

A Decomposition Approach for Stochastic Reward Net Models *

Gianfranco Ciardo

Kishor S. Trivedi

Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA

Department of Electrical Engineering
Duke University
Durham, NC 27706, USA

Abstract

We present a decomposition approach for the solution of large stochastic reward nets (SRNs) based on the concept of near-independence. The overall model consists of a set of submodels whose interactions are described by an *import graph*. Each node of the graph corresponds to a parametric SRN submodel and an arc from submodel A to submodel B corresponds to a parameter value that B must receive from A . The quantities exchanged between submodels are based on only three primitives. The import graph normally contains cycles, so the solution method is based on fixed point iteration. Any SRN containing one or more of the nearly-independent structures we present, commonly encountered in practice, can be analyzed using our approach. No other restriction on the SRN is required. We apply our technique to the analysis of a flexible manufacturing system.

Short Title:

A Decomposition Approach for Stochastic Reward Net Models.

Keywords:

Stochastic Petri Nets,
Stochastic Reward Processes,
Approximate Solution of Continuous Time Markov Chains.

*This work was supported in part by the National Science Foundation under Grant CCR-9108114 and by the Naval Surface Warfare Center.

1 Introduction

Stochastic Petri nets (SPNs) of various ilk are found to be powerful in modeling performance and dependability of computer and communications systems. If the SPN is restricted to be Markovian, then a wealth of numerical solution algorithms of Markov models [3] can be used. The major drawback in this approach, however, is that a large state space of the underlying Markov model needs to be generated, stored, and processed. The generation of a large state space is avoided by resorting to Monte-Carlo simulation. In addition, simulation allows the treatment of non-Markovian behaviors as well. The major drawbacks of simulation are the potentially large execution time needed and the necessity for statistical estimation.

We define the stochastic reward nets (SRNs) as a class of Markovian SPNs where the model definition includes the structural and timing behavior of the system, as well as the specification of the measures to be computed from the analysis of the model. As we allow parametric model specifications, an SRN model with m parameters and n measures can be seen as a function from \mathbb{R}^m to \mathbb{R}^n .

SRNs constitute a powerful modeling formalisms, but their solution is still limited by the size of the underlying Markov reward process. An attractive approach is the use of model decomposition. We visualize the scenario in which the system has been decomposed into subsystems and an SRN submodel is developed for each subsystem. The modeler then composes an overall model from the constituent submodels, carefully taking into account various interactions among subsystems. Our approach consists of taking advantage of this natural decomposition by solving subsystems in isolation and exchanging measures as and when required. These interactions are described by an import graph. In a few lucky cases, submodel solutions can be ordered so that no feedback is necessary. In the general case, however, submodels will inter-communicate, giving rise to cycles in the import graph. Fixed-point iteration is then required in practice to resolve such cyclic situations [4, 10].

Three important issues arise: how to decompose a net into subnets, what kinds of measures must be passed from one subnet to another, and what is the convergence behavior of the iterative schemes. We do not present an automatic method of decomposing a net. Instead, we present several common types of SRN structures that can be decomposed. We show that the probability that a subnet is in a state satisfying a given condition, the average time a given condition remains satisfied, and the expected time until the subnet satisfies a given condition are three quantities that suffice for inter-communication among subnets for the net structure types that we define. Convergence of the iterative schemes is discussed elsewhere [10].

In Section 2, we informally describe the SRN formalism. In Section 3, we introduce the concepts of near-decomposability and near-independence. In Section 4, we introduce our approach based on near-independence. In Section 5, we illustrate our approach using a model of a flexible-manufacturing system. In Section 6, we present some concluding remarks. The Kronecker algebra concepts used in this paper are described in the appendix.

2 SRN concepts

There are several definitions for Petri nets [20, 21] and even more for stochastic Petri nets. Our SRN formalism allows only exponentially distributed or constant zero firing times, so its underlying stochastic process is independent semi-Markov with either exponentially distributed or constant zero

holding times. We assume that the semi-Markov process is regular, that is, the number of transition firings in a finite interval of time is finite with probability one¹. Such a process can then be transformed into a continuous-time Markov chain (CTMC) as it is done for the generalized stochastic Petri net (GSPN) formalism [1].

The SRNs differ from the GSPNs in several key aspects. From a structural point of view, both formalisms are equivalent to Turing machines. But the SRNs provide enabling functions, marking-dependent arc cardinalities, a more general approach to the specification of priorities, and the ability to decide in a marking-dependent fashion whether the firing time of a transition is exponentially distributed or null, often resulting in more compact nets. Perhaps more important, though, are the differences from a stochastic modeling point of view. The SRN formalism considers the measure specification as an integral part of the model. Underlying an SRN is an independent semi-Markov reward process with reward rates associated to the states and reward impulses associated to the transitions between states [16]. Our definition of SRN explicitly includes parameters (inputs) and the specification of multiple measures (outputs). A SRN with m inputs and n outputs defines a function from \mathbb{R}^m to \mathbb{R}^n .

