

# Analysis of Deterministic and Stochastic Petri Nets

Gianfranco Ciardo\*

Department of Computer Science  
College of William and Mary  
Williamsburg, VA 23187-8795, USA  
ciardo@cs.wm.edu

Christoph Lindemann†

Institut für Technische Informatik  
Technische Universität Berlin  
10587 Berlin, Germany  
lind@cs.tu-berlin.de

## Abstract

We present a time and space efficient algorithm for computing steady state solutions of deterministic and stochastic Petri nets (DSPNs) with both stochastic and structural extensions. The algorithm can deal with different execution policies associated with deterministic transitions of a DSPN. The definition of a subordinated Markov chain (SMC) is refined to reduce the computational cost of deriving the transition probabilities of the embedded Markov chain (EMC) underlying a DSPN. Closed-form expressions of these transition probabilities are presented for some SMC topologies. Moreover, we propose to make use of the reward structure defined on the DSPN to reduce memory requirements. The usefulness of the proposed extensions and the steps of the solution algorithm are illustrated using a DSPN of a simple communication protocol.

## 1 Introduction

Petri Nets with deterministic transition firing times are an important tool for modeling discrete-event dynamic systems. Examples of activities which might have a constant duration are transfer times of fixed-size data packets in distributed computing systems, timeouts in real-time systems, and repair times of components in fault-tolerant systems. In timed Petri nets (TPNs) [14] all transitions have associated either constant delays (D-timed Petri Nets) or exponentially distributed delays (M-timed Petri Nets). In generalized timed Petri nets (GTPNs) [10], immediate tran-

sitions fire without a delay and timed transitions fire after a constant delay. TPNs and GTPNs employ discrete time scale for the underlying stochastic process. Their transitions fire in three phases and the next transition to fire is preselected according to a probability distribution.

Deterministic and stochastic Petri nets (DSPNs) have been introduced in [2] as a continuous-time modeling tool which includes both exponentially distributed and constant timing. In DSPNs, transition firing is atomic and the transition with the smallest firing delay is the next transition to fire. The embedded Markov chain (EMC) underlying a DSPN has fewer states than an analogous TPN or GTPN because transition firing is atomic in DSPNs. Moreover, the markings of a DSPN are defined as in an untimed Petri net, thus, standard structural analysis techniques can be employed for DSPNs. In particular, minimal-support place and transition invariants can be calculated.

Under the structural restriction that at most one deterministic transition is enabled in any marking, an analytical method to solve the DSPNs in steady state exists, based on the EMC. This solution method requires the derivation of the transition probability matrix of the EMC and associated holding time vectors. These quantities have to be computed by transient analysis of some continuous-time Markov chains (CTMCs) obtained by considering the possible firings of the exponential transitions enabled competitively or concurrently with a deterministic transition. Such CTMCs have been referred to as subordinated Markov chains (SMCs) and efficient computational formulas for their transient analysis have been presented in [12]. Recently, the software package DSPNexpress [11], implementing a distributed numerical solution algorithm for DSPNs, has become available.

We propose a simplified definition of execution policies for the firing process of deterministic transitions of

---

\*G. Ciardo was a Visiting Professor at the Technische Universität Berlin with support from the German Academic Exchange Office (DAAD).

†C. Lindemann was supported by the Federal Ministry for Research and Development of Germany (BMFT) under grant ITR 9003.

a DSPN and the association of different execution policies to deterministic transitions in a DSPN. We propose also several extensions to the original definition of DSPNs, allowing marking-dependence for weights of immediate transitions, firing delays of both exponential and deterministic transitions, arc multiplicities, and execution policies of deterministic transitions. Marking-dependent weights and rates for exponential transitions were already included in the original definition of DSPNs [2], while marking-dependent deterministic firing times were introduced in [13]. Marking-dependent execution policies have instead, to the best of our knowledge, never been proposed before. Furthermore, the example in Section 2.3 illustrates the usefulness of marking-dependent arc cardinalities.

We present an algorithm for computing steady-state solutions of DSPNs with these extensions. The algorithm formalizes and improves the previously known approach, examining some subtleties which had not been addressed. In particular, the algorithm employs a SMC for each marking in which a deterministic transition can begin its firing time, rather than one for each deterministic transition. This increases parallelism in the calculation of the transition probabilities of the EMC underlying a DSPN. Furthermore, the algorithm employs closed-form expressions of these transition probabilities in case a SMC contains only state transitions corresponding to firings of exponential transitions competitively enabled with a deterministic transition. These closed-form expressions considerably reduce the computational effort required for the transient analysis of such SMCs. Finally, we show how to reduce the memory requirements of the DSPN solution algorithm by using the reward structure defined on the DSPN. The presented algorithm can compute both the steady state probabilities for tangible markings of structurally bounded and live DSPNs, as already considered in [2, 12], and the cumulative sojourn times in tangible markings up to steady state for DSPNs with absorbing markings. The usefulness of the proposed extensions to DSPNs and the steps of the solution algorithm are illustrated using a DSPN modeling a simple communication protocol.

The paper is organized as follows. Section 2 introduces a formal definition of DSPNs, describes their stochastic behavior, and presents the running example. A detailed solution algorithm for DSPNs with the proposed extensions is introduced in Section 3. Section 4 contains a comparison of our algorithm with previous methods and gives concluding remarks.

## 2 Deterministic and stochastic Petri nets

### 2.1 Formal definition

A DSPN is a tuple  $\{P, T, I, O, H, g, M_0, \tau, w, e\}$  where:

- $P$  is a finite set of places, which can contain tokens. A marking  $i \in \mathbb{N}^{|P|}$  defines the number of tokens in each place  $p \in P$ , indicated by  $\#(p, i)$ , or simply  $\#(p)$  when the marking is understood.
- $T$  is a finite set of transitions, partitioned into three disjoint sets,  $T^Z$ ,  $T^E$ , and  $T^D$ , of immediate, exponential, and deterministic transitions, respectively.  $P \cap T = \emptyset$ .
- $\forall p \in P, \forall t \in T, I_{p,t} : \mathbb{N}^{|P|} \rightarrow \mathbb{N}$ ,  $O_{p,t} : \mathbb{N}^{|P|} \rightarrow \mathbb{N}$ , and  $H_{p,t} : \mathbb{N}^{|P|} \rightarrow \mathbb{N}$  are the multiplicities of the input arc from  $p$  to  $t$ , the output arc from  $t$  to  $p$ , and the inhibitor arc from  $p$  to  $t$ , respectively. Marking-dependent arc multiplicities might simplify the modeling of complex system behavior.
- $\forall t \in T, g_t : \mathbb{N}^{|P|} \rightarrow \{\text{True}, \text{False}\}$  is the guard for transition  $t$ .
- $M_0 \in \mathbb{N}^{|P|}$  is the initial marking.
- $\forall t \in T^E \cup T^D, \tau_t : \mathbb{N}^{|P|} \rightarrow (0, +\infty)$  is the mean firing time for timed transition  $t$ , it may be marking-dependent.
- $\forall t \in T^Z, w_t : \mathbb{N}^{|P|} \rightarrow (0, +\infty)$  is the firing weight for immediate transition  $t$ , it may be marking-dependent.
- $\forall t_1 \in T^Z \cup T^E, t_2 \in T^D, e_{t_1, t_2} : \mathbb{N}^{|P|} \rightarrow \{R, C\}$  is the execution policy to be used for transition  $t_2$  when transition  $t_1$  fires, it may be marking-dependent.  $R$  and  $C$  stand for “restart” and “continue”, respectively.

