A New Approach to the Evaluation of Non Markovian Stochastic Petri Nets

Serge Haddad[1], Lynda Mokdad[1], and Patrice Moreaux[2]

[1] LAMSADE, UMR CNRS 7024, Université Paris Dauphine
Place du Maréchal de Lattre de Tassigny
75775 PARIS Cedex 16, FRANCE
{haddad, mokdad}@lamsade.dauphine.fr
[2] LISTIC, ESIA-Université de Savoie
Domaine universitaire d'Annecy le Vieux
BP 806, 74016 ANNECY Cedex, FRANCE
patrice.moreaux@univ-savoie.fr

Abstract. In this work, we address the problem of transient and steady-state analysis of a stochastic Petri net which includes non Markovian distributions with a finite support but without any additional constraint. Rather than computing an approximate distribution of the model (as done in previous methods), we develop an exact analysis of an approximate model. The design of this method leads to a uniform handling of the computation of the transient and steady state behaviour of the model. This method is an adaptation of a former one developed by the same authors for general stochastic processes (which was shown to be more robust than alternative techniques). Using Petri nets as the modelling formalism enables us to express the behaviour of the approximate process by tensorial expressions. Such a representation yields significant savings w.r.t. time and space complexity.

1 Introduction

Non Markovian process analysis The transient and steady-state analysis of Markovian discrete event systems is now well-established with numerous tools at the disposal of the modellers. The main open issue is the reduction of the space complexity induced by this analysis. However in a realistic system, the distribution of the occurrence (or the duration) of some events cannot be described by an exponential law (e.g., the triggering of a time-out). Theoretically any "reasonable" distribution is approximated by a phase-type distribution enabling again a Markovian analysis [1]. Unfortunately the continuous time Markov chain (CTMC) associated with this approximation is so huge that it forbids its analysis (indeed even its construction). Such a phenomenon often occurs when the non exponential distribution has a finite support i.e., when the whole probability mass is included in a finite subset of $\mathbb{R}^+$ (non null Dirac, uniform, etc.); then a good phase-type approximation requires too much stages for a close approximation.

Hence the research has focused on alternative methods. In the case of a single realization of a non Markovian distribution at any time, successful methods have been proposed [2] both for the transient and steady state analysis, especially in the Stochastic Petri Net (SPN) modelling framework. Let us cite, for instance, the method of supplementary variables [3,4] or the method of the subordinated Markov chains [5].

The general case (i.e., simultaneous multiple realizations of such distributions) is more intricate. The method of supplementary variables is still theoretically applicable but the required space and the computation time limit its use to very small examples. An alternative approach is described for non null Dirac distributions (i.e., "deterministic" durations) in [6]. The stochastic process, which is a General State space Markov Process (GSMP) is observed at periodic moments of time ($\{h\Delta \mid h \in \mathbb{N}\}$) and this new process is expressed by a system of integro-differential equations and solved numerically. The steady-state distributions of these processes are identical and, with another computation, one obtains the transient distribution of the original process from some transient distribution of the transformed process. This method has been implemented in the DSPNexpress tool [7] (but currently for only two concurrent "deterministic" events with same duration). By imposing conditions on the simultaneous occurrences of concurrent activities, other authors have also designed efficient algorithms [8,9,10,11,12] (see section 6 for more details).

Our previous contribution In [13] we have proposed a different approach to deal with *multiple concurrent* events with finite support distributions. Moreover, in contrast with other works, our solution does not require specific synchronization between these events such as non overlapped or nested events. The main idea is to define an *approximate* model on which we perform an *exact* analysis. To this end, given a time interval (say Δ) we describe the behaviour of the stochastic model by two components: a CTMC and a discrete time Markov chain (DTMC). During an interval $(h\Delta, (h+1)\Delta)$ the behaviour is driven by the CTMC which corresponds to Markovian events occurring in $(h\Delta, (h+1)\Delta)$. Non Markovian activities are taken into account at $h\Delta$ instants only: the corresponding untimed probabilistic changes of state are processed according to a DTMC.

In the approximate process, the Markovian events are in fact exactly modelled since the set $\{h\Delta \mid h \in \mathbb{N}\}$ has a null measure. The approximation comes from non Markovian events: the distribution of a non Markovian event is approximated by a discrete random variable expressing the number of points $h\Delta$ that must be reached before its occurrence. Thus the residual number of points to be met is included in the state of the approximate process. At any moment $h\Delta$, the current residual numbers are decreased and the corresponding events occur when their residues are null.

The approximate process may be analysed either in transient mode or in steady-state. The transient analysis is done by successively computing the state distribution at the instants $\Delta, 2\Delta, \ldots, h\Delta, \ldots$ applying a transient analysis of the CTMC during an interval Δ (via the uniformization technique [14]) followed by a "step" of the DTMC. In order to smooth the effect of the discretization, we average the distribution upon the last interval with a variant of uniformization. Since the asymptotic behaviour of the process depends on the relative position w.r.t. the points $h\Delta$, the approximate process does not admit a steady-state distribution but it is asymptotically periodic. Hence, for the steady-state analysis, one computes the steady-state distribution at the instants $h\Delta$ and then starting from this distribution, one again averages the steady-state distribution upon an interval.

Our current contribution Standard benchmarks (like $M/D/S/K$ queue) have shown that the implementation of our method is at least as efficient as tools like DSPNexpress-NG [15]. Furthermore it is robust, i.e., it still provides good approximations under extreme situations like loaded heavy queues unlike other tools. However, the space complexity is the main limitation of our method since a state includes its logical part and the delays of each enabled non Markovian event. In order to tackle this problem, we start with a high-level description given by a stochastic Petri net with general distributions. Then we define its semantics by specifying the associated stochastic process. Afterwards, we introduce an approximate stochastic process for the stochastic Petri net whose behaviour is structured as described above.

Thus we give tensorial expressions for the infinitesimal generator of the continuous part of our process and for the probability transition matrix of its discrete part. More precisely, these matrices are decomposed into blocks and each block is associated with a tensorial expression. Such a structure is a consequence of the representation of the state space. Indeed rather than (over-)approximating the state space by a cartesian product (as done in the original tensorial methods), we approximate it by a finite union of such products yielding a significant space reduction. A similar approach has been successfully experimented with in [16] for SPNs with phase-type distributions.

Here, we face an additional difficulty. Inside an interval, the delay of each enabled general transitions is non null whereas it can be null at the bounds of the interval. Thus the state space of the discrete part of the process is an extension of the one of the continuous part. Furthermore, the block decomposition must be refined. Indeed inside the interval, states are grouped w.r.t the general enabled transitions. At the bounds, states are grouped w.r.t both the general enabled transitions and the fireable ones (i.e., with null delay). Thus the alternation between the continuous part and the discrete part requires a tricky expansion of the probability vector. Hopefully, the complexity of this operation is negligible w.r.t. the other parts of the computation.

