distinguish three kinds of t transition firings (for a given extended marking $\overline{\mathbf{m}}$):

Algorithm 2.1 (Building of $\overline{\mathbf{m}} \xrightarrow{t} \overline{\mathbf{m}}'$, $t \in T_E$)

$\mathbf{m}' = \mathbf{m} + \mathbf{Post}(.,t) - \mathbf{Pre}(.,t)$
for each $t' \in T_C$ **do**
  $d'(t') = d(t')$
  (* possible interrupts *)
  if $(\mathrm{EDS}(t',\mathbf{m}') - \mathrm{EDS}(t',\mathbf{m}) < 0) \wedge (\mathrm{mp}(t') = E)$ **then**
    (* definitely interrupt $|\mathrm{EDS}(t',\mathbf{m}') - \mathrm{EDS}(t',\mathbf{m})|$ clients *)
    for i **from** $\mathrm{EDS}(t',\mathbf{m}') + 1$ **to** $\mathrm{EDS}(t',\mathbf{m})$ **do**
      $d'_i(t') = 1$
    endfor
  endif (* nothing to do if $\mathrm{mp}(t') = A$ *)
endfor
add $\overline{\mathbf{m}} \xrightarrow{t} \overline{\mathbf{m}}'$ to $\overline{RG}$

1. t is exponential: we denote by $\overline{\mathbf{m}} \xrightarrow{t} \overline{\mathbf{m}}'$ such a firing;
2. t is Coxian and its firing is the end of service of its rth stage with end of service of the transition. We call *α-firing* such a firing and we denote it by $\overline{\mathbf{m}} \xrightarrow{t_r^\alpha} \overline{\mathbf{m}}'$;
3. t is Coxian and its firing is the end of service of its rth stage with continuation on the following stage. We name *β-firing* such a firing and we denote it by $\overline{\mathbf{m}} \xrightarrow{t_r^\beta} \overline{\mathbf{m}}'$. The marking $\mathbf{m}$ is unchanged in this case.

The stochastic semantics of Coxian GSPNs is given by its $\overline{RG}$. We now provide detailed algorithms to compute $\overline{RG}$ in the case where all Coxian transitions have C-descriptors. Note that such descriptors do not need to be modified when there are new engaged clients. Let $\overline{\mathbf{m}} = (\mathbf{m}, (d(t))_{t \in T_C})$ be an extended marking. The firing $\overline{\mathbf{m}} \xrightarrow{t}$ of an exponential transition t leads to the unique extended marking $\overline{\mathbf{m}}' = (\mathbf{m}', (d'(t))_{t \in T_C})$ which is given by the algorithm 2.1. The rate of this elementary state transition is $\theta(t, \mathbf{m})$.

The firing $\overline{\mathbf{m}} \xrightarrow{t_r^\alpha}$ gives at the most $\mathrm{EDS}(t, \mathbf{m})$ markings $\overline{\mathbf{m}}'$ (one for each possible client in stage r leaving the service) computed by the algorithm 2.2. The corresponding rate is $\mu_r(t).\alpha_r(t)$.

The firing $\overline{\mathbf{m}} \xrightarrow{t_r^\beta}$ provides at the most $\mathrm{EDS}(t, \mathbf{m})$ markings $\overline{\mathbf{m}}'$ (possibly one for each possible client leaving the stage r) (algorithm 2.3) and each elementary state transition has the rate $\mu_r(t).\beta_r(t)$.

Let us emphasis that our goal is precisely *to avoid the direct construction of* $\overline{RG}$ (see below and Sect. 5).

3 Structural decomposition of Coxian GSPNs

The above algorithms show in details the impact of the introduction of Coxian transitions on memory requirements and computational cost of the steady state probabilities of the embedded Markov chain of a Coxian GSPN. The tensor decomposition that we introduce now, is the key point to cope with this problem. Let us recall that tensor decomposition [18, 6, 13, 14] of a Markov chain requires to express the state space of the chain as a *Cartesian product* of

Algorithm 2.2 (Building of the $\overline{\mathbf{m}} \xrightarrow{t_r^\alpha} \overline{\mathbf{m}}'$ from $\overline{\mathbf{m}} \xrightarrow{t_r^\alpha}$)

$\mathbf{m}' = \mathbf{m} + \mathbf{Post}(.,t) - \mathbf{Pre}(.,t)$
for each $t' \in T_C$, $t' \neq t$ **do**
    (* compute $d'(t')$ as in the $\xrightarrow{t}$ case *)
    $d'(t') = d(t')$
    (* possible interrupts *)
    if $(\mathrm{EDS}(t', \mathbf{m}') - \mathrm{EDS}(t', \mathbf{m}) < 0) \wedge (\mathrm{mp}(t') = E)$ **then**
        (* definitely interrupt $|\mathrm{EDS}(t', \mathbf{m}') - \mathrm{EDS}(t', \mathbf{m})|$ clients *)
        for i **from** $\mathrm{EDS}(t', \mathbf{m}') + 1$ **to** $\mathrm{EDS}(t', \mathbf{m})$ **do**
            $d'_i(t') = 1$
        endfor
    endif (* nothing to do if $\mathrm{mp}(t') = A$ *)
endfor
(* discard clients in stage r of t *)
$i = 1$
while $i \leq \mathrm{EDS}(t, \mathbf{m})$ **do**
    if $d_i(t) = r$ **then**
        (* discard this client *)
        $d'(t) = d(t)$
        for j **from** $i + 1$ **to** $e(t)$ **do**
            $d'_{j-1}(t) = d'_j(t)$
        endfor
        $d'_{e(t)}(t) = 1$
        add $\overline{\mathbf{m}} \xrightarrow{t_r^\alpha} \overline{\mathbf{m}}'$ to $\overline{RG}$
    endif
    $i = i + 1$
endwhile