A place  $p$  is drawn as a circle, with the number of tokens in it written inside (default is zero). Transitions in  $T^Z$ ,  $T^E$ , and  $T^D$  are drawn as thin bars, empty rectangles, and filled rectangles, respectively. Input and output arcs have an arrowhead on their destination, inhibitor arcs have a small circle. The multiplicity is written on the arc (default is one); a missing arc indicates that the multiplicity is zero. The default value for guards is *True*.

A transition  $t \in T$  is enabled in marking  $i$  iff  $g_t(i) = \text{True}$  and  $\forall p \in P, (I_{p,t}(i) \leq \#(p, i)) \wedge (H_{p,t}(i) > \#(p, i) \vee H_{p,t}(i) = 0)$ .

When transition  $t$  fires in marking  $i$ , the new marking  $j$  satisfies:

$$\forall p \in P, \#(p, j) = \#(p, i) - I_{p,t}(i) + O_{p,t}(i)$$

Define  $\mathbf{E}(i)$  to be the set of transitions enabled in marking  $i$  and  $\mathbf{f}(s, i)$  to be the marking reached by firing  $s \in T^*$  in  $i$ , where  $T^*$  is the set of all sequences of transitions, including the empty sequence.

## 2.2 Stochastic behavior

Recently, Choi, Kulkarni, and Trivedi observed that the marking process,  $\{\mu(\theta), \theta \geq 0\}$ , underlying a DSPNs without concurrently enabled deterministic transitions constitutes a Markov regenerative stochastic process [4, 5], also known as semi-regenerative process [3], rather than a semi-Markov process as mentioned in [2]. In the companion paper [7] we show that the stochastic process underlying a DSPN with concurrently enabled deterministic transitions can still be a semi regenerative stochastic process in some cases. This paper concentrates on DSPNs without concurrently enabled deterministic transitions and their continuous-time marking process.

The solution method described in Section 3 requires that at most one deterministic transition is enabled in each marking. Using the definitions introduced above, the reachability graph  $(\mathcal{S}, \mathcal{A})$  of a DSPN is given by:

$$\begin{aligned} \mathcal{S} &= \{i \in \mathbb{N}^{|P|} : \exists s \in T^*, i = \mathbf{f}(s, M_0)\} \\ \mathcal{A} &= \{(i, j, t) \in \mathcal{S}^2 \times T : j = \mathbf{f}(t, i)\} \end{aligned}$$

The reachability set  $\mathcal{S}$  can be partitioned into  $\mathcal{S}^E = \{i \in \mathcal{S} : \forall t \in \mathbf{E}(i), t \in T^E\}$ ,  $\mathcal{S}^Z = \{i \in \mathcal{S} : \exists t \in \mathbf{E}(i), t \in T^Z\}$ , and  $\mathcal{S}^D = \{i \in \mathcal{S} : \exists t \in \mathbf{E}(i), t \in T^D\}$ . Denote  $d(i)$  to be the enabled deterministic transition in marking  $i \in \mathcal{S}^D$ . Markings in  $\mathcal{S}^E \cup \mathcal{S}^D = \mathcal{S}^{ED}$  are said to be tangible, while markings in  $\mathcal{S}^Z$  are said to be vanishing.

The firing time  $F_t(i)$  for transition  $t$  in marking  $i$  in isolation is a random variable with distribution  $Const(0)$  if  $t \in T^Z$ ,  $Expo(\tau_t(i)^{-1})$  if  $t \in T^E$ , and  $Const(\tau_t(i))$  if  $t \in T^D$ . If several immediate transitions are enabled, the function  $w$  defines the probability of firing enabled transition  $t_1$  in vanishing marking  $i$  as

$$\hat{w}_{t_1}(i) = \frac{w_{t_1}(i)}{\sum_{t \in \mathbf{E}(i)} w_t(i)}$$

As stated in [1], the description of the stochastic behavior must include, in addition to the distribution of the firing times, the “execution policy”. We adopt the

race policy: each transition  $t \in T$  has an associated remaining firing time (RFT)  $y_t$ . If marking  $i$  is entered at time  $\theta$ , the transition  $t_1$  with the minimum RFT among  $\mathbf{E}(i)$  is fired at time  $\theta + y_{t_1}$  and the RFTs for the other transitions in  $\mathbf{E}(i)$  are adjusted. A new RFT  $y_{t_1}$  for  $t_1$  is sampled from the distribution of  $F_{t_1}(l)$ , where  $l$  is the first marking where  $t_1$  becomes enabled again. Assume that  $j = \mathbf{f}(t_1, i)$ . Three policies for the adjustments to the RFTs of any other transition  $t_2$  enabled in  $i$  are [1]:

- Resampling:  $y_{t_2}$  is resampled from the distribution of  $F_{t_2}(l)$  in the first marking  $l$  where  $t_2$  becomes enabled again, possibly  $j$ .
- Age Memory:  $y_{t_2} \leftarrow (y_{t_2} - y_{t_1})\tau_{t_2}(j)/\tau_{t_2}(i)$ , where  $\tau_{t_2}(j)/\tau_{t_2}(i)$  is the “scaling factor” [1].
- Enabling Memory: use Age Memory if  $t_2$  is still enabled in  $j$ , otherwise use Resampling.

If  $t_2 \in T^Z \cup T^E$ , the choice between the three policies is irrelevant, since the RFT and the firing time have the the same distribution,  $Const(0)$  in the former case,  $Expo(\tau_{t_2}(j)^{-1})$  in the latter case. If  $t_2 \in T^D$ , instead:

- If  $t_2 \in \mathbf{E}(i)$  and  $t_2 \in \mathbf{E}(j)$ , the Enabling Memory and Age Memory policies are equivalent: we indicate them with  $e_{t_1, t_2}(i) = C$  (for continue) and Resampling with  $e_{t_1, t_2}(i) = R$  (for restart).
- If  $t_2 \in \mathbf{E}(i)$  but  $t_2 \notin \mathbf{E}(j)$ , Enabling Memory and Resampling are equivalent, we indicate them with  $e_{t_1, t_2}(i) = R$  and Age Memory with  $e_{t_1, t_2}(i) = C$ .