We define a non-parametric SRN as an 11-tuple:

$$\mathbf{A} = \{P, T, D^-, D^+, D^\circ, g, >, \mu_0, \lambda, w, M\}$$

where:

- $P = \{p_1, \dots, p_{|P|}\}$ is a finite set of places, drawn as circles. Each place contains a non-negative number of tokens. The multiset $\mu \in \mathbb{N}^{|P|}$ describing the number of tokens in each place is called a marking. The notation $\#_{\mathbf{A}}(p, \mu)$ is used to indicate the number of tokens in place p in marking μ for the SRN \mathbf{A} . If the SRN or the marking are clear from the context, the notation $\#(p, \mu)$, $\#_{\mathbf{A}}(p)$, or $\#(p)$ is used.
- $T = \{t_1, \dots, t_{|T|}\}$ is a finite set of transitions ($P \cap T = \emptyset$), drawn as rectangles.
- $\forall p \in P, \forall t \in T, D_{p,t}^- : \mathbb{N}^{|P|} \rightarrow \mathbb{N}, D_{p,t}^+ : \mathbb{N}^{|P|} \rightarrow \mathbb{N},$ and $D_{p,t}^\circ : \mathbb{N}^{|P|} \rightarrow \mathbb{N}$ are the marking-dependent 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. If an arc multiplicity evaluates to zero in a marking, the arc is ignored (does not have any effect) in that marking. Graphically, the multiplicity is drawn on the arc. If the multiplicity is a constant equal to one, it is omitted; if it is a constant equal to zero, the arc is not drawn.

We say that a transition $t \in T$ is arc-enabled in marking μ iff

$$\forall p \in P, D_{p,t}^-(\mu) \leq \#(p, \mu) \wedge \left(D_{p,t}^\circ(\mu) > \#(p, \mu) \vee D_{p,t}^\circ(\mu) = 0 \right)$$

When transition t fires in marking μ the new marking μ' satisfies:

$$\forall p \in P, \#(p, \mu') = \#(p, \mu) - D_{p,t}^-(\mu) + D_{p,t}^+(\mu)$$

- $\forall t \in T, g_t : \mathbb{N}^{|P|} \rightarrow \{true, false\}$ is the guard associated to transition t . If $g_t(\mu) = false$, t is disabled in μ .

¹The presence of an “absorbing vanishing loop” [10] could violate this assumption, but this case has no practical interest so we do not consider it further.

- $>$ is a transitive and irreflexive relation imposing a priority among transitions. In a marking μ , t_1 is marking-enabled iff it is arc-enabled, $g_{t_1}(\mu) = true$, and no other transition t_2 exists such that $t_2 > t_1$, t_2 is arc-enabled, and $g_{t_2}(\mu) = true$. This definition is more flexible than the one adopted by other SPN formalisms, where integers are associated with transitions (e.g., imagine the situation where $t_1 > t_2$, $t_3 > t_4$, but t_1 has no priority relation with respect to t_3 and t_4).

- μ_0 is the initial marking.

- $\forall t \in T, \lambda_t : \mathbb{N}^{|P|} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is the rate of the exponential distribution for the firing time of transition t . If the rate is ∞ in a marking, the transition firing time is zero. This definition differs from the one in [1], where transitions are *a priori* classified as “timed” or “immediate”. The definition of vanishing and tangible marking, though, is still applicable: a marking μ is said to be vanishing if there is a marking-enabled transition t in μ such that $\lambda_t(\mu) = \infty$; μ is said to be tangible otherwise. Let \mathcal{V} and \mathcal{T} be the sets of vanishing and tangible markings, respectively.

We additionally impose the interpretation that, in a vanishing marking μ , all transitions t with $\lambda_t(\mu) < \infty$ are implicitly inhibited. Hence, a transition t is enabled in a marking μ (in the usual sense) and can actually fire iff it is marking-enabled and either μ is tangible or μ is vanishing and $\lambda_t(\mu) = \infty$. The notation $\mu \xrightarrow{t}$ indicates that transition t is enabled in marking μ .

We assume a race model, hence, in a tangible marking μ , the firing probability of enabled transition t is

$$\frac{\lambda_t(\mu)}{\sum_{y \in T: \mu \xrightarrow{y}} \lambda_y(\mu)}$$

- $\forall t \in T, w_t : \mathbb{N}^{|P|} \rightarrow \mathbb{R}^+$ describes the weight assigned to the firing of enabled transition t , whenever its rate λ_t evaluates to ∞ . This is needed because the race model is of no use with zero firing times. In a vanishing marking μ , the firing probability of enabled transition t is