Algorithm 2.3 (Building of the $\overline{\mathbf{m}} \xrightarrow{t_r^\beta} \overline{\mathbf{m}}'$ from $\overline{\mathbf{m}} \xrightarrow{t_r^\beta}$)

$\mathbf{m}' = \mathbf{m}$
for each $t' \in T_C$, $t' \neq t$ **do**
    $d'(t') = d(t')$
endfor
(* advance clients which are in stage r *)
$i = 1$
while $i \leq \mathrm{EDS}(t, \mathbf{m})$ **do**
    if $d_i(t) = r$ **then**
        (* advance this client *)
        $d'(t) = d(t)$
        $d'_i(t) = r + 1$
        add $\overline{\mathbf{m}} \xrightarrow{t_r^\beta} \overline{\mathbf{m}}'$ to $\overline{RG}$
    endif
    $i = i + 1$
endwhile

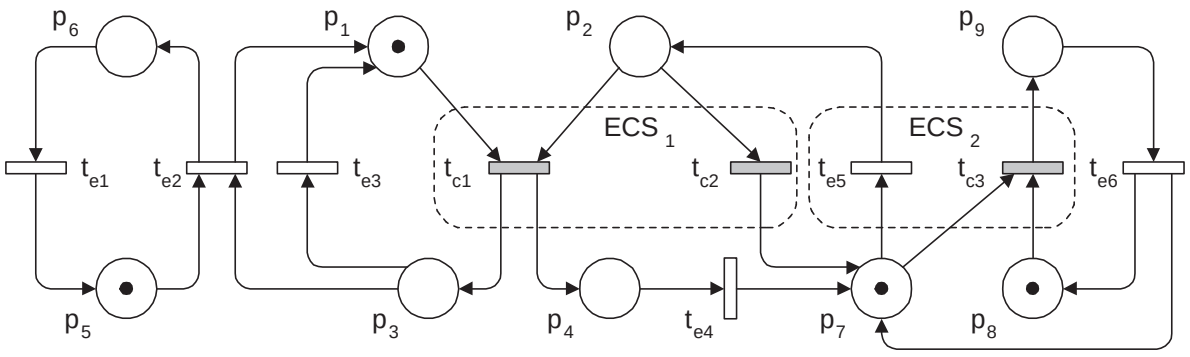


Figure 3: Example of Coxian GSPN

smaller state spaces and to classify state transitions as local, that is to say changing only one component of the state, or global, changing simultaneously several components of the state. Since an extended marking consists of a standard marking and of a tuple of descriptors of the Coxian transitions, we have to introduce a *partition* of the set of places, which allows also to isolate the effects of the transition firings on the descriptors. As we have seen in Sect. 2.4, the descriptor of a given transition t may be modified by a firing of t' iff t' may change $\mathrm{EDS}(t, .)$.

Let us remind that two transitions t and t' are in symmetric structural conflict (SSC) iff the firing of t *may* disable t' or vice versa. The extended conflict sets (ECSs) of the net are the equivalence classes of the transitive closure of the SSC relation [3].

Hence, to ensure that a partition will allow a tensor decomposition, we impose to such a partition conditions based on the extended conflict sets (ECSs) of the net.

Definition 3.1 (Compatible partition) *Let $(ECS_j)_{1 \leq j \leq n}$ be the* ECSs *of a Coxian GSPN. A partition $(P_i)_{1 \leq i \leq m}$ of the set of places P is compatible with respect to Coxian transitions iff*

$$\forall 1 \leq j \leq n, \ \ (ECS_j \bigcap T_C \neq \emptyset \Longrightarrow \exists i_j, \ {}^\bullet ECS_j \bigcup {}^\circ ECS_j \subseteq P_{i_j})$$

where ${}^\bullet ECS_j = \{\mathbf{Pre}^{-1}(.,t) \mid t \in ECS_j\}$ *and* $ECS_j{}^\circ = \{\mathbf{Inh}^{-1}(.,t) \mid t \in ECS_j\}$.

We denote by $\mathbf{m}_i = \mathbf{m}(P_i)$ *the restriction to* P_i *of a marking* $\mathbf{m}$*, so that* $\mathbf{m} = (\mathbf{m}_i)_{1 \leq i \leq m}$.