It is natural to extend the definition of  $e$ ,  $\hat{w}$ , and  $\tau$  to firing sequences  $s = (t_1, \dots, t_n) \in T^n$ . Let  $i_1$  be a marking and  $t$  a transition in  $T^D$ :

$$\begin{aligned} e_{s,t}(i_1) &= \begin{cases} C & \text{if } n > 0 \wedge \forall k, 1 \leq k \leq n, \\ & i_{k+1} = \mathbf{f}(t_k, i_k), e_{t_k, t}(i_k) = C \\ R & \text{if } s = t \vee (n > 0 \wedge \forall k, 1 \leq k \leq n, i_{k+1} = \\ & \mathbf{f}(t_k, i_k) \wedge \exists l, 1 \leq l \leq n, e_{t_l, t}(i_l) = R) \\ O & \text{otherwise} \end{cases} \\ \hat{w}_s(i_1) &= \begin{cases} \prod_{k=1}^n \hat{w}_{t_k}(i_k) & \text{if } s \in T^{Z^*} \wedge \forall k, 1 \leq k \leq n, \\ & i_{k+1} = \mathbf{f}(t_k, i_k) \\ 0 & \text{otherwise} \end{cases} \\ \tau_s(i_1) &= \begin{cases} \frac{\tau_{t_1}(i_1)}{\hat{w}_{(t_2, \dots, t_n)}(\mathbf{f}(t_1, i_1))} & \text{if } s \in T^E T^{Z^*} \wedge t_1 \in \mathbf{E}(i_1) \\ \infty & \text{otherwise} \end{cases} \end{aligned}$$

With the algorithm described in Section 3,  $e_{t_1, t_2}(i) = R$  can always be used, but  $e_{t_1, t_2}(i) = C$

can be used only if  $j \in \mathcal{S}^Z$  or  $t_2 \in \mathbf{E}(j)$ . Furthermore, if  $j \in \mathcal{S}^Z$  and  $t_3 \in \mathbf{E}(j)$  fires  $e_{t_3, t_2}(j)$  can also be set to  $C$ , even if  $t_2 \notin \mathbf{E}(j)$ : the work done by  $t_2$  in  $i$  is not lost even if it becomes disabled, as long as it is disabled only in vanishing markings.

The Resampling policy has been dismissed in [1] for being “of little interest in practical applications”, but this is true only if it used for every transition in every marking. If the choice between  $R$  and  $C$  can be made selectively, the  $R$  policy becomes instead useful in conjunction with non-memoryless distributions. For example, the example in Section 2.3 uses the  $R$  policy to restart a timeout timer when noise is detected by the transmitter. The transition modeling the timer has a deterministic distribution and is enabled both before and after the detection of noise, but its RFT must restart after this detection.

In general, a DSPN model is used to compute one or more measures, defined by associating a reward rate  $\rho[i]$  to each marking  $i$ . In section 3 we show how the expected instantaneous reward rate in steady state or the expected reward accumulated in the transient markings until steady state can be obtained.

### 2.3 An example

The DSPN in Figure 1 models a simple transmission protocol. Messages to be transmitted are stored in a buffer with capacity  $K$  (place  $S$ , *source*, with  $K$  initial tokens to enforce the capacity, exponential transition  $G$ , *generate*, and place  $W$ , *wait*). Whenever the buffer is not empty and the previous transmission has completed (a token is in place  $R$ , *ready*), the transmission protocol begins (immediate transition  $I$ , *initiate*, fires). A timeout timer is started (place  $B$ , *busy*, and deterministic transition  $T$ , *timeout*) and the transmission of the message initiates (place  $M$ , *message*, and exponential transition  $X$ , *transmit*).

Once the transmission of the message has been completed ( $X$  fires and deposits a token into place  $C$ , *Check*), the receiver might either acknowledge receiving a complete message (immediate transition  $O$ , *OK*, place  $P$ , *positive acknowledgment*, and exponential transition  $A$ , *transmit acknowledgment*), or do nothing (immediate transition  $E$ , *error*, and place  $D$ , *do nothing*). If an acknowledgment is received, the protocol is completed and the transmitter can send a new message (the token put back in  $S$  indicates that the current message can be discarded and its slot is available to store a new message).

The transmitter can sense the medium while transmitting a message. If noise is sensed (exponential transition  $N$ , *noise*), the transmission is restarted and the

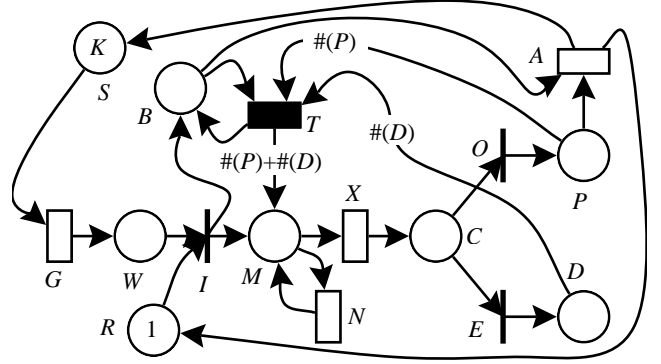


Figure 1: A protocol modeled by a DSPN.

timer is reset ( $e_{N,T} = R$ ). The timer is also reset at the beginning of a transmission ( $e_{I,T} = R$ ). No other activity affects the timer ( $e_{G,T} = e_{X,T} = e_{O,T} = e_{E,T} = C$ ).

Finally, if the timeout elapses (transition  $T$  fires) before an acknowledgment is received, the message is retransmitted (the marking-dependent multiplicity arcs from  $P$  and  $D$  to  $T$  and from  $T$  to  $M$  ensure that, after a timeout, the state of the DSPN is reinitialized to what it was at the beginning of the transmission). In practice, this type of feedback in the DSPN does not correspond to a physical feedback from the receiver, rather, it simply implies that any late acknowledgment can be recognized as such and discarded (this can be implemented by associating a sequence number to each transmission).

For the computations presented in Section 3, we assume  $K = 2$ ,  $\tau_T = 30$ ,  $\tau_G = 60$ ,  $\tau_X = 5$ ,  $\tau_A = 1$ ,  $\tau_N = 300$ ,  $\hat{w}_O = 0.99$ , and  $\hat{w}_E = 0.01$ .