$$\frac{w_t(\mu)}{\sum_{y \in T: \mu \xrightarrow{y}} w_y(\mu)}$$

The elements described so far define a trivariate discrete-time stochastic process: $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$, with $t^{[0]} = NULL$, $\theta^{[0]} = 0$, and $\mu^{[0]} = \mu_0$. For $n > 0$, $t^{[n]} \in T$ is the n -th transition to fire; its firing leads to $\mu^{[n]}$, the n -th marking encountered, and $\theta^{[n]}$ is the time at which it fires. Hence, $\mu^{[n-1]} \xrightarrow{t^{[n]}} \mu^{[n]}$ and $\theta^{[n]} - \theta^{[n-1]} \geq 0$ is the sojourn time for the corresponding visit in marking $\mu^{[n-1]}$. This process is a SMP where the sojourn times are either exponentially distributed, for tangible markings, or zero, for vanishing markings. It is also possible to define a continuous-time process describing the marking at time θ , $\{\mu(\theta), \theta \geq 0\}$, which is completely determined given $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$:

$$\mu(\theta) = \mu_{\max\{n: \theta^{[n]} \leq \theta\}}$$

This process considers only the evolution with respect to the tangible markings, that is, $\mu(\theta) \in \mathcal{T}$. $\{\mu(\theta), \theta \geq 0\}$ is the CTMC underlying the SRN.

The last component of an SRN specification defines the measures to be computed:

- $M = \{(\rho_1, r_1, \psi_1), \dots, (\rho_{|M|}, r_{|M|}, \psi_{|M|})\}$ is a finite set of measures, each specifying the computation of a single real value. A measure $(\rho, r, \psi) \in M$ has three components. The first and second

components specify a reward structure over the underlying stochastic process $\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$.

$$\rho : \mathbb{N}^{|\mathcal{P}|} \rightarrow \mathbb{R}$$

is a reward rate: $\rho(\mu)$ is the rate at which reward is accumulated when the marking is μ .

$$\forall t \in T, r_t : \mathbb{N}^{|\mathcal{P}|} \rightarrow \mathbb{R}$$

is a reward impulse: $r_t(\mu)$ is the instantaneous reward gained when firing transition t while in marking μ .

$\{(t^{[n]}, \theta^{[n]}, \mu^{[n]}), n \in \mathbb{N}\}$ and the reward structure specified by ρ and r define a new stochastic process $\{Y(\theta), \theta \geq 0\}$, describing the reward accumulated by the SRN up to time θ :

$$Y(\theta) = \int_0^\theta \rho(\mu(u)) du + \sum_{j=1}^{\max\{n: \theta^{[n]} \leq \theta\}} r_{t^{[j]}}(\mu^{[j-1]})$$

The third component of a measure specification, ψ , is a function that computes a single real value from $\{Y(\theta), \theta \geq 0\}$. If \mathcal{R} is the set of real-valued stochastic processes with index over the reals, then $\psi : \mathcal{R} \rightarrow \mathbb{R}$. The generality of this definition is best illustrated by showing the wide range of measures the triplet (ρ, r, ψ) can capture (in some SRNs, some of these measures might be infinite):

- Expected number of transition firings up to time θ : this is simply $E[Y(\theta)]$ when all reward rates are zero and all reward impulses are one.
- Expected time-averaged reward up to time θ :

$$E \left[\frac{Y(\theta)}{\theta} \right]$$

- Expected instantaneous reward rate at time θ :

$$E \left[\lim_{\delta \rightarrow 0} \frac{Y(\theta + \delta) - Y(\theta)}{\delta} \right]$$

- Expected accumulated reward in the steady-state:

$$E \left[\lim_{\theta \rightarrow \infty} Y(\theta) \right]$$

- Mean time to absorption: this is a particular case of the previous measure, obtained by setting the reward rate of transient and absorbing states to one and zero, respectively, and all reward impulses to zero.
- Expected instantaneous reward rate in steady-state:

$$E \left[\lim_{\theta \rightarrow \infty} \lim_{\delta \rightarrow 0} \frac{Y(\theta + \delta) - Y(\theta)}{\delta} \right]$$

which is also the same as the expected time-average reward in the steady-state:

$$E \left[\lim_{\theta \rightarrow \infty} \frac{Y(\theta)}{\theta} \right]$$

Parametric SRNs are obtained by allowing each component to depend on a set of parameters $\nu = (\nu_1, \dots, \nu_m) \in \mathbb{R}^m$:

$$\mathbf{A}(\nu) = \{P(\nu), T(\nu), D^-(\nu), D^+(\nu), D^\circ(\nu), e(\nu), >(\nu), \mu_0(\nu), \lambda(\nu), w(\nu), M(\nu)\}$$

Once the parameters ν are fixed, a simple (non-parametric) SRN is obtained.