The experimentations are still in progress and their results will be provided in a forthcoming LISTIC technical report. Nevertheless we will detail here some key features of the implementation.

Organization of the paper In section 2 we recall the principle of our approach for a general stochastic process. Then we present the approximate process of a stochastic Petri net in section 3. Afterwards we develop in section 4 the tensorial expressions of the matrices defining the behaviour of this approximate process.

We give information about our implementation in section 5. In section 6, we show that all the previous methods handle particular cases of the systems that we are able to analyse. Finally we conclude and we give indications on future developments of our work.

2 The approximate method

2.1 Principle

Semantics of the approximate process As mentioned in the introduction, we define an *approximate* model (say $Y^{(\Delta)}$) of the initial model (say X), on which we perform an *exact* analysis. The main idea is to choose a time interval Δ and to restrict in $Y^{(\Delta)}$ the non Markovian events to only occur at times $t_h = h\Delta$. We then study in an exact way the evolution of the stochastic process $Y^{(\Delta)}$ in each interval (t_h, t_{h+1}) and during the state changes at time t_h . We stress that the starting times of the active non Markovian events are *in no way related.* We obtain such a model $Y^{(\Delta)}$ from a general model with non Markovian finite

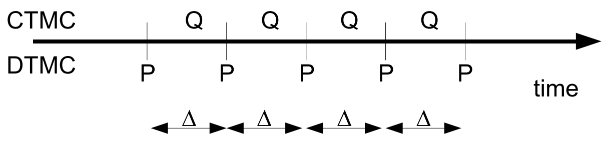

Fig. 1: Time decomposition

support distributions as follows. The distribution of every non Markovian event is approximated by a discrete time distribution lying on points $h\Delta$. Let us note that although Δ seems to be the approximation parameter, the appropriate parameter is the maximum number of points used to express the distribution. Moreover this indicator is the key factor for the complexity of our analysis.

In the approximate process, the Markovian events occur during the intervals $(h\Delta, (h+1)\Delta)$. Non Markovian events always occur in $\{h\Delta \mid h \in \mathbb{N}\}$. Let us describe how they are scheduled. When a non Markovian event is enabled in an interval $(h\Delta, (h+1)\Delta)$ due to the occurrence of a Markovian event, then its approximate distribution is interpreted as the number of points $k\Delta$ that must be met before its occurrence. Here we can choose whether we count the next point (i.e., an under-evaluation of the approximated distribution) or not (i.e., an overestimation of the approximated distribution). The impact of this choice will be discussed later. Thus the residual number of points to be met is included in the state of $Y^{(\Delta)}$. At any moment $h\Delta$, the current residual numbers corresponding to non Markovian events are decreased. If some residues are null then the corresponding (non Markovian) events occur with possibly some probabilistic choice in case of conflicts. The occurrence of these events may enable new non Markovian events. Such events are handled similarly except that the next point is always counted since now it corresponds to a complete interval. If we denote by t_h^- (t_h^+) the "time" before (after) the state change in t_h, the process $Y^{(\Delta)}$ is defined by two components:

– the subordinated process in (t_h, t_{h+1}) associated with states at t_h^+ records only exponential events. It is then a CTMC defined by its generator $\mathbf{Q}$;
– the state changes at t_h are defined by a stochastic matrix $\mathbf{P}[i,j] = \Pr(Y^{(\Delta)}(h\Delta^+) = j \mid Y^{(\Delta)}(h\Delta^-) = i)$.

Thus the Markov Regenerative Process (MRGP) $Y^{(\Delta)}$ is fully defined by its initial probability vector $\boldsymbol{\pi}(0)$ and the matrices $\mathbf{P}, \mathbf{Q}$ (figure 1). These three components depend on Δ since the state space includes the residual number of instants per activated event. We stress however that, even if non Markovian events occur at $h\Delta$, *all kinds of concurrency are allowed between the activities of the system, contrary to previous methods.*

Furthermore, it is important to note that in the approximate process, the Markovian events are in fact *exactly* modelled since the set $\{h\Delta \mid h \in \mathbb{N}\}$ has a null measure. The only approximation comes from non Markovian events: their approximate distribution is interpreted as the number of points $k\Delta$ that must be met before their occurrence.

This approximate process may be analysed either in transient mode or in steady-state. The proposed analysis is an adaptation of the classical Markovian renewal theory methods.

Transient analysis The transient analysis is performed by successively computing the state distribution at the instants $\Delta, 2\Delta, \ldots$ applying a transient analysis of the CTMC during an interval Δ (via the uniformization technique [14]) followed by a "step" of one of the DTMC. In order to smooth the effect of the discretization, we average the distribution on the last interval (with a variant of the uniformization).

Let $\boldsymbol{\pi}(h\Delta^+)$ be the probability vector of the process $Y^{(\Delta)}$ at time $h\Delta$ after the discrete time change, and $\boldsymbol{\pi}_X$ the probability vector of the initial model. We have:

$$\boldsymbol{\pi}(h\Delta^+) = \boldsymbol{\pi}((h-1)\Delta^+)e^{\mathbf{Q}\Delta}\mathbf{P} = \boldsymbol{\pi}(0)(e^{\mathbf{Q}\Delta}\mathbf{P})^h$$

Since we want to smooth the discretization effect, we define the approximate value $\boldsymbol{\pi}^{(a)}(h\Delta)$ of $\boldsymbol{\pi}_X(h\Delta)$ as the averaged value of the probabilities of the states of $Y^{(\Delta)}$ in $[t_h, t_{h+1}]$:

$$\boldsymbol{\pi}^{(a)}(h\Delta) = \frac{1}{\Delta}\int_{h\Delta}^{(h+1)\Delta} \boldsymbol{\pi}(\tau)d\tau = \frac{1}{\Delta}\boldsymbol{\pi}(0)(e^{\mathbf{Q}\Delta}\mathbf{P})^h \int_0^{\Delta} e^{\mathbf{Q}\tau}d\tau \qquad (1)$$

Finally, we are in general interested by performance measures defined on the states *of the system*, and not on the states of the stochastic process $Y^{(\Delta)}$. Hence, all components of $\boldsymbol{\pi}^{(a)}(t)$ corresponding to a given state of the original system (i.e., when neglecting the residual numbers) are summed up to compute performance measures.

Steady-state analysis Since the distribution depends on the relative position w.r.t. the points $h\varDelta$, the approximate process does not admit a steady-state distribution but it is asymptotically periodic, with period $\varDelta$. We first compute the (approximate) steady-state distribution at times $h\varDelta$: $\boldsymbol{\pi}^{(\varDelta)} \overset{def}{=} \lim_{h\to\infty} \boldsymbol{\pi}(h\varDelta^+)$. This steady-state distribution is computed by a transient analysis stopped when the distribution is stabilised. Since $Y^{(\varDelta)}$ is asymptotically periodic with $\varDelta$ as period, we average the steady-state distributions on an interval $[0, \varDelta]$. Then the approximate steady-state distribution is given by:

$$\boldsymbol{\pi}^{(a)} = \frac{1}{\varDelta}\boldsymbol{\pi}^{\varDelta} \int_0^{\varDelta} e^{\mathbf{Q}\tau} d\tau \tag{2}$$

As in the transient case, all components of $\boldsymbol{\pi}^{(a)}$ corresponding to a given state of the system are summed up to compute performance indices.