### 2.4 Structural analysis

Marking-dependent arc multiplicities result in more compact models, but they reduce the applicability of invariant analysis. For example, consider the DSPN in Figure 2(a), modeling a deterministic clock which counts time modulo two ( $p_1$  can contain either zero or one token): no place invariant covering  $p_1$  exists. The same behavior is represented more conventionally by the DSPN in Figure 2(b), which uses two transitions with the same distribution as  $t_1$  and has the advantage of being covered by the place-invariant

$$\#(p_{1,0}) + \#(p_{1,1}) = 1$$

In other cases, though, marking-dependent arc multiplicities do not preclude the existence of place-invariants. The DSPN of Figure 1 is one example:

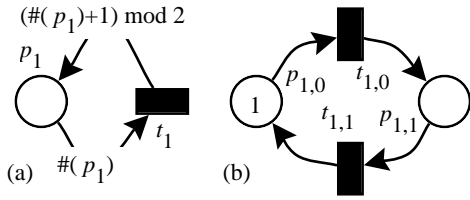


Figure 2: Marking-dependent arc multiplicities can destroy place invariants.

the place-invariants

$$\#(R) + \#(B) = 1$$

$$\#(R) + \#(M) + \#(C) + \#(P) + \#(D) = 1$$

$$\#(S) + \#(W) + \#(M) + \#(C) + \#(P) + \#(D) = K$$

can be easily verified even if marking-dependent arc multiplicities are used. The equivalent DSPN without marking-dependent arc multiplicities would be unnatural and cumbersome.

This example indicates that, under some restrictions, invariant analysis is still feasible for Petri nets with marking-dependent arc multiplicities. In this paper, though, we focus on the steady-state analysis of DSPNs, and we do not discuss this topic further.

### 3 Steady state analysis of DSPNs

#### 3.1 The numerical solution algorithm

An analytical solution approach is presented concisely in [2], for DSPNs where, for each deterministic transition  $t_d$ ,  $\tau_{t_d}$  is constant,  $e_{t_1, t_d}(i) = C$  if  $t_d$  is enabled in both  $i$  and in  $\mathbf{f}(t_1, i)$ , and only the expected reward rate in steady state is considered. The idea is to study the evolution of the DSPN while  $t_d$  is enabled in a continuous-time Markov chain (CTMC) to be solved at the transient time  $\tau_{t_d}$ , at which time  $t_d$  must fire unless it has been disabled in the meantime. Then, an embedded Markov chain (EMC) is used, which observes the DSPN at any change of marking when no deterministic transition is enabled, but only at the time a deterministic transition either fires or becomes disabled, if one is enabled.

In [12], it is noted that a different CTMCs should be defined for each deterministic transition  $t_d$ , the “subordinated Markov chain (SMC) of  $t_d$ ”. In [13], we showed how marking-dependent  $\tau_{t_d}$  can be handled by scaling the time index of the corresponding SMC.

The outgoing rates for state  $i$  are multiplied by  $\tau_{t_d}(i)$  and the transient solution at time 1 is computed for this scaled SMC.

In the following, we retain the idea of the SMC, but we associate it with each marking  $i$  where the RFT of a deterministic transition  $t_d$  can be resampled. This results in a greater number of SMCs, but each of them is at most as large as the corresponding SMC for  $t_d$  and must be solved only once, with  $i$  as the initial state. In addition to reduce the size of the data structures being manipulated, this increases the parallelism, since each SMC can be solved independently.

As our extensions, formalization, and improvements require a substantially different notation from the one used in [2, 12, 13], we present a detailed description of an algorithm for steady state analysis of DSPNs. We use “elimination” of the vanishing markings: they do not appear in the state spaces of the EMC or of the SMCs. An alternative approach, “preservation”, can be used for the steady-state solution, where vanishing markings are explicitly considered [8].

**Step 1.** Build the reachability graph  $(\mathcal{S}, \mathcal{A})$  of the DSPN ignoring the timing information associated to the transitions. If a marking with two or more enabled deterministic transitions is found, exit: the algorithm does not handle this case.

In our example, the reachability graph is shown in Figure 3. For each marking, the places having one token in them are listed, with the exception of place  $S$ , for which the number of tokens is determined by the invariants on the DSPN. Normal lines are used for exponential transition firings and markings in  $\mathcal{S}^E$ ; dashed lines are used for immediate transition firings and markings in  $\mathcal{S}^Z$ ; heavy lines are used for deterministic transition firings and markings in  $\mathcal{S}^D$ .

**Step 2.** Find the set of markings where the RFT of a deterministic transition is resampled. These are the only markings of  $\mathcal{S}^D$  which are also states of the EMC:

$$\mathcal{E}^D = \{i \in \mathcal{S}^D : \exists j \in \mathcal{S}^{ED}, \exists s \in (T^E \cup T^D)T^{Z*},$$

$$i = \mathbf{f}(s, j) \wedge e_{s, d(i)}(j) = R\}$$

In our example,  $\mathcal{E}^D = \{BM, BMW\}$ .

**Step 3.**  $\forall i \in \mathcal{E}^D$  define and solve the scaled SMC  $\{\mu_i(\theta), \theta \geq 0\}$  associated to marking  $i$ :

**Step 3.1.** Find

$$\mathcal{S}_i^C = \{j \in \mathcal{S}^D : \exists s \in (T^Z \cup T^E)^*,$$

$$j = \mathbf{f}(s, i) \wedge e_{s, d(i)}(i) = C\}$$

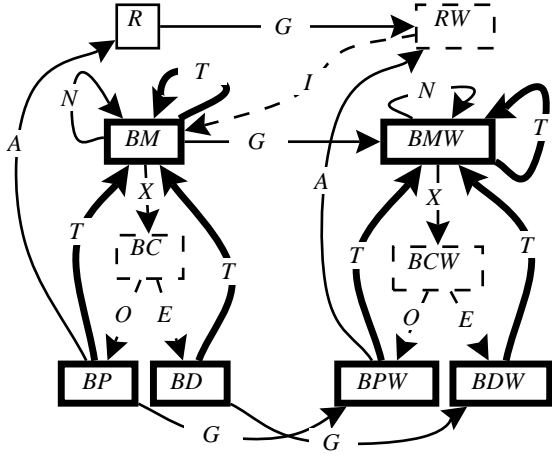


Figure 3: The reachability graph.

$$\mathcal{S}_i^R = \{l \in \mathcal{S}^{ED} : \exists j \in \mathcal{S}_i^C, \exists s \in T^E T^{Z*}, \\ l = \mathbf{f}(s, j) \wedge e_{s, d(i)}(j) = R\}$$

$\mathcal{S}_i^C$  is the set of tangible markings that can be reached from marking  $i$  without restarting  $d(i)$ , by firing only immediate or exponential transitions, while  $\mathcal{S}_i^R$  is the set of tangible markings reachable from  $\mathcal{S}_i^C$  by firing one exponential transition followed by a sequence of immediate transitions, such that  $d(i)$  is restarted without having fired.