Let us consider for example the net of Fig. 3 (Coxian transitions have gray boxes). We have two ECSs with Coxian transitions (drawn with dashed line): $ECS_1 = \{t_{C1}, t_{C2}\}$ and $ECS_2 = \{t_{e5}, t_{C3}\}$. So, we must choose a partition such that the input and inhibition places of t_{C1} and t_{C2} (resp. t_{e5} and t_{C3}) belong to *the same* P_i. Note that these conditions allow an automated computation of a compatible partition in polynomial time, namely the ${}^\bullet ECS_j \bigcup ECS_j{}^\circ$, since the ECSs may be computed with polynomial time with respect to the size of P and T.

To distinguish local and global firings, we introduce the sets T_i of transitions which are the only ones which may change the marking of a place of P_i:

$$T_i = {}^\bullet P_i \bigcup P_i{}^\bullet$$

Furthermore, the set of Coxian transitions descriptors of which depend on P_i is

$$T_{C,i} = \{t \in T_C \mid {}^\bullet ECS(t) \bigcup ECS(t)^\circ \subseteq P_i\}$$

Please note that T_i may have Coxian transitions that *are not in* $T_{C,i}$: these are the Coxian transitions which put tokens into P_i without taking token from it.

If we define the (P_i) for the example net as $P_1 = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ and $P_2 = \{p_7, p_8, p_9\}$, we have:

$$T_1 = \{t_{e1}, t_{e2}, t_{e3}, t_{e4}, t_{e5}, t_{C1}, t_{C2}\} \text{ and } T_2 = \{t_{e4}, t_{e5}, t_{e6}, t_{C2}, t_{C3}\}$$

(the other ECSs of $\mathcal{N}$ are: $ECS_3 = \{t_{e2}, t_{e3}\}$, $ECS_4 = \{t_{e1}\}$, $ECS_5 = \{t_{e4}\}$ and $ECS_6 = \{t_{e6}\}$).

3.1 Extended Local reachability graphs

Let $(P_i)_{1 \leq i \leq m}$ be a compatible partition. Instead of building the extended reachability graph $(\overline{RG})$ of the whole net, we build *local*, that is to say restricted to the (P_i, T_i), extended reachability graphs. We first compute the reachability graph of the whole net. $\forall 1 \leq i \leq m$, the set of possible values of the $\mathbf{m}_i$ is denoted by RS_i.

Definition 3.2 (Extended local marking - extended Local reachability graph) *An extended local marking with index* i $(1 \leq i \leq m)$ *of a marking* $\mathbf{m}$ *is*

$$\overline{\mathbf{m}}_i = (\mathbf{m}_i, d(t)_{t \in T_{C,i}}) \quad \textit{with} \quad \mathbf{m}_i \in RS_i$$

and $d(t)$ *a possible value of the descriptor of* t *compatible with its execution policy.*

We denote by $\overline{RS_i}$ *the set of markings* $\overline{\mathbf{m}}_i$.

The i*th extended local reachability graph of* $\mathcal{S}$*, denoted by* $\overline{RG_i}$ *is the extended reachability graph computed as in Sect. 2.4 but restricted to extended local markings of* $\overline{RS_i}$ *and to transitions of* T_i.

3.2 Tensor expression of the generator

Each firing of a transition of the net may be local, that is to say changing only one extended local marking or, global, changing several ones at the same time. As usual with tensor decomposition, we deduce a tensor expression of the generator of the CTMC from this classification. Let us first introduce transitions which change only the extended local markings of their own subnet.

Definition 3.3 (local transition) *A transition t is a local transition with respect to the partition $(P_i)_{1\leq i\leq m}$ iff*

$$\exists i,\ {}^\bullet t \bigcup {}^\circ t \bigcup t^\bullet \subseteq P_i$$

We can know describe local and global firings:

local firings are all firings of local transitions of T_i and all β-firings of non local Coxian transitions of $T_{C,i}$.

global firings are all firings of non local exponential transitions of T_i, as well as α-firings of non local Coxian transitions of $T_{C,i}$.

Let us denote by $T_i^{(L)}$ the set of local transitions of T_i, $T_i^{(X)}$ the set of non local transitions of T_i and $T_{C,i}^{(X)}$ the subset of non local Coxian transitions of $T_{C,i}$. We have:

$$\forall 1\leq i\leq m,\ \ T_i = T_i^{(L)} \biguplus T_i^{(X)} = T_i^{(L)} \biguplus T_{C,i}^{(X)} \biguplus T_i'^{(X)}$$

where $T_i'^{(X)}$ is the set of non local exponential transitions of T_i and of non local Coxian transitions of $T_i \backslash T_{C,i}$. The set of all non local exponential transitions is

$$T' = \left(\bigcup_{i=1}^{m} T_i^{(X)}\right) \backslash T_C$$

From these definitions we can write:

$$T = \left(\biguplus_{i=1}^{m} T_i^{(L)}\right) \biguplus \left(\biguplus_{i=1}^{m} T_{C,i}^{(X)}\right) \biguplus T'$$

From the distinction between local and global firings, we deduce that the generator Q of the embedded CTMC of the net is a sub-matrix of

$$Q' = \bigoplus_{i=1}^{m} Q_i^0 + \sum_{i=1}^{m} \sum_{t\in T_{C,i}^{(X)}} \left[\bigotimes_{j=1}^{m} C_j(t) - \bigotimes_{j=1}^{m} A_j(t)\right] + \sum_{t\in T'} \left[\bigotimes_{j=1}^{m} C_j(t) - \bigotimes_{j=1}^{m} A_j(t)\right] \qquad (1)$$

the tensor sum corresponds to local firings whereas the second and third products correspond to global firings. The expression "sub-matrix" means that if $x = (\overline{\mathbf{m}}_i)_{1\leq i\leq m}$ is a tuple of reachable extended markings, then for any tuple of extended markings y, $q_{x,y} = q'_{x,y}$ if y is reachable, and $q'_{x,y} = 0$ if y is unreachable.