2.2 Numerical considerations

Formulae (1) and (2) for transient and steady-state probabilities involve vector-matrix products with possibly very large matrices, either $e^{\mathbf{Q}\varDelta}$ or $I^{(\varDelta)} = \int_0^{\varDelta} e^{\mathbf{Q}\tau} d\tau$. Moreover, it is well-known (and reported as the “fill in” phenomenon) that, although $\mathbf{Q}$ is generally very sparse, $e^{\mathbf{Q}\tau}$ is not sparse at all. Since these matrices are only required through vector-matrix products, the usual approach [17] is to never compute these matrices explicitly but to compute directly the vector-matrix products avoiding the fill in phenomenon. The products of a vector by an exponential matrix are based on the series expansion of the exponential matrix (uniformization) and numerical summation until a required precision level is reached. This is the method that we have implemented.

When we need $e^{\mathbf{Q}\tau}$ we follow the uniformization approach [18]. If $\mathbf{A}_u = \mathbf{I} + \frac{1}{u}\mathbf{Q}$ is the uniformised matrix of $\mathbf{Q}$ with rate $u > \max_i\{|q_{ii}|\}$, we have

$$e^{\mathbf{Q}\tau} = \sum_{k\geq 0} e^{-u\tau} \frac{(u\tau)^k}{k!} (\mathbf{A}_u)^k \tag{3}$$

For the transient solution (1), $\boldsymbol{\pi}(0)\,(e^{\mathbf{Q}\varDelta}\mathbf{P})^h$ is computed iteratively. During the algorithm only one current vector $\mathbf{V}$ indexed on the state space is required (two for the intermediate computations) and for each step we apply the vector-matrix product method to $\mathbf{V} \cdot e^{\mathbf{Q}\varDelta}\mathbf{P}$. The computation of $I^{(\varDelta)} = \int_0^{\varDelta} e^{\mathbf{Q}\tau} d\tau$ is based on (3). By definition,
$I^{(\varDelta)} = \sum_{k\geq 0} \left[\int_0^{\varDelta} e^{-ut} \frac{(ut)^k}{k!} dt\right] (\mathbf{A}_u)^k$

An elementary derivation with integration by parts and summation gives:

$$I^{(\varDelta)} = \frac{1}{u} \sum_{k\geq 0} \left[1 - e^{-u\varDelta} \sum_{h=0}^{h=k} \frac{(u\varDelta)^h}{h!}\right] (\mathbf{A}_u)^k \tag{4}$$

As for $e^{\mathbf{Q}\varDelta}$, we only need $I^{(\varDelta)}$ through products $\frac{1}{\varDelta} \cdot V \cdot I^{(\varDelta)}$. We compute these products iteratively to avoid the fill in. An analogous approach was used in [19] for steady-state solution of Deterministic Stochastic Petri Nets (DSPN) but restricted to one deterministic event at any given time.

Steady-state solution is obtained in a similar way, steps $\mathbf{V}^{(m+1)} = \mathbf{V}^{(m)} e^{\mathbf{Q}\Delta}\mathbf{P}$ being computed until convergence.

Algorithm 2.1 : Computing the approximate probability distribution (time horizon τ)

```
// $\epsilon$ is the required precision
// $n_0$ is the initial value of $n$, the subdivision factor
// $inc$ is the additive term applied to $n$ at each step of the iteration
begin
    $n \leftarrow n_0$
    compute $\boldsymbol{\pi}_n^{(L)}(\tau)$ and $\boldsymbol{\pi}_n^{(H)}(\tau)$
    $\mathbf{V} \leftarrow (\boldsymbol{\pi}_n^{(L)}(\tau) + \boldsymbol{\pi}_n^{(H)}(\tau))/2$
    repeat
        $n \leftarrow n + inc$
        $old\mathbf{V} \leftarrow \mathbf{V}$
        $old\boldsymbol{\pi}_n^{(L)}(\tau) \leftarrow \boldsymbol{\pi}_n^{(L)}(\tau)$
        $old\boldsymbol{\pi}_n^{(H)}(\tau) \leftarrow \boldsymbol{\pi}_n^{(H)}(\tau)$
        compute $\boldsymbol{\pi}_n^{(L)}(\tau)$ and $\boldsymbol{\pi}_n^{(H)}(\tau)$
        $d_n^{(L)} \leftarrow 1/ \parallel \boldsymbol{\pi}_n^{(L)}(\tau) - old\boldsymbol{\pi}_n^{(L)}(\tau) \parallel$
        $d_n^{(H)} \leftarrow 1/\parallel \boldsymbol{\pi}_n^{(H)}(\tau) - old\boldsymbol{\pi}_n^{(H)}(\tau) \parallel$
        $\mathbf{V} \leftarrow \left(d_n^{(L)}\boldsymbol{\pi}_n^{(L)}(\tau) + d_n^{(H)}\boldsymbol{\pi}_n^{(H)}(\tau)\right) / \left(d_n^{(L)} + d_n^{(H)}\right)$
        $d \leftarrow \parallel \mathbf{V} - old\mathbf{V} \parallel$
    until $d \leq \epsilon$
    // $\mathbf{V}$ is the approximation
end
```

Choosing an approximate probability vector Recall that our goal is to give an approximate probability vector $\boldsymbol{\pi}^{(a)}$ either at time τ or in steady-state for models with finite support distributions. The parameter of the approximation is given by n leading to the interval length $\Delta = 1/n$.

The computation of the approximate $\boldsymbol{\pi}^{(a)}(\tau)$ is given in Algorithm 2.1 for the transient case with a given time horizon τ. The main idea is to compute successive approximation vectors until a given level ϵ of precision is reached. At each step we increase the precision of the approximation by decreasing the size Δ of the elementary interval. In the algorithm, we use the L_1 norm $\parallel \boldsymbol{\pi}_1 - \boldsymbol{\pi}_2 \parallel = \sum_i |\boldsymbol{\pi}_1[i] - \boldsymbol{\pi}_2[i]|$ to compare two probability distributions $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ and the precision of the approximation is given by the distance between successive vectors.