A fundamental capability captured by this parametrization is that of specifying the initial marking not as a single marking, but as a probability vector defined over a set of markings. This is needed in our decomposition approach, but it is often required also in a transient analysis, if the initial state of the system is uncertain. If the initial marking is μ_0^i , $1 \leq i \leq K$ with probability π_0^i , the following construction will suffice:

- Add a place p_0 .
- Add transitions t_0^i , $1 \leq i \leq K$ with rate $\lambda_{t_0^i} = \infty$ and weight $w_{t_0^i} = \pi_0^i$.
- For each new transition t_0^i , set $D_{p_0, t_0^i}^- = 1$.
- For each existing place p_j , set $D_{p_j, t_0^i}^+ = \#(p_j, \mu_0^i)$.
- Set the initial marking to have one token in p_0 , no tokens elsewhere.

The modified SRN will initiate its activity by removing the token from p_0 and adding the appropriate tokens to create one of the possible initial markings, with the appropriate probability. While this is not a practical method if the number of possible initial markings K is large, it shows that the probability of being initially in one among a set of markings can be parametrized.

3 Decomposability and independence

Courtois [13] proposed the idea of decomposing a Markov model to obtain an approximate steady-state solution. The quality of the approximation is related to the degree of coupling among the blocks into which the Markov matrix is decomposed. Better approximations are obtained when the off-diagonal blocks are close to zero.

In practice, the size of the underlying Markov reward process is the main limitation to the study of SRNs, so an analysis approach that modifies the model, rather than the Markov matrix, is extremely desirable. Examples of approximate approaches that do not require the generation of the Markov chain in its entirety are state truncation [18, 28], state aggregation [19], behavioral decomposition [27], hierarchical decomposition [23], and folding [9] (the last two might not involve an approximation). These methods have proved to be difficult to generalize but are very effective, when applicable. More recently, several papers have appeared on exact solution approaches which allow either to reduce the size of the state space, by exploiting symmetries [24, 7], or at least to reduce the storage required for the Markov matrix [6, 22, 15]. These last three “decomposition” approaches are perhaps closest to ours, but fundamental differences exist. They impose strong restrictions on the possible interactions between submodels and use Kronecker algebra to store the Markov matrix in implicit form, thus reducing the storage complexity, but at best leaving unchanged the execution complexity. In any case, though, the steady-state probability vector for the exact process is still needed explicitly, and its size alone prevents the solution of large problems.

$$R^{\mathbf{C}} = \begin{bmatrix} R^{\mathbf{B}} & R_{1,2}^{\mathbf{A}} & \dots & R_{1,m}^{\mathbf{A}} \\ \dots & R_{1,2}^{\mathbf{A}} & \dots & R_{1,m}^{\mathbf{A}} \\ R_{2,1}^{\mathbf{A}} & R^{\mathbf{B}} & \dots & R_{2,m}^{\mathbf{A}} \\ \dots & R_{2,1}^{\mathbf{A}} & \dots & R_{2,m}^{\mathbf{A}} \\ \dots & \dots & \dots & \dots \\ R_{m,1}^{\mathbf{A}} & R_{m,2}^{\mathbf{A}} & \dots & R^{\mathbf{B}} \\ \dots & \dots & \dots & \\ R_{m,1}^{\mathbf{A}} & R_{m,2}^{\mathbf{A}} & \dots & \end{bmatrix}$$

Figure 1: The structure of $R^{\mathbf{C}}$.

Instead, we use Kronecker sums to illustrate our approach, but these sums are not carried out in practice, since each submodel in the decomposition is solved in isolation, thus requiring to store neither the complete Markov matrix nor the complete steady-state probability vector. The actual storage and computational savings are much greater, and this, in addition to the wide range of allowed interactions between subsystems, largely extends the practical applicability of the approach. The price to be paid is, of course, an approximation error.

3.1 Independent CTMC structures

Consider two SRNs \mathbf{A} and \mathbf{B} , with no places or transitions in common, and the SRN \mathbf{C} obtained by simply considering \mathbf{A} and \mathbf{B} together. Since \mathbf{A} and \mathbf{B} have no relation with each other, we can say that they are, in some sense, “independent subnets” in \mathbf{C} . We formalize this concept based on the structure of the transition rate matrices² of the CTMC underlying \mathbf{C} .

Define $\mathcal{T}^{\mathbf{A}}$, $R^{\mathbf{A}}$, $\mathcal{T}^{\mathbf{B}}$, $R^{\mathbf{B}}$, $\mathcal{T}^{\mathbf{C}}$, and $R^{\mathbf{C}}$ to be the the tangible reachability set and the transition rate matrix of the underlying CTMC for SRN \mathbf{A} , \mathbf{B} , and \mathbf{C} , respectively.