3.2.1 Computation of the Q_i^0

The terms of Q_i^0 are the contributions of all firings of local exponential or Coxian transitions of T_i, and of β-firings of non local Coxian transitions of $T_{C,i}$:

$$q^0_{i,\overline{\mathbf{m}}_i,\overline{\mathbf{m}}'_i} = \sum_{\substack{t\in T_i^{(L)} \bigcap T_E \\ \overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}'_i}} \theta(t,\mathbf{m}_i) \ + \sum_{\substack{t\in T_i^{(L)} \bigcap T_C \\ \overline{\mathbf{m}}_i \xrightarrow{t_r^\alpha} \overline{\mathbf{m}}'_i}} \mu_{r(t)}.\alpha_r(t) \ + \sum_{\substack{t\in T_{C,i} \\ \overline{\mathbf{m}}_i \xrightarrow{t_r^\beta} \overline{\mathbf{m}}'_i}} \mu_{r(t)}.\beta_r(t)$$

for $\overline{\mathbf{m}}_i \neq \overline{\mathbf{m}}'_i$.

Algorithm 3.1 (Computation of the $C_j(t)$ matrices)

1) Case $t \in T_{C,i}$

if $j \neq i$ **then**

if $t \notin T_j$ **then**

$C_j(t) = I_{|\overline{RG}_j|}$

else (* a Coxian transition may be in several T_j *)

$$c_j(t)_{\overline{\mathbf{m}}_j, \overline{\mathbf{m}}'_j} = \begin{cases} 1 & \text{if } \overline{\mathbf{m}}_j \xrightarrow{t} \overline{\mathbf{m}}'_j \in \overline{RG}_j \\ 0 & \text{else} \end{cases}$$

endif

else

(* α-firings of t : end of service of the Coxian distribution *)

$$c_i(t)_{\overline{\mathbf{m}}_i, \overline{\mathbf{m}}'_i} = \begin{cases} \mu_r \alpha_r & \text{if } \overline{\mathbf{m}}_i \xrightarrow{t_r^\alpha} \overline{\mathbf{m}}'_i \in \overline{RG}_i \\ 0 & \text{else} \end{cases}$$

endif

2) Case $t \in T_i \bigcap T'$

if $j \neq i$ **then**

if $t \notin T_j$ **then**

$C_j(t) = I_{|\overline{RG}_j|}$

else

$$c_j(t)_{\overline{\mathbf{m}}_j, \overline{\mathbf{m}}'_j} = \begin{cases} 1 & \text{if } \overline{\mathbf{m}}_j \xrightarrow{t} \overline{\mathbf{m}}'_j \in \overline{RG}_j \\ 0 & \text{else} \end{cases}$$

endif

else

$$c_i(t)_{\overline{\mathbf{m}}_i, \overline{\mathbf{m}}'_i} = \begin{cases} \theta(t) & \text{if } \overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}'_i \in \overline{RG}_i \\ 0 & \text{else} \end{cases}$$

endif

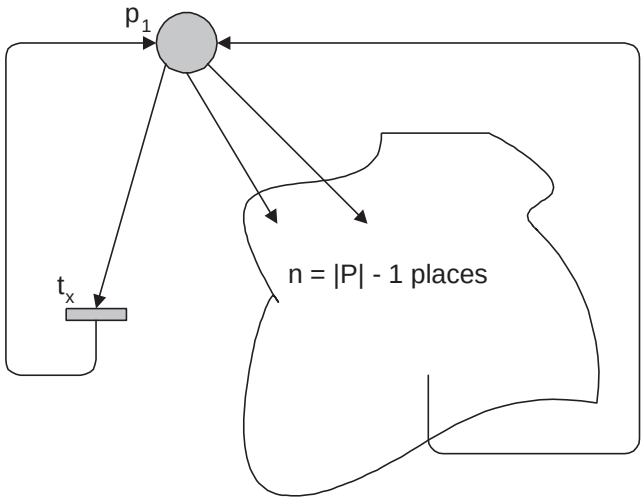


Figure 4: Model for evaluation of the decomposition method

3.2.2 Computation of the $C_j(t)$

The computation of the $C_j(t)$ matrices is given by the algorithm 3.1 in which we have split the Coxian transitions case (second term of Q') and the T' transitions case (third term of Q'). Note that in the first case, we write $\overline{\mathbf{m}}_j \xrightarrow{t} \overline{\mathbf{m}}'_j$ since we do not care about the fact that we have an α-firing of t.

4 Evaluation of the method

We study the savings induced by our method with respect to the two other ones [12] and [7]. Let us first remark that whatever the method, we have to solve the same CTMC. Hence the comparison has to be done at this level: what are the savings using the decomposition method during the steady state probabilities computation of the embedded CTMC instead of the "standard" method?