The special feature of the algorithm lies in the definition of our approximate vector ($\mathbf{V}$). Recall (see section 2.1), that for a given n (hence Δ) we can choose between two approximations depending whether we count or not the next $k\Delta$ to be met in the value returned by the discrete random variable corresponding the distribution of a non Markovian event. This gives us two approximate vectors at time $h\Delta$ denoted by $\boldsymbol{\pi}_n^{(L)}(\tau)$ and $\boldsymbol{\pi}_n^{(H)}(\tau)$. We observed during our experiments that the sequences $(\boldsymbol{\pi}_n^{(L)}(\tau))_{n\in\{n_0+k\cdot inc\}}$ and $(\boldsymbol{\pi}_n^{(H)}(\tau))_{n\in\{n_0+k\cdot inc\}}$ are both convergent but that $\parallel \boldsymbol{\pi}_n^{(L)}(\tau) - \boldsymbol{\pi}_n^{(H)}(\tau) \parallel_{n\in\{n_0+k\cdot inc\}}$ does not necessarily converge to 0. Moreover, several comparisons have shown that depending on the parameters, one of the two sequences converges faster than the other and that the corresponding limit is closer to the exact distribution (when available) than the other one. These behaviours have led us to define the approximate distribution for n as a weighed sum of $\boldsymbol{\pi}_n^{(L)}(\tau)$ and $\boldsymbol{\pi}_n^{(H)}(\tau)$ based on their respective convergence rate as given in the algorithm. Note that, as usual with efficient iterative methods, we are not able to estimate analytically the convergence rate.

The steady-state approximate distribution algorithm is defined similarly except that the successive approximations are computed with the method explained in the steady-state analysis paragraph.

Note that we compute iteratively the sums (1) and (2) so that we only store two probability vectors during computation and no (full) exponential matrix.

3 Application to stochastic Petri nets

3.1 Presentation of stochastic Petri nets

Syntax A stochastic Petri net is a Petri net enhanced by distributions associated with transitions. In the following definition, we distinguish two kinds of transitions depending on whether their distribution is *exponential* or *general.*

Definition 1. *A (marked) stochastic Petri net (SPN) $N = (P, T, Pre, Post, \mu, \Phi, w, m_0)$ is defined by:*

- *P, a finite set of places*
- *$T = T_X \uplus T_G$ with $P \cap T = \emptyset$, a finite set of transitions, disjoint union of exponential transitions T_X and general transitions T_G sets*
- *Pre (resp. $Post$), the backward (resp. forward) incidence matrix from $P \times T$ to $\mathbb{N}$*
- *μ, a function from T_X to $\mathbb{R}^{+*}$, the strictly positive rate of exponential transitions*
- *Φ, a function from T_G to the set of finite support distributions defining the distributions of the general transitions*
- *w, a function from T_G to $\mathbb{R}^{+*}$, the weight of the general transitions*
- *m_0, a integer vector of $\mathbb{N}^P$ the initial marking*

Notations

- $\mathbf{dmax}(t) = \lceil Inf(x \mid \Phi(t)(x) = 1) \rceil$ denotes the integer least upper bound of the support of $\Phi(t)$ (i.e., the range of a random variable with distribution $\Phi(t)$).
- Let m be a marking, $En(m) = \{t \mid m \geq Pre(t)\}$ denotes the set of enabled transitions. $En^X(m) = En(m) \cap T_X$ (resp. $En^G(m) = En(m) \cap T_G$) denotes the set of enabled exponential (resp. general) transitions.
- $T_G = \{t_1, \ldots, t_{n_g}\}$.

We assume that $\forall t \in T_G,\ \Phi(t)(0) = 0$, meaning that a general transition cannot be immediately fired. This also excludes the possibility of immediate transitions. This restriction is introduced only for readability purposes. In a forthcoming technical report, we will indicate how we handle this case which requires more complicated computation also encountered with discrete time SPN [20,21,11].

Semantics We briefly sketch the dynamic behaviour of the stochastic process associated with an SPN. In fact, we give two *equivalent* descriptions. The former is a standard one whereas the latter takes into account the properties of the exponential distribution.

• *First description* At some time τ, a *tangible state* of the stochastic process is given by a marking m and a vector $\mathbf{d}$ of (residual) non null delays over $En(m)$. The process lets time elapse until $\tau + dmin$ where $dmin = Inf(\mathbf{d}(t))$ decrementing the delays. Let $Fired$ be the subset of transitions such that the corresponding delay is now null. The process performs a probabilistic choice between these transitions whose distribution is defined according to their weights (for this semantics, we associate also weights with exponential transitions). Let t be the selected transition, an intermediate state is reached with marking $m' = m - Pre(t) + Post(t)$. If $(Fired \setminus \{t\}) \cap En(m') \neq \emptyset$, the process performs again a choice between these transitions and fires the selected transition. This iterative step ends when all the transitions in $Fired$ have been selected or disabled at least once. Given m' the reached marking, a new tangible state is now obtained by choosing a delay for every $t \in En(m')$ such that either t was not enabled in one of the previous markings or t was fired. This probabilistic choice is done according to $\Phi(t)$ or to the exponential distribution of parameter $\mu(t)$ (depending on the type of the transition). Note that delaying these choices after $Fired$ has been exhausted does not modify the semantics due to our assumption about $\Phi(t)(0)$.

• *Second description* At some time τ, a *tangible state* of the stochastic process is given by a marking m and a vector $\mathbf{d}$ of (residual) non null delays over $En^G(m)$. The process "computes" probabilities of some events related an *hypothetical* delay (say also $\mathbf{d}(t)$) for every $t \in En^X(m)$ chosen according to the exponential distribution of parameter $\mu(t)$ and an *hypothetical* induced $dmin = Inf(\mathbf{d}(t) \mid t \in En(m))$. These (mutually exclusive) events are: $dmin$ is associated with a *single* exponential transition t or $dmin$ is associated with a set of general transitions, $Fired$. The other cases have a null probability to occur. We note

$dmin^G = Inf(\mathbf{d}(t) \mid t \in En^G(m))$ Thus the process randomly selects one of these cases according to these probabilities and acts as follows:

- The hypothetical $dmin$ is associated with a *single* exponential transition t. The process selects a delay $dcur$ for firing t according to a conditional distribution obtained from the exponential one by requiring that $dcur < dmin^G$. Then the process lets time elapse until $\tau + dcur$ and the delays of general transitions are decremented. Afterwards t is fired and for every transition $t' \in En^G(m') \setminus En^G(m)$ a new delay is chosen according to $\Phi(t')$.
- $dmin = dmin^G$ is associated with a set of general transitions denoted $Fired$. The process lets time elapse until $\tau + dmin^G$ decrementing the delays. Then the process performs a probabilistic choice between the transitions of $Fired$ w.r.t. their weights. Let t be the selected transition, an intermediate state is reached with marking $m' = m - Pre(t) + Post(t)$. If $(Fired \setminus \{t\}) \cap En(m') \neq \emptyset$, the process performs again a choice between these transitions and fires the selected transition. This iterative step ends when all the transitions in $Fired$ have been selected or disabled at least once. Given m' the reached marking, a new tangible state is now obtained by choosing a delay for every $t \in En^G(m')$ such that either t was disabled in one of the previous markings or t was fired. This probabilistic choice is done according to $\Phi(t)$.