In our example,

$$\begin{aligned} \mathcal{S}_{BM}^C &= \{BM, BP, BD, BMW, BPW, BDW\} \\ \mathcal{S}_{BM}^R &= \{R, BM, BMW\} \\ \mathcal{S}_{BMW}^C &= \{BMW, BPW, BDW\} \\ \mathcal{S}_{BMW}^R &= \{BM, BMW\} \end{aligned}$$

$\mathcal{S}_i^C$  and  $\mathcal{S}_i^R$  are not necessarily disjoint. The state space  $\mathcal{S}_i$  of  $\{\mu_i(\theta), \theta \geq 0\}$  must include both  $\mathcal{S}_i^C$  and  $\mathcal{S}_i^R$ , but there must be a way to distinguish between their elements even if they correspond to the same marking. For this reason, we define  $\mathcal{S}_i = (\mathcal{S}_i^C \times C) \cup (\mathcal{S}_i^R \times R)$  and add  $C$  or  $R$  as a superscript to the marking. For example,

$$\mathcal{S}_{BMW} = \{BMW^C, BPW^C, BDW^C, BM^R, BMW^R\}$$

**Step 3.2.** Define the total rate from  $i_1$  to  $i_2$  due to the firing of an exponential transition followed by a sequence of immediate transitions:

$$\forall i_1, i_2 \in \mathcal{S}^{ED}, \Lambda[i_1, i_2] = \sum_{s \in T^E T^{Z*}: i_2 = \mathbf{f}(s, i_1)} \tau_s(i_1)^{-1}$$

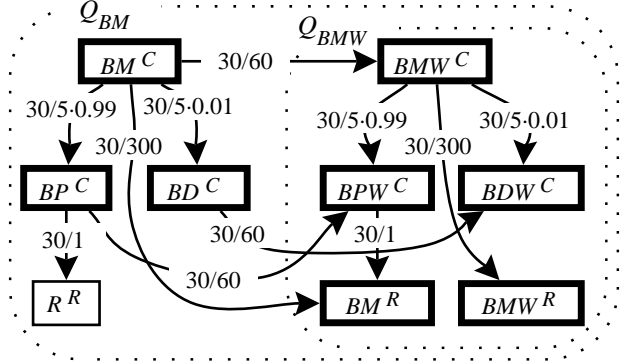


Figure 4:  $Q_{BM}$  and  $Q_{BMW}$ .

If  $i_1 \in \mathcal{S}^D$ , we must distinguish whether the sequences  $s$  continues or restarts  $d(i_1)$ :  $\forall i_1 \in \mathcal{S}^D, \forall i_2 \in \mathcal{S}^{ED}$ ,

$$\begin{aligned} \Lambda^C[i_1, i_2] &= \sum_{\substack{s \in T^E T^{Z*} : i_2 = \mathbf{f}(s, i_1), \\ e_{s, d(i_1)}(i_1) = C}} \tau_s(i_1)^{-1} \\ \Lambda^R[i_1, i_2] &= \sum_{\substack{s \in T^E T^{Z*} : i_2 = \mathbf{f}(s, i_1), \\ e_{s, d(i_1)}(i_1) = R}} \tau_s(i_1)^{-1} \end{aligned}$$

Adopting the convention  $\forall i_1 \in \mathcal{S}^E, \forall i_2 \in \mathcal{S}^{ED}$ ,  $\Lambda^C[i_1, i_2] = \Lambda^R[i_1, i_2] = 0$ , the total outgoing rate from  $i_1$  due to the firing of exponential transitions is

$$\forall i_1 \in \mathcal{S}^{ED}, \Lambda^\Sigma[i_1] = \Lambda^R[i_1, i_1] + \sum_{i_2 \in \mathcal{S}_i, i_2 \neq i_1} \Lambda[i_1, i_2]$$

This rate takes into account all the firing sequences  $s \in T^E T^{Z*}$  from  $i_1$  with the exception of those returning to  $i_1$  without restarting a deterministic transition.

The infinitesimal generator of the SMC is then:

$$\forall i_1^{a_1}, i_2^{a_2} \in \mathcal{S}_i, Q_i[i_1^{a_1}, i_2^{a_2}] = \begin{cases} 0 & \text{if } a_1 = R \\ \tau_{d(i)}(i_1) \Lambda^{a_2}[i_1, i_2] & \text{if } a_1 = C, i_1^{a_1} \neq i_2^{a_2} \\ -\Lambda^\Sigma[i_1] & \text{otherwise} \end{cases}$$

Figure 4 shows the infinitesimal generators for our example,  $Q_{BM}$  and  $Q_{BMW}$  (the latter is contained in the former).

**Step 3.3.** Solve the scaled SMC to compute [13]:

$$\begin{aligned} \forall j^a \in \mathcal{S}_i, \pi_i[j^a] &= \Pr\{\mu_i(1) = j^a | \mu_i(0) = i^C\} \\ &= (e^{Q_i})[i^C, j^a] \end{aligned}$$

$$\forall j \in \mathcal{S}_i^C, \sigma_i[j^C] = \tau_{d(i)}(j) \int_0^1 \Pr\{\mu_i(\theta) = j^C | \mu_i(0) = i^C\} d\theta$$

$$= \tau_{d(i)}(j) \int_0^1 (e^{Q_i \theta}) [i^C, j^C] d\theta$$

Assume that the DSPN enters marking  $i$  and that  $d(i)$  is restarted at that time:  $\pi_i[j^C]$  is the probability that  $d(i)$  fires while the DSPN is in marking  $j$ ;  $\pi_i[j^R]$  is the probability that the DSPN restarts  $d(i)$  by reaching marking  $j$ ;  $\sigma_i[j^C]$  is the expected time spent in marking  $j$ , until  $d(i)$  either fires or is restarted.

Efficient computational formulas for  $\pi_i$  and  $\sigma_i$  based on Jensen's method, also called Uniformization or Randomization, [9] have been introduced in [12]. Due to the particular nature of the SMCs, though, the matrix  $Q_i$  has often some special structure and its transient study can be performed using closed-form expressions. In particular, if there are no exponential transitions concurrently enabled with the deterministic transition, two cases can be considered:

- $S_i^C = \{i\}$ ,  $S_i^R = \emptyset$ : the SMC contains a single state, no exponential transition is either concurrently or competitively enabled with  $d(i)$ :

$$\pi_i[i^C] = 1 \quad \sigma_i[i^C] = \tau_{d(i)}(i)$$

- $S_i^C = \{i\}$ ,  $S_i^R \neq \emptyset$ : one or more exponential transition is competitively enabled with  $d(i)$ , if one of them fires,  $d(i)$  is restarted:

$$\begin{aligned} \pi_i[i^C] &= e^{-\tau_{d(i)}(i)\Lambda^\Sigma[i]} \\ \forall j \in S_i^R, \pi_i[j^R] &= \left(1 - e^{-\tau_{d(i)}(i)\Lambda^\Sigma[i]}\right) \frac{\Lambda^R[i, j]}{\Lambda^\Sigma[i]} \\ \sigma_i[i^C] &= \left(1 - e^{-\tau_{d(i)}(i)\Lambda^\Sigma[i]}\right) \frac{1}{\Lambda^\Sigma[i]} \end{aligned}$$