As complexity measure we simply take the sum of the cardinalities of the involved state spaces (denoted by S for the standard method and D for our method): for large models, this is the most restrictive parameter and the temporal complexity strongly depends on it (moreover, our decomposition method allows an effective parallel computation [19, 15] which in fact may reduce the extra computation time needed by tensor operators). The comparison between the two methods is then carried out studying the ratio $R = \frac{S}{D}$.

The diversity of possible interactions between Coxian transitions and others elements of the net forbids any general comparison. We propose, as a first step, to study a classical situation for two of the policies defined in Sect. 2: Single Server and identified clients. The generic net (Fig. 4) includes only one Coxian transition, t_x controlled by a single place p_1. The maximal marking of p_1 is c. t_x has s stages.The rest of the net consists of $n = |P| - 1$ places, no one controlling a transition in conflict with t_x. We may hence choose the partition $P_1 = \{p_1\}$ and $P_2 = P \backslash P_1$ ($|P_2| = n$).

In all cases we approximate the number of markings of a net with p places and j tokens by its upper bound $\binom{p+j-1}{p-1}$.

4.1 Single Server policies

The memory policy may be Age or Enabling Memory.

With the standard method, we have

$$S \approx s.\left[\sum_{i=0}^{c-1}\binom{n+i-1}{n-1}\right]+\binom{n+c-1}{n-1}$$

the first term stands for situations with at least one client in p_1. As $\binom{n+c}{n}=\sum_{i=0}^{c}\binom{n+i-1}{n-1}$,

$$S \approx (s-1).\left[\sum_{i=0}^{c-1}\binom{n+i-1}{n-1}\right]+\binom{n+c}{n}$$

For the decomposition method, we have

$$D = |\overline{RS}_1| + |\overline{RS}_2| \approx s.c + 1 + \binom{n+c}{n}$$

since $P_1 = \{p_1\}$.

If $n \gg c$, $\binom{n+c}{n} \approx \frac{n^c}{c!}$, so that $R = \frac{S}{D} \approx 1 + \frac{\sum_{i=0}^{c-1}\frac{n^i}{i!}}{\frac{n^c}{c!}}$. Hence $R > 1$, but $\lim_{n\to\infty} R = 1$: intuitively, if n increases faster than c, the influence of the Coxian transition becomes negligible so that the effects of the decomposition method, designed to cope with the Coxian transitions impact, are reduced. The direct method may be used in this simple case.

If $c \gg n$, $R \approx 1 + (s-1)\frac{\sum_{i=0}^{c-1}\frac{i^n}{n!}}{\frac{c^n}{n!}}$ and, in this case, $\lim_{c\to\infty} R = s$: if the number of clients increases faster than the complexity of their behaviour in the net, the influence of their position in the Coxian distribution increases and our method takes advantage of this situation.

4.2 Policies with identified clients

We assume an Age Memory policy.

With the standard method, whatever the value of EDS$(t_x, \mathbf{m})$, there are s^c extended local markings for P_1, in such a way that

$$S \approx s^c.\binom{n+c}{n}$$

and with the decomposition method we have

$$D \approx \binom{n+c}{n} + (c+1).s^c$$

If $n \gg c$, $S \approx s^c.\frac{n^c}{c!}$ and $D \approx \frac{n^c}{c!}$ since $(c+1).s^c << \frac{n^c}{c!}$ so $R \approx s^c >> 1$. For instance, for $s = 2, c = 10$, $R \approx 1000$.

If $c \gg n$, $R \approx \frac{c^{n-1}}{n!}$ and increases very fast with c, for a fixed n: for $n = 10, s = 2$, for instance we have for $c = 10$, $R \approx 200$, for $c = 20$, $R \approx 10^5$ and for $c = 30$, $R \approx 5.10^6$.

Clearly, with Multiple Servers and identified clients policies, the decomposition method is far better than the standard method, by transforming a "product of complexities into a sum".

5 Introduction of immediate transitions

We extend the previous method to Coxian GSPNs with immediate transitions and Coxian transitions with a possible immediate firing i.e., $\alpha_0(t) \neq 0$ (immediate Coxian transitions). This introduces two problems: what is the semantics of immediate Coxian transitions and how to obtain a tensor decomposition of the underlying CTMC of the net?

5.1 Stochastic semantics of immediate Coxian transitions

In this section, we define the supplementary properties of the stochastic model which allows us to fully specify the stochastic process underlying the net.

Since an immediate Coxian transitions may be seen as an immediate transition, its priority must be defined accordingly: it must be 1 if all immediate transitions have priority 1, or else any positive value. The priority of a non-immediate, Coxian transition is 0.

We have three situations of conflict which must be specified.

1. Conflict with at least one immediate transition: from a modelling point of view, the conflict must be solved between immediate firings only. We attribute a *weight* $\theta(t)$ to the immediate Coxian transitions as for standard immediate transitions and solve the conflict like for standard GSPNs. α_0 is not used to solve the conflict.

2. Conflict with exponential or non-immediate Coxian transitions only: we must be able to choose between the α_0 and β_0 firings of t: with probability $\alpha_0(t)$, t fires immediately.