Discussion When defining semantics for SPNs one must fix three policies: the service policy, the choice policy and the memory policy. Here in both cases we have chosen the simplest policies. However most of the other choices do not yield significant additional difficulties w.r.t. the application of our generic method. Let us detail our policies. First, we have chosen the *single server* policy meaning that whatever is the enabling degree of an enabled transition t we consider a single instance of firing for t. Second, we have chosen the *race* policy meaning that the selection of the next transition to be fired is performed according to shortest residual delay: this is the standard assumption. In order to select transitions with equal delays, we perform a probabilistic choice defined by weights. Last, we have chosen the *enabling* memory meaning that the delay associated with a transition is kept until the firing or the disabling of the transition. Finally, the firing of a transition t is considered as *atomic* meaning that we do not look at the intermediate marking obtained after consuming the tokens specified by $Pre(t)$ in order to determine which transitions are still enabled.

3.2 An approximate stochastic process for SPNs

The approximate process we propose behaves as the generic process of section 2. It is parameterized by n and $\Delta = 1/n$ where the greater is n the better is the approximation.

First we compute for $t \in T_G$, $\Phi^n(t)$ an approximate discrete distribution of $\Phi(t)$.

Definition 2. *Let $\Phi(t)$ be a finite support distribution with integer l.u.b.* $\mathbf{dmax}(t)$ *such that $\Phi(t)(0) = 0$. Then the distribution $\Phi^n(t)$ is defined by the random variable X:*

- *whose range is defined by $range^n(t) = \{1, 2, \ldots, n{\cdot}\mathbf{dmax}(t){-}1, n{\cdot}\mathbf{dmax}(t)\}$,*
- *and $Prob(X = i) = \Phi(t)(i/n) - \Phi(t)((i-1)/n)$.*

The semantics of the approximate process is close to the second description of the semantics of the SPN. However, here the time is divided in intervals of length Δ and the firings of a general transition may only occur at some $h\Delta$. At a time $\tau \in [h\Delta, (h+1)\Delta)$, a tangible state of the approximate process is defined by a marking m and and a vector $\mathbf{d}$ of (residual) non null delays over $En^G(m)$. The residual delay $\mathbf{d}(t) \in \{1, 2, \ldots, n \cdot \mathbf{dmax}(t) - 1, n \cdot \mathbf{dmax}(t)\}$ and is now interpreted as the number of intervals to elapse before the firing of t.

The process "computes" probabilities of some events related to an *hypothetical* delay (say also $\mathbf{d}(t)$) for every $t \in En^X(m)$ chosen according to the exponential distribution of parameter $\mu(t)$ and an *hypothetical* induced $dmin = Inf(\mathbf{d}(t) \mid t \in En^X(m))$. These (mutually exclusive) events are: $dmin < (k+1)\Delta - \tau$ and $dmin$ is associated with a *single* exponential transition t or $dmin > (k+1)\Delta - \tau$. The other cases have a null probability to occur. Thus the process randomly selects one of these cases according to these probabilities and acts as follows:

- The hypothetical $dmin < (k+1)\Delta - \tau$ is associated with a *single* exponential transition t. The process selects a delay $dcur$ for firing t according to a conditional distribution obtained from the exponential one by requiring that $dcur < (k+1)\Delta - \tau$. Then the process lets time elapse until $\tau + dcur$ and the delays of general transitions are *unchanged* as the process lies in the same interval. Afterwards t is fired and for every transition $t' \in En^G(m') \setminus En^G(m)$ a new delay is chosen according to $\Phi^n(t')$.
- $dmin > (k+1)\Delta - \tau$. The process lets time elapse until $(k+1)\Delta$ decrementing by one unit the delays. Let $Fired$ be the subset of transitions of $En^G(m)$ with a null delay. Then the process performs an probabilistic choice between the transitions of $Fired$ defined by their weight. Let t be the selected transition, an intermediate state is reached with marking $m' = m - Pre(t) + Post(t)$. If $(Fired \setminus \{t\}) \cap En(m') \neq \emptyset$, the process performs again a choice between these transitions and fires the selected transition. This iterative step ends when all the transitions in $Fired$ have been selected or disabled at least once. Given m' the reached marking, a new tangible state is now obtained by choosing a delay for every $t \in En^G(m')$ such that either t was disabled in one of the previous markings or t was fired. This probabilistic choice is done according to $\Phi^n(t)$.

At this stage, it should be clear that this approximate process is a special case of the process we have described in section 2. During an interval $(h\Delta, (h+1)\Delta)$, only the Markovian transitions fire whereas at time points $h\Delta$, only the general transitions fire. It remains to define the state space of this stochastic process and the associated matrices $\mathbf{P}$ and $\mathbf{Q}$.

4 From a SPN to the tensorial expressions of its approximate process

4.1 The state space associated with the matrix Q

As usual with the tensorial based methods, we build an over-approximation of the state space. However here, this over-approximation is reduced since we represent the state space as a finite union of cartesian products of sets instead of a single cartesian product.

First, one builds the reachability graph of the untimed version of the SPN. It is well-known that all the reachable markings of the SPN are also reachable in this setting. Let us denote the set of reachable markings M, the second step consists in partitioning M w.r.t. the equivalence relation $m \equiv m'$ iff $En^G(m) = En^G(m')$. Thus $M = M(TE_1) \uplus \ldots \uplus M(TE_e)$ with $TE_i \subseteq T_G$ and $M(TE_i) = \{m \in M \mid En^G(m) = TE_i\}$.

According to section 3.2, MS the set of tangible states may be decomposed as follows:

$$MS = \biguplus_{i=1..e} M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i) \right)$$

where $range^n(t, TE_i)$ is:

- $\{0\}$ if $t \notin TE_i$,
- $range^n(t)$ otherwise

In this expression, we have associated an artificial delay (0) with each disabled general transition. This does not change the size of the state space and makes easier the design of the tensorial expressions.

4.2 Tensorial expression of Q

Let us recall that the matrix $\mathbf{Q}$ expresses the behaviour of the SPN inside an interval $(h\Delta, (h+1)\Delta)$. Thus a state change is only due to the firing of an exponential transition. As usual with tensorial methods for continuous time Markov chains, we represent $\mathbf{Q} = \mathbf{R} - \mathbf{Diag}(\mathbf{R} \cdot \mathbf{1})$. The matrix $\mathbf{R}$ includes only the rates of state changes whereas the matrix $\mathbf{Diag}(\mathbf{R} \cdot \mathbf{1})$ accumulates the rates of each row and s it on the diagonal coefficient. Generally the latter matrix is computed by a matrix-vector multiplication with the tensorial expression of $\mathbf{R}$ and then stored as a (diagonal) vector. Thus we focus on the tensorial expression of $\mathbf{R}$. More precisely, due to the representation of the state space, $\mathbf{R}$ is a block matrix where each block has a tensorial expression. We denote $\mathbf{R}_{i,j}$ the block corresponding to the state changes from $M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i)\right)$ to $M(TE_j) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_j)\right)$

We first give its expression and then we detail each component.

$$\mathbf{R}_{i,j} = \left[\sum_{t \in T_X} \mu(t) \mathbf{R}'_{i,j}(t) \right] \bigotimes \left(\bigotimes_{g=1}^{n_g} \mathbf{R'}_{i,j}(t_g) \right)$$

For $t \in T_X$, the matrix $\mathbf{R}'_{i,j}(t)$ is a binary matrix (i.e., with 0's and 1's) only depending on the reachability relation between markings: the presence of a 1 for row m and column m' witnesses the fact that $m \xrightarrow{t} m'$.