If $|\mathcal{T}^{\mathbf{A}}| = m$ and $|\mathcal{T}^{\mathbf{B}}| = n$, then $R^{\mathbf{A}}$ is $m \times m$ and $R^{\mathbf{B}}$ is $n \times n$. $\mathcal{T}^{\mathbf{C}}$ is the cross-product $\mathcal{T}^{\mathbf{A}} \times \mathcal{T}^{\mathbf{B}}$ and $R^{\mathbf{C}}$ can be expressed as the Kronecker sum

$$R^{\mathbf{C}} = R^{\mathbf{A}} \oplus R^{\mathbf{B}}$$

$R^{\mathbf{C}}$ is a matrix of $m \times m$ blocks of size $n \times n$ (Figure 1). Each of the m diagonal blocks is equal to $R^{\mathbf{B}}$, while the off-diagonal block in position (i,j) has zero entries except on the diagonal, where all the elements are equal to $R_{i,j}^{\mathbf{A}}$.

The same relationship holds for the infinitesimal generators: $Q^{\mathbf{C}} = Q^{\mathbf{A}} \oplus Q^{\mathbf{B}}$. We use the transition rate matrices in our discussion simply because the corresponding matrices are visually simpler.

²We use the symbols $Q = [Q_{i,j}]$ and $R = [R_{i,j}]$ to denote the infinitesimal generators and transition rate matrices of a CTMC, respectively; they only differ in their diagonal: $Q_{i,i} = -\sum_{j \neq i} Q_{i,j}$, while $R_{i,i} = 0$.

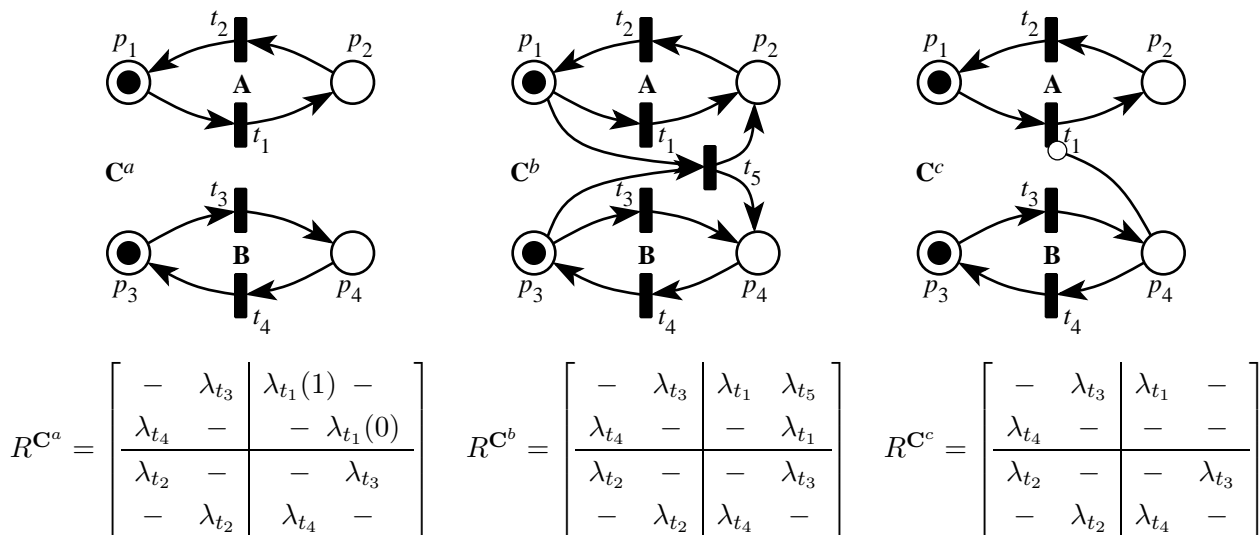


Figure 2: Near-independent CTMC structures.

The ordering of the states in $\mathcal{T}^{\mathbf{C}}$ and $R^{\mathbf{C}}$ is straightforward. When \mathbf{C} is in tangible marking having index i , \mathbf{A} is in tangible marking having index $\text{div}(i, m)$ and \mathbf{B} is in tangible marking having index $\text{mod}(i, m)$ where “div” and “mod” are the integer quotient and modulo operators, respectively.

Assuming a direct method is used, the steady-state solution of \mathbf{C} requires $O(m^3n^3)$ operations, while only $O(m^3 + n^3)$ operations are needed if \mathbf{A} and \mathbf{B} are solved separately and their probability vectors are composed using the formula

$$\pi_i^{\mathbf{C}} = \pi_{\text{div}(i, m)}^{\mathbf{A}} \pi_{\text{mod}(i, m)}^{\mathbf{B}}$$

where $\pi_j^{\mathbf{X}}$ is the probability of marking having index j for the SRN \mathbf{X} . Iterative methods offer analogous savings.

We say that a transition rate matrix having the structure of Figure 1 is *independent* and we stress that this is completely orthogonal to being *decomposable*. This is apparent when comparing the concepts of *near-independence* and *near-decomposability*³.