3. Conflict between L immediate Coxian transitions and exponential or non-immediate Coxian transitions: we have $L+1$ possible firings; L immediate firings and one firing where all transitions engage one client in their first exponential stage. Here also, the weights $\theta(t_l)$ allow to select the immediate firing; this choice is then weighted by $\alpha_0(t_l)$.

$$\Pr(\text{immediate firing of } t_l) = \frac{\theta(t_l)}{\sum_{l=1}^{L} \theta(t_l)} \alpha_0(t_l)$$

and

$$\Pr(\text{engagement in the first stage of the } (t_l)_{1 \leq l \leq L}) = \frac{\sum_{l=1}^{L} \theta(t_l)\beta_0(t_l)}{\sum_{l=1}^{L} \theta(t_l)}$$

Let us point out that although the definition of these parameters may seem a blocking to the usability of the method, we expose here the most general case, and in practical situations, the modeller will have to give only a few of them.

5.2 Decomposition of the net

Since a tensor decomposition may only be applied to a CTMC, and as firings of immediate transitions and of α_0 firings of immediate Coxian transitions (Coxian immediate firing) do not occur in the tangible reachability graph (TRG), the modifications resulting from these firings must be local to subsets P_i.

Hence we impose that *all immediate and immediate Coxian transitions are local.* Notice that, given a partition (P_i) it is often easy to change some subsets to transform a non local immediate or immediate Coxian transition into a local one as in Fig. 5 where we replace P_t and P_2 by P_{2t} which makes t a P_{2t} local transition.

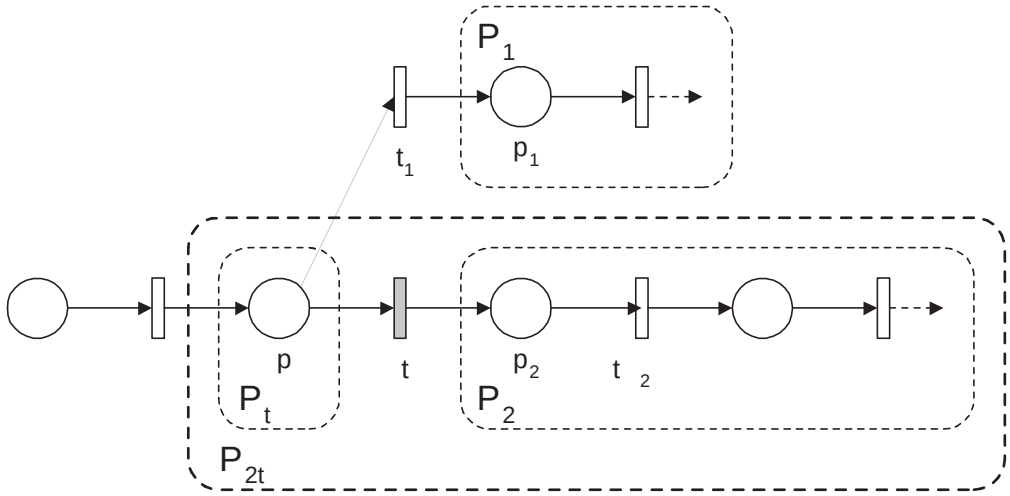


Figure 5: Changing a transition into a local transition

5.3 Solution of the Markov chain

We describe now the modifications of the different steps of the non immediate case. We refer the reader to [3, 4] for a general presentation of the extraction of the embedded CTMC of a GSPN.

The building of the reachability graph takes into account the new semantics. We shall yet call "tangible" a marking from which there is no immediate neither Coxian immediate firing. The edges with non tangible origin have to be labeled with transition probabilities deduced from above discussion.

5.3.1 Computation of the $\overline{TRG_i}$

$\overline{TRG_i}$ is the graph of the tangible extended local markings of the ith sub-net. Note that the same descriptors are used since all states are tangible.

The algorithms are globally the same, but after an exponential or α firing, we compute the sequences of immediate and Coxian immediate firings generated in each of the subnets: in the net of Fig. 6 (top) for instance, the firing of t_{11} (exponential) produces immediate firings inside three subnets. The Fig. 6 (bottom) gives some possible sequences. If

$$\overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}'_i \xrightarrow{\sigma} \overline{\mathbf{m}}_i^{(v)}$$