For $t_g \in T_G$, the matrix $\mathbf{R}'_{i,j}(t_g)$ expresses what happens to the delay of t_g when an exponential transition is fired. An important observation is that this matrix does not depend on the fired transition. Indeed it only depends on the enabling of t_g in $M(TE_i)$ and $M(TE_j)$, i.e., whether $t_g \in TE_i$ and $t_g \in TE_j$. If t_g belongs to both the subsets, the delay is unchanged (yielding the identity matrix $\mathbf{I}$). If t_g does not belong to TE_j, the delay is reset to 0. Finally, if t_g belongs to TE_j and does not belong to TE_i the delay is randomly selected w.r.t. the distribution $\Phi^n(t_g)$ (see table 1).

	$t_g \in TE_j$	$t_g \notin TE_j$
$t_g \in TE_i$	$\mathbf{I}$	$\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$
$t_g \notin TE_i$	$\Phi^n(t_g)$	$[1]$

Table 1: Structure of matrices $\mathbf{R}'_{i,j}(t_g)$

Due to the previous observation, rather than computing the matrices the matrix $\mathbf{R}'_{i,j}(t)$ for every $t \in T_X$, it is more efficient to compute their weighted sum:

$$\mathbf{RX}_{i,j} = \sum_{t \in T_X} \mu(t)\mathbf{R}'_{i,j}(t)$$

Furthermore, this computation can be performed on the fly when building the reachability graph of the net.

4.3 The state space associated with the matrix P

The matrix $\mathbf{P}$ expresses the instantaneous changes at instants $h\Delta$. Such a change consists in decrementing the delays of the enabled general transitions followed by successive firings of general transitions with null delays (see section 3.2).

First, observe that the delay of a general enabled transition t during the intermediate stages may be either 0 (when ready to fire) or any value in $range^n(t)$ (when newly enabled). Thus the state space needs to be expanded to include null delays for enabled transitions. Furthermore, the matrix decomposition into blocks needs to be refined in order to obtain a tensorial expression. Indeed the decomposition associated with $\mathbf{Q}$ was based on the set of the enabled general transitions. Here a block will correspond to a pair (TE_i, TF) where $TF \subseteq TE_i$ represents the enabled transitions with null delay. We call such a transition a *fireable* transition.

Thus MS' the set of vanishing states may be decomposed as follows:

$$MS' = \biguplus_{i=1..e} \biguplus_{TF \subseteq TE_i} M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i, TF) \right)$$

where $range^n(t, TE_i, TF)$ is:

– $\{0\}$ if $t \notin TE_i$ or $t \in TF$,
– $range^n(t)$ otherwise

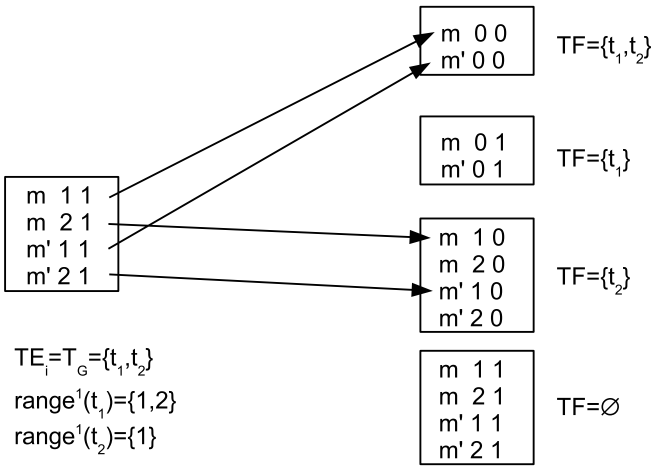

Fig. 2: Expanding the state vector while decrementing the delay

During the computations, we will expand the current probability vector then we will multiply it by $\mathbf{P}$ and afterwards contract it. Figure 2 illustrates the expansion of a block $M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i)\right)$ into blocks $\biguplus_{TF \subseteq TE_i} M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i, TF)\right)$. We have represented inside the boxes the indices of the vectors (and not their contents). In fact, we perform simultaneously the expansion and the decrementation of the delay. The copy of contents are represented by the arrows. The contraction is not represented as it is straightforward to design. It consists in copying the contents of the block associated with $TF = \emptyset$ to the original block.

4.4 Tensorial expression of P

In order to describe the matrix $\mathbf{P}$ we adopt a top-down approach. First, $\mathbf{P} = \mathbf{DE} \cdot (\mathbf{P}^{one})^{n_g} \cdot \mathbf{C}$ where $\mathbf{DE}$ corresponds to the delay decrementation phase with expansion of the probability vector and $\mathbf{C}$ corresponds the contraction of this vector. We do not detail these matrices since, in the implementation, they are not stored. Rather the effect of their multiplication is coded by two algorithms whose time complexities are linear w.r.t. the size of the probability vector. The matrix $\mathbf{P}^{one}$ represents the change due to the firing of one fireable general transition in a state or no change if there is no such transition. Depending on the state reached at the instant $h\Delta$ and of the choice of the transitions, a sequence with variable length of general transitions may be fired. However due to our assumption about the distributions, this length is bounded by n_g. Thus our expression is sound.

Let us detail $\mathbf{P}^{one}$. We denote $\mathbf{P}^{one}(i, TF, j, TF')$ the block corresponding to the state changes from $M(TE_i) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_i, TF)\right)$ to $M(TE_j) \times \left(\prod_{g=1}^{n_g} range^n(t_g, TE_j, TF')\right)$.

If $TF = \emptyset$ then $\mathbf{P}^{one}(i, \emptyset, i, \emptyset) = \mathbf{I}$ and for every $(j, TF') \neq (i, \emptyset)$ $\mathbf{P}^{one}(i, \emptyset, j, TF') = \mathbf{0}$.

Assume now that $TF \neq \emptyset$ then $\mathbf{P}^{one}$ corresponds to a probabilistic choice of a transition in TF and its firing. Thus we write:

$$\mathbf{P}^{one}(i, TF, j, TF') = \sum_{t \in TF} \frac{w(t)}{\sum_{t' \in TF} w(t')} \mathbf{P}^{one}_t(i, TF, j, TF')$$

Similarly to the case of $\mathbf{Q}$, $\mathbf{P}^{one}_t(i, TF, j, TF')$ can be expressed as a tensorial product:

$$\mathbf{P}^{one}_t(i, TF, j, TF') = \mathbf{PM}(t, i, j) \bigotimes \left(\bigotimes_{g=1}^{n_g} \mathbf{PG}(t_g, t, i, TF, j, TF') \right).$$

The binary matrix $\mathbf{PM}(t, i, j)$ only depends on the reachability relation between markings: the presence of a 1 for row m and column m' witnesses the fact that $m \xrightarrow{t} m'$. These matrices can be computed on the fly when building the reachability graph of the untimed net.