A transition rate matrix is *nearly-decomposable* [13] if it can be reordered and partitioned so that the entries of the off-diagonal blocks are small relative to the nonzero entries of the diagonal blocks. The diagonal blocks are not required to have anything in common, they correspond to disjoint subsets of states for a system. By contrast, we say that a transition rate matrix is *nearly-independent* if it can be partitioned so that the diagonal contains two or more occurrences of (approximately) the same block, while the off-diagonal blocks must have (approximately) the structure in Figure 1, but they are not at all required to have small entries. Each diagonal block corresponds to the same set of “local states” (obtained by considering only some of the components in the description of the states of \mathbf{C}).

³A completely independent matrix is indeed just a particular case of strong lumpability [17]. Complete independence, just as complete decomposability, is quite uninteresting and of scarce relevance, since it practically never holds in real applications.

3.2 Near-independent CTMC structures

The degree of decomposability is given by how much the off-diagonal blocks differ from being zero, so it can be easily characterized. The characterization of the degree of independence is more complex, since it must measure the distance of each block (including the diagonal ones) from the corresponding “ideal” block, which, not only is not zero, but is not even uniquely defined. We then limit ourselves to a qualitative characterization of near-independence, by describing the three basic structures that can arise in the CTMC underlying a nearly-independent SRN.

Rate-dependence. Figure 2 contains three SRNs, \mathbf{C}^a , \mathbf{C}^b , and \mathbf{C}^c , each with two subnets, \mathbf{A} and \mathbf{B} , composed in different ways. In \mathbf{C}^a , \mathbf{A} and \mathbf{B} are structurally disconnected. If all rates are constant, \mathbf{A} and \mathbf{B} are also behaviorally disconnected, and we can say that \mathbf{C}^a is completely independent.

The simplest deviation from this complete independent structure arises when the pattern of zero and positive elements remains the same, but the values of some elements vary. This requires that at least one rate in one subnet be a function of the marking of another subnet. For example, if λ_{t_1} is a function of $\#(p_3)$, the corresponding $R^{\mathbf{C}^a}$ is in Figure 2, assuming that the order of the SRN markings is $((1010), (1001), (0110), (0101))$. The smaller the difference between $\lambda_{t_1}(1)$ and $\lambda_{t_1}(0)$ is, the closer $R^{\mathbf{C}^a}$ is to be independent.

Synchronization-dependence. If the pattern of zero and positive elements is not enforced in the CTMC, any positive element in the original CTMC may become zero, and vice versa. Consider the case of an off-diagonal zero element in an off-diagonal block becoming positive. In the corresponding SRN, such an entry describes the simultaneous change of marking in two or more subnets. Since the simultaneous firing of two or more timed transitions has probability zero, the change must be caused by the firing of a synchronizing transition such as t_5 in SRN \mathbf{C}^b , with inputs or output places in \mathbf{A} and \mathbf{B} . The smaller λ_{t_5} is with respect to λ_{t_1} and λ_{t_3} , the closer $R^{\mathbf{C}^b}$ is to be independent.

External-dependence. Another case to consider is that of a positive diagonal element on an off-diagonal block becoming zero. In the SRN, this must be caused by an inhibiting construct (such as the inhibitor arc from p_4 to t_1 in \mathbf{C}^c) that may inhibit the firing of one transition in a subnet (\mathbf{A}) depending on the marking of another subnet (\mathbf{B}). The quantification of the effect of this type of near-independence is more difficult than for the previous two. In \mathbf{C}^c , the smaller the value of λ_{t_1} is with respect to λ_{t_4} (the rate at which the inhibiting arc becomes irrelevant), the closer $R^{\mathbf{C}^c}$ is to be independent.

Multiple nearly-independent interactions. We have listed three interactions that can cause the transition rate matrix of a perfectly independent CTMC to lose its independence:

- Changing one of its positive entries to a different positive value.
- Changing one of its off-diagonal zero entries in an off-diagonal block to a positive value.
- Changing one of its diagonal positive entries in an off-diagonal block to zero.

All the other manifestations of near-independence are obtained from these three basic types of interactions:

- Changing an off-diagonal positive entry in a diagonal block to zero can be reduced to the third type of interaction by reordering the states, that is, considering $R^{\mathbf{B}} \oplus R^{\mathbf{A}}$ instead of $R^{\mathbf{A}} \oplus R^{\mathbf{B}}$.

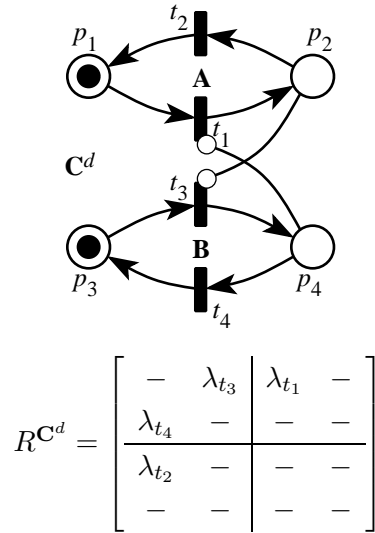


Figure 3: A nearly-independent SRN with defective state-space.