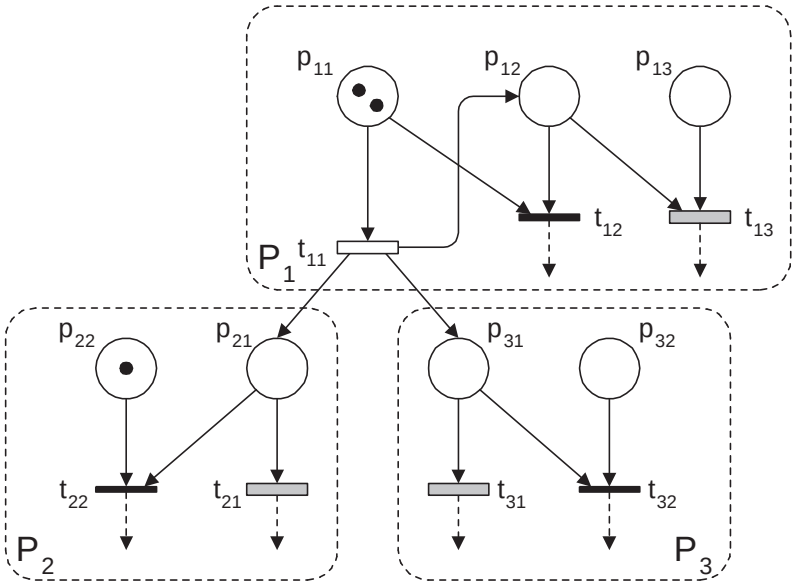
where σ is a sequence of immediate firings, we add to the label of the edge $\overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}_i^{(v)}$ the probability

$$\sum_{\sigma} \Pr(\overline{\mathbf{m}}'_i \xrightarrow{\sigma} \overline{\mathbf{m}}_i^{(v)})$$

of all sequences of immediate firings between these markings. Note also that during these computations, the descriptors of the involved transitions are updated.

5.3.2 Tensor expression of the generator

We have the same expression as for the non immediate case. However, matrices include now the effects of the immediate firings. The corresponding probabilities are retrieved from the $\overline{TRG_i}$.



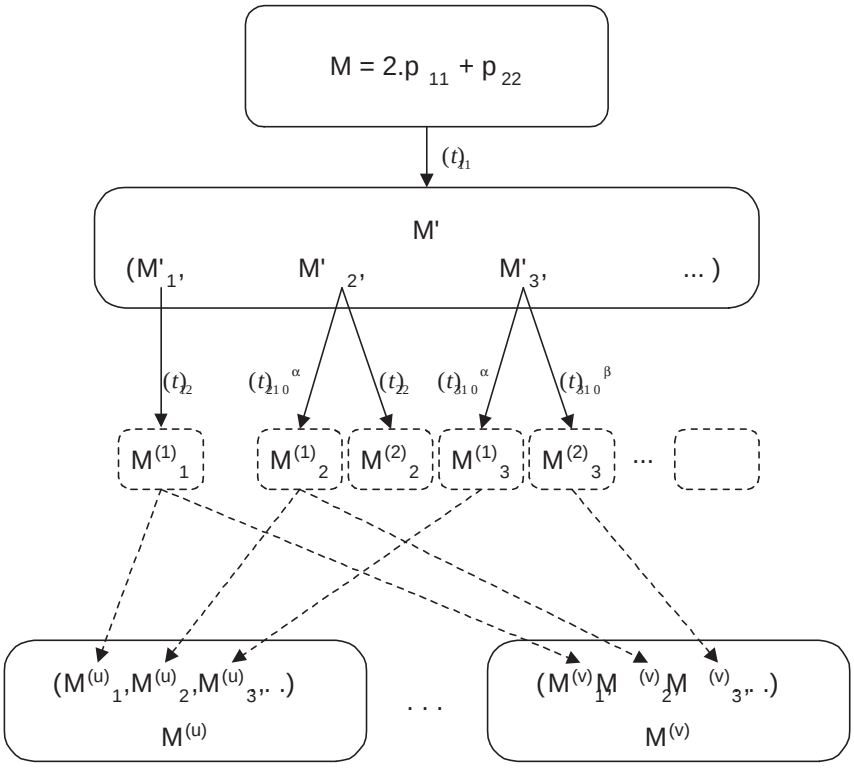


Figure 6: Exponential firing followed by several immediate firings

For the matrices Q_i^0:

$$q^0_{i,\overline{\mathbf{m}}_i,\overline{\mathbf{m}}'_i} = \sum_{\substack{t\in T_i^{(L)}\bigcap T_E \\ \overline{\mathbf{m}}_i \xrightarrow{t} \overline{\mathbf{m}}''_i}} \theta(t,\mathbf{m}_i) \sum_{\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i} \Pr(\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i)$$
$$+ \sum_{\substack{t\in T_i^{(L)}\bigcap T_C \\ \overline{\mathbf{m}}_i \xrightarrow{t^\alpha_r} \overline{\mathbf{m}}''_i}} \mu_{r(t)}.\alpha_r(t) \sum_{\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i} \Pr(\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i)$$
$$+ \sum_{\substack{t\in T_{C,i} \\ \overline{\mathbf{m}}_i \xrightarrow{t^\beta_r} \overline{\mathbf{m}}'_i}} \mu_{r(t)}.\beta_r(t)$$

For the matrices $C_j(t)$ of the algorithm 3.1:
case 1) first `else` block:

$$c_j(t)_{\overline{\mathbf{m}}_j,\overline{\mathbf{m}}'_j} = \begin{cases} \sum_{\overline{\mathbf{m}}''_j \xrightarrow{\sigma} \overline{\mathbf{m}}'_j} \Pr(\overline{\mathbf{m}}''_j \xrightarrow{\sigma} \overline{\mathbf{m}}'_j) & \text{if } \overline{\mathbf{m}}_j \xrightarrow{t} \overline{\mathbf{m}}'_j \in \overline{TRG}_j \\ 0 & \text{else} \end{cases}$$

case 1) second `else` block:

$$c_i(t)_{\overline{\mathbf{m}}_i,\overline{\mathbf{m}}'_i} = \begin{cases} \mu_r\alpha_r \sum_{\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i} \Pr(\overline{\mathbf{m}}''_i \xrightarrow{\sigma} \overline{\mathbf{m}}'_i) & \text{if } \overline{\mathbf{m}}_i \xrightarrow{t^\alpha_r} \overline{\mathbf{m}}'_i \in \overline{TRG}_i \\ 0 & \text{else} \end{cases}$$

Same modifications are applied in the two `else` blocks of the case 2).