For $t_g \in T_G$, the matrix $\mathbf{PG}(t_g, t, i, TF, j, TF')$ expresses what happens to the delay of t_g when t is fired. Its structure is described in table 2. The only possibility for t_g to be fireable is that it was still fireable and not fired (second row of first column). If t_g is enabled but not fireable, then it was still in this situation with the same delay or was disabled and then the delay is chosen according to its distribution (see the second column). If, after the firing of t, t_g is disabled then its delay is reset (see the third column of the table).

5 Implementation details

Our implementation is coded in the Python language [22] completed with the Numerical package [23] for better linear algebra computations and the sparse package [24] for efficient handling of sparse matrices. All our computations were done on a Pentium-PC 2.6Ghz, 512MB.

Let us explain some important features of this implementation. The SPN is described by means of a Python function called during the initial phase of the

	$t_g \in TF'$	$t_g \in TE_j \setminus TF'$	$t_g \in T_G \setminus TE_j$
$t_g = t \vee t_g \in T_G \setminus TE_i$	$[0]$	$\Phi^n(t_g)$	$[1]$
$t_g \in TF \setminus \{t\}$	$[1]$	$[0, 0, \ldots, 0]$	$[1]$
$t_g \in TE_i \setminus TF$	$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$	$\mathbf{I}$	$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

Table 2: Structure of matrices $\mathbf{PG}(t_g, t, i, TF, j, TF')$

computation. During this step all data structures which are independent of the "precision" parameter n (or Δ) are built. This includes the reachability graph, the sets TE_i, TF for a given TE_i and the state spaces $M(TE_i)$. We also store the matrices $\mathbf{RX}_{i,j}$ and the $\mathbf{PM}(t, i, j)$ matrices in a sparse format. The $\mathbf{R}'_{i,j}(t_g)$ matrices are encoded in a symbolic way, that is to say we store what kind of matrix we are dealing with, among the ones given in Table 1. Each of these matrices will be "expanded" for a given n when needed during the probability vector computation.

The main part of the algorithm is to iteratively compute the approximate (either steady-state or transient) probability vector for a given precision n. Accordingly with the introduction of the tensorial description of the matrices, we never compute global matrices but in contrast vector-matrix products $\mathbf{V} \bigotimes_{k=1}^{K} \mathbf{X}_k$ where $\mathbf{V}$ and $\mathbf{X}_k$ have compatible dimensions. Theses products are implemented following the so-called Shuffle algorithm adapted to non square matrices (see the synthesis [25]) and taking into account the symbolic representation of the $\mathbf{R}'_{i,j}(t_g)$ matrices. Such products are involved during the CTMC computation and during computation at times $h\Delta$. Note that, in the present work, we do not exploit anymore the efficient implementation of the product $\mathbf{VX}$ (where $\mathbf{X}$ is a sparse matrix) provided by the Pysparse package, since we always deal with a tensorial expression of the matrix $\mathbf{X}$.

A tricky point is the expansion phase of the state space during the computation at times $h\Delta$. We chose to store our data structure as multilevel lists so that the expansion is mainly a matter of careful traversals of the list structures for which Python is well-suited.

6 Related works

Since we deal with systems composed of general and exponential distributions, it is impossible, except for special cases, to derive analytical expressions of the transient or even steady-state distributions of the states. Thus most results are developed on so-called state based models and they involve numerical solution algorithms.

When the system exhibits complex synchronization, the Queueing Network framework becomes frequently too restrictive and in fact, many works have studied non exponential activities with the help of the non Markovian Stochastic Petri Nets (NMSPN) formalism, some of them being adapted to general distributions and other ones to deterministic distributions only. In this context, there are two main categories of works.

The first family of solutions defines conditions under which i) the underlying stochastic process is a MRGP and ii) the parameters of this MRGP can be derived from the NMSPN definition. In [8], the author introduces "Cascaded" Deterministic SPN (C-DSPN). A C-DSPN is a DSPN for which *when two or more deterministic transitions (activities) are concurrently enabled they are enabled in the same states.* With the additional constraint that the $(k+1)$th firing time is a multiple of the kth one, it is possible to compute efficiently the probability distribution as we do. In [9], the authors derive the elements of the MRGP underlying a SPN with general finite support distributions. However, the NMSPN must satisfy the condition that *several generally distributed transitions concurrently enabled must become enabled at the same time* (being able to become disabled at various times). The transient analysis is achieved first in the Laplace transform domain and then by a numerical Laplace inverse transformation. A simpler method is used for the steady-state solution.

The second family of solutions is based on phase-type distributions, either continuous (CPHD) or discrete (DPHD). In [10], the authors compare the qualities of fitting general distributions with DPHD or CPHD. It is shown that the time step (the scale factor) of DPHD plays a essential role in the quality of the fitting. [11] introduces the Phased Delay SPNs (PDSPN) which mix CPHD and DCPHD (general distributions must have been fitted to such distributions by the modeller). As pointed out by the authors, without any restriction, the transient or steady-state solutions of PDSPN can only be computed by stochastic simulation. However *when synchronization is imposed between firings of a CPHD transition and resamplings of DPHD transitions* the underlying stochastic process is a MRGP and its parameters can be derived from the reachability graph of the PDSPN. The approach of [12] is based on full discretization of the process. The distributions of the transitions are either DPHD or exponential (general distributions must be fitted with DPHD). For an appropriate time step, all exponential distributions are then discretized as DPHD and the solution is computed through the resulting process which is a DTMC. We note that discretization may introduce simultaneous event occurrences corresponding to achievement of continuous Markovian activities, an eventuality with zero probability in the continuous setting.

In contrast to our approach, the other approaches derive the stochastic process underlying the SPN which is then solved, possibly with approximate methods. However, restrictions on the concurrency between generally distributed activities are always imposed in order to design efficient methods for transient or steady-state solutions.

7 Conclusions and perspectives

Main results We have presented a new approximate method for stochastic Petri nets including non Markovian concurrent activities associated with transitions. Contrary to the other methods, we have given an approximate semantics to the net and applied an exact analysis rather than the opposite. The key factor for the quality of this approximation is that the occurrences of Markovian transitions are not approximated as it would be in a naive discretisation process. Furthermore, the design of its analysis is based on robust numerical methods (i.e., uniformization) and the steady-state and transient cases are handled similarly. Finally due to the Petri nets formalism we have been able to exploit tensorial methods which have led to significant space complexity savings.