In our example, we obtain:

State	$\pi_{BM}$	$\sigma_{BM}$	$\pi_{BMW}$	$\sigma_{BMW}$
$BM^C$	0.001360	4.53927	-	-
$BP^C$	0.000338	0.88371	-	-
$BD^C$	0.005952	0.18756	-	-
$BMW^C$	0.000882	0.36773	0.002243	4.90700
$BPW^C$	0.000219	0.08732	0.000557	0.97103
$BDW^C$	0.003861	0.05925	0.009814	0.24682
$R^R$	0.883709	-	-	-
$BM^R$	0.102451	-	0.971029	-
$BMW^R$	0.001225	-	0.016356	-

**Step 4.** Define the state space  $\mathcal{E}$  of the EMC  $\{\mu_{\mathcal{E}}(k), k \in \mathbb{N}\}$  as  $\mathcal{E} = \mathcal{E}^D \cup \mathcal{S}^E$  and the holding time vector  $h_{\mathcal{E}}$  as

$$\forall i \in \mathcal{E}, h_{\mathcal{E}}[i] = \begin{cases} \Lambda^\Sigma[i]^{-1} & \text{if } i \in \mathcal{S}^E \\ \sum_{j \in S_i^C} \sigma_i[j^C] & \text{if } i \in \mathcal{E}^D \end{cases}$$

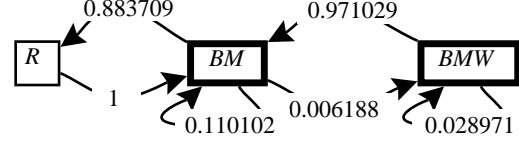


Figure 5:  $\Pi_{\mathcal{E}}$ .

In our example,  $\mathcal{E} = \{R, BM, BMW\}$  and

$$h_{\mathcal{E}}[R, BM, BMW] = [60, 6.12483, 6.12485]$$

**Step 5.** Define the probability of going from  $i_1$  to  $i_2$  in the EMC due to the firing of any number of immediate transitions as:

$$\forall i_1 \in \mathcal{S}^Z, \forall i_2 \in \mathcal{E}, \hat{w}[i_1, i_2] = \sum_{s \in T^{Z^*}: i_2 = \mathbf{f}(s, i_1)} \hat{w}_s(i_1)$$

The transition probability matrix  $\Pi_{\mathcal{E}}$  of  $\{\mu_{\mathcal{E}}(k), k \in \mathbb{N}\}$  is then:

$$\forall i, j \in \mathcal{E}, \Pi_{\mathcal{E}}[i, j] = \begin{cases} \Lambda[i, j] / \Lambda^\Sigma[i] & \text{if } i \in \mathcal{S}^E \wedge i \neq j \\ 0 & \text{if } i \in \mathcal{S}^E \wedge i = j \\ \pi_i[j^R] + \sum_{l \in S_i^C} \pi_i[l^C] \hat{w}[\mathbf{f}(d(i), l), j] & \text{if } i \in \mathcal{E}^D \wedge j \in S_i^R \\ \sum_{l \in S_i^C} \pi_i[l^C] \hat{w}[\mathbf{f}(d(i), l), j] & \text{if } i \in \mathcal{E}^D \wedge j \notin S_i^R \end{cases}$$

Figure 5 shows  $\Pi_{\mathcal{E}}$  for our example.

**Step 6.** Solve the EMC to compute either the steady-state probabilities of its recurrent states or the expected total number of visits in its transient states up to steady state

$$\forall i \in \mathcal{E}, \pi_{\mathcal{E}}[i] = \lim_{k \rightarrow \infty} \Pr\{\mu_{\mathcal{E}}(k) = i | \mu_{\mathcal{E}}(0)\}$$

$$\forall i \in \mathcal{E}, i \text{ transient}, n_{\mathcal{E}}[i] = \sum_{k=0}^{\infty} \Pr\{\mu_{\mathcal{E}}(k) = i | \mu_{\mathcal{E}}(0)\}$$

$n_{\mathcal{E}}$  is computed as the solution of  $n_{\mathcal{E}}(I - \Pi_{\mathcal{E}}^*) = \pi_{\mathcal{E}}^*(0)$  where  $\Pi_{\mathcal{E}}^*$  and  $\pi_{\mathcal{E}}^*(0)$  are the restrictions of  $\Pi_{\mathcal{E}}$  and the initial probability vector  $\pi_{\mathcal{E}}(0)$  to the transient markings in  $\mathcal{E}$  (it is often assumed that  $\Pr\{\mu_{\mathcal{E}}(0) = M_0\} = 1$ , but this is neither useful nor necessary). If  $\mathcal{E}$  contains a single recurrent class,  $\pi_{\mathcal{E}}$  is computed as the solution of  $\pi_{\mathcal{E}} = \pi_{\mathcal{E}} \Pi_{\mathcal{E}}$ . An efficient solution approach when multiple recurrent classes exist is discussed in [6], which also contains an iterative method for the

numerical solution of these linear systems, “optimal” Successive Over-Relaxation.

In our example, there are no transient markings, so only  $\pi_{\mathcal{E}}$  is computed and the initial probability distribution is irrelevant:

$$\pi_{\mathcal{E}}[R, BM, BMW] = [0.467550, 0.529077, 0.003373]$$

**Step 7.** Compute the probabilities  $\pi[i]$  for each tangible marking  $i \in \mathcal{S}^{ED}$  of the DSPN in steady state:

$$\pi[i] = \begin{cases} \frac{\pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]}{\sum_{j \in \mathcal{E}} \pi_{\mathcal{E}}[j]h_{\mathcal{E}}[j]} & \text{if } i \in \mathcal{S}^E \\ \frac{\sum_{j \in \mathcal{E}^D: i \in \mathcal{S}_j^C} \pi_{\mathcal{E}}[j]\sigma_j[i]}{\sum_{j \in \mathcal{E}} \pi_{\mathcal{E}}[j]h_{\mathcal{E}}[j]} & \text{if } i \in \mathcal{S}^D \end{cases}$$

or the cumulative sojourn times  $\sigma[i]$  for each tangible transient marking  $i \in \mathcal{S}^{ED}$  of the DSPN up to steady state:

$$\sigma[i] = \begin{cases} n_{\mathcal{E}}[i]h_{\mathcal{E}}[i] & \text{if } i \in \mathcal{S}^E \\ \sum_{j \in \mathcal{E}^D: i \in \mathcal{S}_j^C} n_{\mathcal{E}}[j]\sigma_j[i] & \text{if } i \in \mathcal{S}^D \end{cases}$$