Conclusion

We have presented in this paper a new method to introduce PH distribution firing time in generalized stochastic Petri nets. It allows the specification of all types of stochastic semantics usually encountered, including immediate transitions and PH transitions with immediate firing. We have shown how to define a decomposition of the net which allows the derivation of a tensor expression of the generator. This expression reduces the increase in complexity induced by the PH transitions, keeping it of the same order of magnitude as the complexity of the reachability graph of the net when PH transitions are weakly coupled with other parts of the net.

Future work will investigate more deeply the savings provided with respect to the topology of the net and evaluate the possible integration of the method in a tool like [8]. However we would like to stress the novelty of our approach: previous works addressed mainly the problem of being able to define PH distributions in the SPN framework, without providing any suggestion on how to handle the additional complexity induced on the underlying Markov chain. To the best of our knowledge this is the first attempt to relate the introduction of non exponential distributions to the efficient solution techniques based on tensor expressions. On the other hand, tensor algebra based techniques, although potentially very efficient are usually difficult to apply to general SPN structures. The application to PH expansion represents a case in which the partition of the state space in Cartesian products of nearly independent parts is suggested by the structure of the model in a simple way.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, . Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, 15(7):832–846, July 1989.

[2] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri nets with stochastic timing. In *Proc. of the International Workshop on Timed Petri Nets*, pages 80–87, Torino, Italy, July 1-3 1985. IEEE Computer Society Press.

[3] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets revisited: random switches and priorities. In *Proc. of the International Workshop on Petri Nets and Performance Models*, pages 44–53, Madison, Wisconsin, USA, August 24–26 1987. IEEE Computer Society Press.

[4] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized of stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on computer systems*, 2(2):93–122, May 1984.

[5] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, number 266 in LNCS, pages 132–145. Springer–Verlag, 1987.

[6] P. Buchholz. Aggregation and reduction techniques for hierarchical GCSPN. In *Proc. of the 5th International Workshop on Petri Nets and Performance Models*, pages 216–225, Toulouse, France, October 19–22 1993. IEEE Computer Society Press.

[7] P. Chen, S. C. Bruell, and G. Balbo. Alternative methods for incorporating non-exponential distributions into stochastic timed Petri nets. In *Proc. of the third International Workshop on Petri Nets and Performance Models*, pages 187–197, Kyoto, Japan, December 11–13 1989. IEEE Computer Society Press.

[8] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: graphical editor and analyzer for timed and stochastic Petri nets. *Performance Evaluation*, 24(1&2):47–68, 1995.

[9] G. Ciardo, R. German, and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. In *Proc. of the 5th International Workshop on Petri Nets and Performance Models*, pages 170–179, Toulouse, France, October 19–22 1993. IEEE Computer Society Press.

[10] E. Çinlar. *Introduction to stochastic processes*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

[11] D. R. Cox. A use of complex probabilities in the theory of stochastic processes. In *Proc. Cambridge Philosophical Society*, pages 313–319, 1955.

[12] A. Cumani. ESP - a package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proc. of the International Workshop on Timed Petri Nets*, pages 144–151, Torino, Italy, July 1-3 1985. IEEE Computer Society Press.

[13] S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In Robert Valette, editor, *Proc. of the 15th International Conference on Application and Theory of Petri Nets*, number 815 in LNCS, pages 258–277, Zaragoza, Spain, June 20–24 1994. Springer–Verlag.

[14] S. Haddad and P. Moreaux. Asynchronous composition of high level Petri nets: a quantitative approach. In *Proc. of the 17th International Conference on Application and Theory of Petri Nets*, number 1091 in LNCS, pages 193–211, Osaka, Japan, June 24–28 1996. Springer–Verlag.

[15] P. Kemper. Closing the gap between classical and tensor based iteration techniques. In W. J. Stewart, editor, *Proc. of the Second International Workshop on Numerical Solution of Markov Chains*, Raleigh, NC, USA, January 16–18 1995. Kluwer Academic Press, 1995.

[16] M. K. Molloy. *On the integration of delay and throughput in distributed processing models.* PhD dissertation, University of California, Los Angeles, CA, USA, September 1981.

[17] M. F. Neuts. *Matrix-geometric solutions in stochastic models - an algorithmic approach.* The John Hopkins University Press, London, 1981.

[18] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 147–154, Austin, Texas, USA, August 1985. ACM.

[19] B. Plateau and J.M. Fourneau. A methodology for solving Markov models of parallel systems. *Journal of parallel and distributed computing*, 12:370–387, 1991.

Contents