- Changing an off-diagonal zero entry in a diagonal block to a positive value can also be reduced to the third type of interaction, by considering an independent CTMC where that entry is positive in *all* the diagonal blocks, reordering the states as above, and then changing the entry back to zero in all but one block.

In many models, multiple interactions are present and it might be difficult to recognize them individually. In addition, the state-space itself may be incomplete. For example, consider SRN \mathbf{C}^d in Figure 3, obtained by adding an inhibitor arc from p_2 to t_3 in SRN \mathbf{C}^c of Figure 2. \mathbf{A} and \mathbf{B} now exhibit external-dependence on each other and the reachability set becomes a proper subset of the original one, since marking (0101) is not reachable. The projections of the new reachability set $\mathcal{T}^{\mathbf{C}^d}$ to (p_1, p_2) and to (p_3, p_4) , though, contain $(01 \cdot \cdot)$ and $(\cdot \cdot 01)$, respectively. The possibility of an approximate decomposition of \mathbf{C}^d into two SRNs \mathbf{A} and \mathbf{B} is not out of the question, but trying to obtain measures relating the markings of \mathbf{A} and \mathbf{B} is difficult. For example, $Pr\{\#\mathbf{A}(p_2) = 1\} > 0$, $Pr\{\#\mathbf{B}(p_4) = 1\} > 0$, but $Pr\{\#\mathbf{C}^d(p_2) = 1 \wedge \#\mathbf{C}^d(p_4) = 1\} = 0$, since no marking satisfies this condition in $\mathcal{T}^{\mathbf{C}^d}$.

4 SRN decomposition

In general, the following steps are needed to exploit near-independence at the SRN level:

1. **Decomposition.** Given an SRN \mathbf{A} , generate SRNs $\mathbf{A}_1, \dots, \mathbf{A}_k$. Often, \mathbf{A}_i is not strictly a subnet of \mathbf{A} , but it shares a common subnet with it. These SRNs are parametric because the firing rates of some of their transitions might be expressed as a function of real parameters to be specified.
2. **Import graph.** For each SRN \mathbf{A}_i , fix the value of its parameters using *imports* from \mathbf{A}_j , $1 \leq j \leq k$. Also, specify the measures exported from \mathbf{A}_i after its solution (see Section 4.1). If \mathbf{A}_i imports a measure from \mathbf{A}_j , we write $\mathbf{A}_j \succ \mathbf{A}_i$. The graph describing the import relation may

be cyclic and the case $\mathbf{A}_i \succ \mathbf{A}_i$, a cycle of length one, may occur. We denote the transitive closure of the import relation with the symbol \succ .

3. **Iteration.** If the import graph is acyclic, it implicitly defines a (partial) order for the solution of the SRNs \mathbf{A}_i . If $\mathbf{A}_i \succ \mathbf{A}_j$, \mathbf{A}_i must be studied before \mathbf{A}_j . If neither $\mathbf{A}_i \succ \mathbf{A}_j$ nor $\mathbf{A}_j \succ \mathbf{A}_i$, then \mathbf{A}_i and \mathbf{A}_j can be studied in any order. If \mathbf{A}_i and \mathbf{A}_j belong to a cycle, one of them must be chosen to be studied first, but its imports are not available and initial guesses must be provided for them. After each \mathbf{A}_i has been studied once, more iterations can be performed, each time using the most recent value for the imports. Convergence is reached when the relative change between successive iterations in all the imports is below a given tolerance.

4.1 Primitives for the exchange of data

The values exchanged between two SRN models can represent a wide range of model-specific information, but their computation can be expressed using only three primitive types of SRN measures.

Given an SRN \mathbf{A} , we can define a condition c as a boolean marking-dependent expression; in each marking, c is either *ON* (holds) or *OFF* (does not hold). The following quantities can then be defined:

$$\begin{aligned}\Lambda_{\mathbf{A}}(c)^{-1} &= E[\text{time } c \text{ remains } ON \text{ in steady-state}] \\ Pr_{\mathbf{A}}\{c\} &= Pr\{c \text{ is } ON \text{ in steady-state}\} \\ \Delta_{\mathbf{A}}(c)^{-1} &= E[\text{time until } c \text{ is } ON \text{ in steady-state}]\end{aligned}$$

(we can eliminate the subscript \mathbf{A} if the SRN is clear from the context). $\Lambda(c)$, $Pr\{c\}$, and $\Delta(c)$ can be effectively computed in practice using numerical methods. Define \mathcal{T}_c and $\mathcal{T}_{\bar{c}}$ to be the partition of \mathcal{T} according to whether the condition c is respectively *ON* or *OFF*. If $\underline{\pi}$ is the steady-state probability vector for the underlying CTMC,

$$\Lambda(c)^{-1} = \frac{\sum_{k \in \mathcal{T}_c} \pi_k}{\sum_{i \in \mathcal{T}_c, j \in \mathcal{T}_{\bar{c}}} \pi_i Q_{i,j}}$$

That is, $\Lambda(c)$ is the rate at which the condition goes *OFF* given that it is *ON*, in steady-state. This quantity is a positive finite quantity if the SRN is ergodic and neither \mathcal{T}_c nor $\mathcal{T}_{\bar{c}}$ are empty.

$$Pr\{c\} = \sum_{k \in \mathcal{T}_c} \pi_k$$

$Pr\{c\}$ is simply the sum of the steady-state probability of each marking in \mathcal{T}_c .