In our example, only  $\pi$  needs to be computed:

$$\begin{aligned} \pi[R] &= 0.895856 & \pi[BM] &= 0.076695 \\ \pi[BP] &= 0.014931 & \pi[BD] &= 0.003169 \\ \pi[BMW] &= 0.006742 & \pi[BPW] &= 0.001580 \\ \pi[BDW] &= 0.001028 \end{aligned}$$

**Step 8.** Compute the expected reward rate in steady state or the cumulative reward until steady state:

$$\sum_{i \in \mathcal{S}} \pi[i]\rho[i] \quad \sum_{i \in \mathcal{S}} \sigma[i]\rho[i]$$

In our example, only the expected reward rate in steady state is meaningful. Measures of interest could be the average number of messages waiting for the transmitter to become ready, obtained by using the reward rate specification  $\rho[i] = \#(W, i)$ , resulting in a value of 0.009349, or the throughput of the transmitter, defined as the rate of messages acknowledged, is obtained by defining the reward rate specification  $\rho[i] = \tau_G(i)^{-1}$ , resulting in a value of 0.016511.

### 3.2 Exploiting the reward structure

We now present a modification to the algorithm of the previous section, which allows to store permanently only the markings in  $\mathcal{E}$ . To avoid storing the

vanishing markings, “elimination on the fly” can be used, a method previously proposed for GSPNs [8]. In our case, this would correspond to storing only the tangible markings and the matrices  $\Lambda$  and  $\Lambda^C$ , instead of storing the entire reachability set and reachability graph. This is possible because  $\forall i \in \mathcal{S}^Z, \pi_i = \sigma_i = 0$ , hence vanishing markings do not contribute to the computation of the measures.

The elimination on the fly of markings  $i \in \mathcal{S}^D \setminus \mathcal{E}^D$  requires more care, because  $\pi_i$  and  $\sigma_i$  are not zero in this case. During the reachability graph generation, as soon as a marking  $i \in \mathcal{E}^D$  is found, the set of markings  $\mathcal{S}_i$  can be generated and the SMC  $\{\mu_i(\theta), \theta \geq 0\}$  can be solved as before, to compute the vectors  $\pi_i$  and  $\sigma_i$ . But then, the information in  $\sigma_i$  can be distilled into the “equivalent reward rate of the EMC marking  $i$ ”,  $\rho_{\mathcal{E}}[i]$ , defined as:

$$\rho_{\mathcal{E}}[i] = \sum_{j \in \mathcal{S}_i^C} \sigma_i[j]\rho[j]$$

After finding the markings reachable from  $\mathcal{S}_i^C$  by firing any sequence  $s \in T^D T^{Z*}$ , the markings in  $\mathcal{S}_i^D \setminus \{i\}$  and  $\sigma_i$  can be destroyed. Only  $i$ ,  $\rho_{\mathcal{E}}[i]$ ,  $\pi_i$ , and the markings in  $\mathcal{S}_i^R$  need to be kept. To compute the expected reward rate in steady state, consider that

$$\begin{aligned} & \sum_{i \in \mathcal{S}^{ED}} \pi[i]\rho[i] \\ &= \sum_{i \in \mathcal{S}^E} \pi[i]\rho[i] + \sum_{i \in \mathcal{S}^D} \pi[i]\rho[i] \\ &= \sum_{i \in \mathcal{S}^E} \frac{\pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]}{\sum_{i \in \mathcal{E}} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]} \rho[i] + \sum_{i \in \mathcal{S}^D} \frac{\sum_{j \in \mathcal{E}^D: i \in \mathcal{S}_j^C} \pi_{\mathcal{E}}[j]\sigma_j[i]}{\sum_{i \in \mathcal{E}} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]} \rho[i] \\ &= \frac{\sum_{i \in \mathcal{S}^E} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]\rho[i] + \sum_{j \in \mathcal{E}^D} \sum_{i \in \mathcal{S}_j^C} \pi_{\mathcal{E}}[j]\sigma_j[i]\rho[i]}{\sum_{i \in \mathcal{E}} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]} \\ &= \frac{\sum_{i \in \mathcal{S}^E} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]\rho[i] + \sum_{j \in \mathcal{E}^D} \pi_{\mathcal{E}}[j] \sum_{i \in \mathcal{S}_j^C} \sigma_j[i]\rho[i]}{\sum_{i \in \mathcal{E}} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]} \\ &= \frac{\sum_{i \in \mathcal{S}^E} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]\rho[i] + \sum_{j \in \mathcal{E}^D} \pi_{\mathcal{E}}[j]\rho_{\mathcal{E}}[j]}{\sum_{i \in \mathcal{E}} \pi_{\mathcal{E}}[i]h_{\mathcal{E}}[i]} \end{aligned}$$

The last expression uses only the probabilities  $\pi_{\mathcal{E}}$  of the EMC. This modified approach has the advantage of a smaller total memory requirement: at most

the information relative to the EMC and one SMC is stored at any time.

If a marking  $j$  belongs to both  $\mathcal{S}_{i_1}^C$  and  $\mathcal{S}_{i_2}^C$ ,  $j$  and all the markings not in  $\mathcal{E}$  reachable from it will be “rediscovered”, since they are not stored. This problem is analogous to that encountered when performing elimination of the vanishing markings on the fly. If the cost of this rediscovery is too high, a different version of this approach can be employed, where all the markings are stored (thus eliminating the problem of rediscovery), but  $\sigma_i$  is destroyed after having computed  $\rho_{\mathcal{E}}[i]$ . The DSPN solution algorithm currently implemented in the software package DSPN-express [11] requires to store the entire collection of vectors  $\{\sigma_i, i \in \mathcal{E}^D\}$ , forming the so-called “conversion matrix” [2]. With our modified approach, only one vector  $\sigma_i$  at a time needs to be stored. This represents a substantial savings, since  $\sigma_i$  is typically a full vector.

#### 4 Comparison and concluding remarks

The possibility that  $\mathcal{S}_i^C \cap \mathcal{S}_i^R \neq \emptyset$ , has not been addressed in [2], but this can happen even if only the Enabling Memory policy without any marking-dependence is allowed for specifying the remaining firing time of a deterministic transition. Consider the DSPN in Figure 6(a) and its reachability graph in Figure 6(b). Marking  $[p_2p_5]$  can be reached from  $[p_2p_3]$  by firing  $t_3$ , which does not restart  $t_4$ , hence  $[p_2p_5] \in \mathcal{S}_{[p_2p_3]}^C$ , but it can also be reached by firing  $t_2$  and then  $t_1$ , a sequence which restarts  $t_4$  since  $t_4$  is not enabled in  $[p_1]$ , hence  $[p_2p_5] \in \mathcal{S}_{[p_2p_3]}^R$ . The algorithm presented in Section 3, applies to any DSPN, even those with a marking-dependent choice between restarting and continuing the RFT of a deterministic transition.