Future work: applications We informally illustrate here the usefulness of our method for some classes of "periodic" systems. Let us suppose that we want to analyse a database associated to a library. At any time, interactive research transactions may be activated by local or remote clients. In addition, every day at midnight, a batch transaction is performed corresponding to the update of the database by downloading remote information from a central database. In case of an overloaded database, the previous update may be still active. Thus the new update is not launched. Even if the modeller considers that all the transactions durations are defined by memoryless distributions, this non Markovian model does not admit a stationary distribution. However applying the current tools for non Markovian models will give the modeller a useless steady-state distribution with no clear interpretation. Instead we can model such a system in an exact way by considering that our approximate process of a SPN is in this case its real semantics. Then with our method, the modeller may analyse the asymptotic load of its system at different moments of the day in order to manage the additional load due to the batch transaction.

Another application area of our method is the real-time systems domain. Such systems are often composed by periodic tasks and sporadic tasks both with deadlines. With our hybrid model, we can efficiently compute the steady-state probability of deadline missing tasks.

Future work: SPNs with multiple time-scales It is well known that stochastic systems with events having very different time scales often lead to difficulties during numerical transient or steady-state analysis. This is even worse when these events are non Markovian. These difficulties mainly arise because we need to study the stochastic process during the largest time scale but with a precision which is driven by the smallest time scale. Thus the resulting state space is generally huge. In [26], we have extended our generic method in order to efficiently deal with very different time scales. We plan to adapt this work in the framework of SPNs.

References

1. Cox, D.R.: A use of complex probabilities in the theory of stochastic processes. Proc. Cambridge Philosophical Society (1955) 313–319
2. German, R., Logothesis, D., Trivedi, K.: Transient analysis of Markov regenerative stochastic Petri nets: A comparison of approaches. In: Proc. of the 6th International Workshop on Petri Nets and Performance Models, Durham, NC, USA, IEEE Computer Society Press (1995) 103–112
3. Cox, D.R.: The analysis of non-Markov stochastic processes by the inclusion of supplementary variables. Proc. Cambridge Philosophical Society (Math. and Phys. Sciences) **51** (1955) 433–441
4. German, R., Lindemann, C.: Analysis of stochastic Petri nets by the method of supplementary variables. Performance Evaluation **20**(1–3) (1994) 317–335 special issue: Peformance'93.
5. Ajmone Marsan, M., Chiola, G.: On Petri nets with deterministic and exponentially distributed firing times. In Rozenberg, G., ed.: Advances in Petri Nets 1987. Number 266 in LNCS. Springer–Verlag (1987) 132–145
6. Lindemann, C., Schedler, G.: Numerical analysis of deterministic and stochastic Petri nets with concurrent deterministic transitions. Performance Evaluation **27–28** (1996) 576–582 special issue: Proc. of PERFORMANCE'96.
7. Lindemann, C., Reuys, A., Thümmler, A.: DSPNexpress 2.000 performance and dependability modeling environment. In: Proc. of the 29th Int. Symp. on Fault Tolerant Computing, Madison, Wisconsin (1999)
8. German, R.: Cascaded deterministic and stochastic petri nets. In B. Plateau, W.J.S., Silva, M., eds.: Proc. of the third Int. Workshop on Numerical Solution of Markov Chains, Zaragoza, Spain, Prensas Universitarias de Zaragoza (1999) 111–130
9. Puliafito, A., Scarpa, M., Trivedi, K.: K-simultaneously enable generally distributed timed transitions. Performance Evaluation **32**(1) (1998) 1–34
10. Bobbio, A., Telek, A.H.M.: The scale factor: A new degree of freedom in phase type approximation. In: International Conference on Dependable Systems and Networks (DSN 2002) - IPDS 2002, Washington, DC, USA, IEEE C.S. Press (2002) 627–636
11. Jones, R.L., Ciardo, G.: On phased delay stochastic petri nets: Definition and an application. In: Proc. of the 9th Int. Workshop on Petri nets and performance models (PNPM01), Aachen, Germany, IEEE Comp. Soc. Press. (2001) 165–174
12. Horváth, A., Puliafito, A., Scarpa, M., Telek, M.: A discrete time approach to the analysis of non-markovian stochastic Petri nets. In: Proc. of the 11th Int. Conf. on Computer Performance Evaluation. Modelling Techniques and Tools (TOOLS'00). Number 1786 in LNCS, Schaumburg, IL, USA, Springer–Verlag (2000) 171–187
13. Haddad, S., Mokdad, L., Moreaux, P.: Performance evaluation of non Markovian stochastic discrete event systems - a new approach. In: Proc. of the 7th IFAC Workshop on Discrete Event Systems (WODES'04), Reims, France, IFAC (2004)
14. Gross, D., Miller, D.: The randomization technique as a modeling tool an solution procedure for transient markov processes. Operations Research **32**(2) (1984) 343–361
15. Lindemann, C.: DSPNexpress: A software package for the efficient solution of deterministic and stochastic Petri nets. In: Proc. of the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Edinburgh, Scotland, UK, Edinburgh University Press (1992) 9–20

16. Donatelli, S., Haddad, S., Moreaux, P.: Structured characterization of the Markov chains of phase-type SPN. In: Proc. of the 10th International Conference on Computer Performance Evaluation. Modelling Techniques and Tools (TOOLS'98). Number 1469 in LNCS, Palma de Mallorca, Spain, Springer–Verlag (1998) 243–254
17. Sidje, R., Stewart, W.: A survey of methods for computing large sparse matrix exponentials arising in Markov chains. Computational Statistics and Data Analysis **29** (1999) 345–368
18. Stewart, W.J.: Introduction to the numerical solution of Markov chains. Princeton University Press, USA (1994)
19. German, R.: Iterative analysis of Markov regenerative models. Performance Evaluation **44** (2001) 51–72
20. Ciardo, G., Zijal, R.: Well defined stochastic Petri nets. In: Proc. of the 4th Int. Workshop on Modeling, Ananlysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96), San Jose, CA, USA, IEEE Comp. Soc. Press (1996) 278–284
21. Scarpa, M., Bobbio, A.: Kronecker representation of stochastic Petri nets with discrete PH distributions. In: International Computer Performance and Dependability Symposium - IPDS98, Duke University, Durham, NC, IEEE Computer Society Press (1998)
22. Python team: Python home page: `http://www.python.org` (2004)
23. Dubois, P.: Numeric Python home page: `http://www.pfdubois.com/numpy/` (2004) and the Numpy community.
24. Geus, R.: PySparse home page: `http://www.geus.ch` (2004)
25. Buchholz, P., Ciardo, G., Kemper, P., Donatelli, S.: Complexity of memory-efficient kronecker operations with applications to the solution of markov models. INFORMS Journal on Computing **13**(3) (2000) 203–222
26. Haddad, S., Moreaux, P.: Approximate analysis of non-markovian stochastic systems with multiple time scale delays. In: Proc. of the 12th Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2004), Volendam, The Netherlands (2004) 23–30