Previously, a SMC  $\{\mu_t(\theta), \theta \geq 0\}$  was defined for a deterministic transition  $t \in T^D$  [12], for which the state space is denoted in the sequel by  $\mathcal{S}_t$ . We define instead a SMC for each marking where a deterministic transition can be restarted. Moreover, we observe that not all the states in  $\mathcal{S}_t \cap \mathcal{S}^D$  need to be considered as potential initial states. This has been mentioned in [12] for a particular DSPN, but has never been formalized in general. We define the subset of states  $\mathcal{E}^D \subseteq \mathcal{S}^D$  for which a transient solution needs to be computed. Even if the previous approaches had defined the set  $\mathcal{E}^D$  and had performed only the  $|\mathcal{E}^D|$  transient solutions which are really needed, our choice of associating the SMC to states of  $\mathcal{E}^D$  instead of

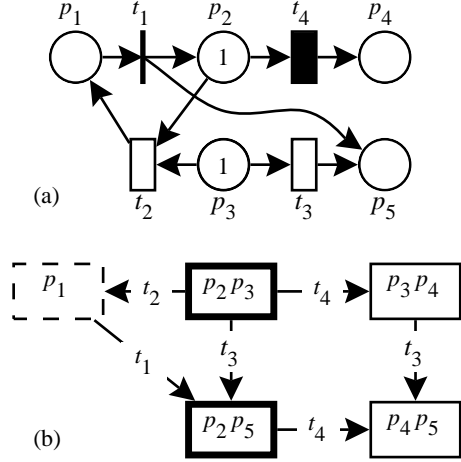


Figure 6:  $\mathcal{S}_i^C \cap \mathcal{S}_i^R \neq \emptyset$  with Enabling Memory.

to the deterministic transitions is advantageous. Assume that the matrix describing  $\{\mu_t(\theta), \theta \geq 0\}$  has  $\eta_t$  nonzero entries and that Jensen’s method is used. The previous approaches require  $m_t = |\mathcal{S}_t \cap \mathcal{E}^D|$  different transient solutions for each  $t \in T^D$ , with a total computational cost  $\sum_{t \in T^D} O(m_t \eta_t q_t)$ , where  $q_t$  is the maximum rate leaving any state  $i \in \mathcal{S}_t$  [9].

The algorithm of Section 3 requires instead to solve once a SMC  $\{\mu_i(\theta), \theta \geq 0\}$  for each marking  $i \in \mathcal{E}^D$ , with state space  $\mathcal{S}_i$  and  $\eta_i$  entries. The total number of transient solutions to be computed is the same, since  $\sum_{t \in T^D} m_t = |\mathcal{E}^D|$ , but there are both memory and computational savings:

- For each state  $i \in \mathcal{S}_t \cap \mathcal{E}^D$ , we have  $|\mathcal{S}_i| \leq |\mathcal{S}_t|$  and  $\eta_i \leq \eta_t$ . Thus, the iterations of Jensen’s method in the transient analysis typically use smaller data structures.
- The computational effort of the transient analysis is now  $\sum_{i \in \mathcal{E}^D} O(\eta_i q_i)$ , where  $q_i$  is the maximum rate leaving any state  $j \in \mathcal{S}_i \subseteq \mathcal{S}_d$ . Since  $\forall i \in \mathcal{S}_d \cap \mathcal{E}^D, \eta_i \leq \eta_d$  and  $q_i \leq q_d$ , this typically reduces the computational effort, especially for acyclic SMCs.
- In the current version of the software package DSPNexpress [11], a distributed implementation is provided, where the SMCs  $\{\mu_t(\theta), \theta \geq 0\}$  can be solved in parallel, but only after building the entire reachability graph. The approach described in Section 3 substantially increases the parallelism, since the transient analysis of a SMC

$\{\mu_i(\theta), \theta \geq 0\}$  can start immediately after  $\mathcal{S}_i$  has been built, even if only a portion of the reachability graph has been explored.

We also presented an approach for reducing the memory requirements of the numerical solution algorithm of DSPNs. The method is based on the “equivalent reward rate” of each marking  $i \in \mathcal{E}^D$  and avoids storing the entire state space and the conversion matrix.

## References

- [1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of Stochastic Petri Nets. *IEEE Trans. Softw. Eng.*, 15(7):832–846, July 1989.
- [2] M. Ajmone Marsan and G. Chiola. On Petri Nets with deterministic and exponentially distributed firing times. In G. Rozenberg, editor, *Adv. in Petri Nets 1987, Lecture Notes in Computer Science 266*, pages 132–145. Springer-Verlag, 1987.
- [3] E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.
- [4] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov regenerative stochastic Petri nets. In G. Iazeolla and S. S. Lavenberg, editors, *Performance '93*. North-Holland, Sept. 1993.
- [5] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Transient analysis of deterministic and stochastic Petri nets. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993, Lecture Notes in Computer Science*, volume 691, pages 166–185. Springer-Verlag, 1993.
- [6] G. Ciardo, A. Blakemore, P. F. J. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of Markov reward models using Stochastic Reward Nets. In C. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, volume 48 of *IMA Volumes in Mathematics and its Applications*, pages 145–191. Springer-Verlag, 1993.
- [7] G. Ciardo, R. German, and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. In *Proc. of the Fifth Int. Workshop on Petri Nets and Performance Models (PNPM93)*, Toulouse, France, Oct. 1993.
- [8] G. Ciardo, J. Muppala, and K. S. Trivedi. On the solution of GSPN reward models. *Perf. Eval.*, 12(4):237–253, 1991.
- [9] D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Oper. Res.*, 32(2):926–944, Mar.-Apr. 1984.
- [10] M. A. Holliday and M. K. Vernon. A Generalized Timed Petri Net model for performance analysis. *IEEE Trans. Softw. Eng.*, 12:1297–1310, 1987.
- [11] C. Lindemann. DSPNexpress: A software package for the efficient solution of Deterministic and Stochastic Petri Nets. In *Proc. 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 15–29, Edinburgh, Great Britain, 1992.
- [12] C. Lindemann. An improved numerical algorithm for calculating steady-state solutions of Deterministic and Stochastic Petri Net models. *Perf. Eval.*, 8, 1993.
- [13] C. Lindemann and R. German. Modeling discrete event systems with state-dependent deterministic service times. *Discrete Event Dynamic Systems: Theory and Applications*, 3:249–270, 1993.
- [14] W. L. Zuberek. Timed Petri nets definitions, properties, and applications. *Microelectronics and Reliability*, 31:627–644, 1991.