$\Delta(c)^{-1}$ is the expected time to absorption when the markings in \mathcal{T}_c are considered absorbing, starting from steady-state (this includes taking into account the probability that c is *ON* in steady-state):

$$\Delta(c)^{-1} = \sum_{k \in \mathcal{T}_{\bar{c}}} x_k^{\bar{c}} \quad \text{where} \quad \underline{x}^{\bar{c}} Q^{\bar{c}} = -\underline{\pi}^{\bar{c}}$$

$x^{\bar{c}}$ is the expected time spent in each non-absorbing tangible marking before absorption, while $Q^{\bar{c}}$ and $\underline{\pi}^{\bar{c}}$ are the restriction of Q and $\underline{\pi}$ to the states in $\mathcal{T}_{\bar{c}}$.

Both $\Lambda(c)$ and $\Delta(\bar{c})$ refer to sojourns in markings where c holds, but they have different values. For example, consider the CTMC in Figure 4, where the sojourn time in state i has exponential

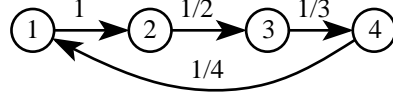


Figure 4: Example for the computation of Λ and Δ .

distribution with parameter i^{-1} . The steady-state probability vector is $\underline{\pi} = [0.1, 0.2, 0.3, 0.4]$. If $\mathcal{T}_c = \{1, 2\}$ and $\mathcal{T}_{\bar{c}} = \{3, 4\}$,

$$Q^{\bar{c}} = \left[\begin{array}{c|c} -\frac{1}{3} & \frac{1}{3} \\ \hline 0 & -\frac{1}{4} \end{array} \right], \quad \underline{\pi}^{\bar{c}} = [0.3, 0.4], \quad \underline{x}^{\bar{c}} = [0.9, 2.8]$$

$$Q^c = \left[\begin{array}{c|c} -1 & 1 \\ \hline 0 & -\frac{1}{2} \end{array} \right], \quad \underline{\pi}^c = [0.1, 0.2], \quad \underline{x}^c = [0.1, 0.6]$$

Then,

$$\Lambda(c)^{-1} = \frac{0.1 + 0.2}{0.2\frac{1}{2}} = 3 \neq \Delta(\bar{c})^{-1} = 0.1 + 0.6 = 0.7$$

$$\Lambda(\bar{c})^{-1} = \frac{0.3 + 0.4}{0.4\frac{1}{4}} = 7 \neq \Delta(c)^{-1} = 0.9 + 2.8 = 3.7$$

If we know that condition b holds in the set of markings we are interested in, but nothing else about the probability of these markings, we can use a modified version of the steady state vector, $\underline{\pi}^b$, defined as

$$\forall i \in S, \pi_i^b = \begin{cases} \frac{\pi_i}{\sum_{j \in \mathcal{T}_b} \pi_j} & \text{if } i \in \mathcal{T}_b \\ 0 & \text{otherwise} \end{cases}$$

The quantities $\Lambda(c)$, $Pr\{c\}$, and $\Delta(c)$ can then be generalized to $\Lambda(c|b)$, $Pr\{c|b\}$, and $\Delta(c|b)$.

$$\Lambda(c|b)^{-1} = \frac{\sum_{k \in \mathcal{T}_c} \pi_k^b}{\sum_{i \in \mathcal{T}_c, j \in \mathcal{T}_{\bar{c}}} \pi_i^b Q_{i,j}}$$

The denominator in the equation for $\Lambda(c|b)^{-1}$ is zero if conditions b and c are incompatible or, even if they are compatible, if no direct transition is possible from any marking where both b and c hold to a marking where c does not hold. We then define $\Lambda(c|b)$ to be zero, regardless of the value of the numerator, since this corresponds to a condition which cannot hold.

$$Pr\{c|b\} = \sum_{k \in \mathcal{T}_c} \pi_k^b$$

$$\Delta(c|b)^{-1} = \sum_{k \in \mathcal{T}_{\bar{c}}} x_k^{\bar{c}|b} \quad \text{where} \quad \underline{x}^{\bar{c}|b} Q^{\bar{c}} = -\underline{\pi}^{\bar{c}|b}$$

and $\underline{\pi}^{\bar{c}|b}$ is the restriction of $\underline{\pi}^b$ to the states in $\mathcal{T}_{\bar{c}}$.

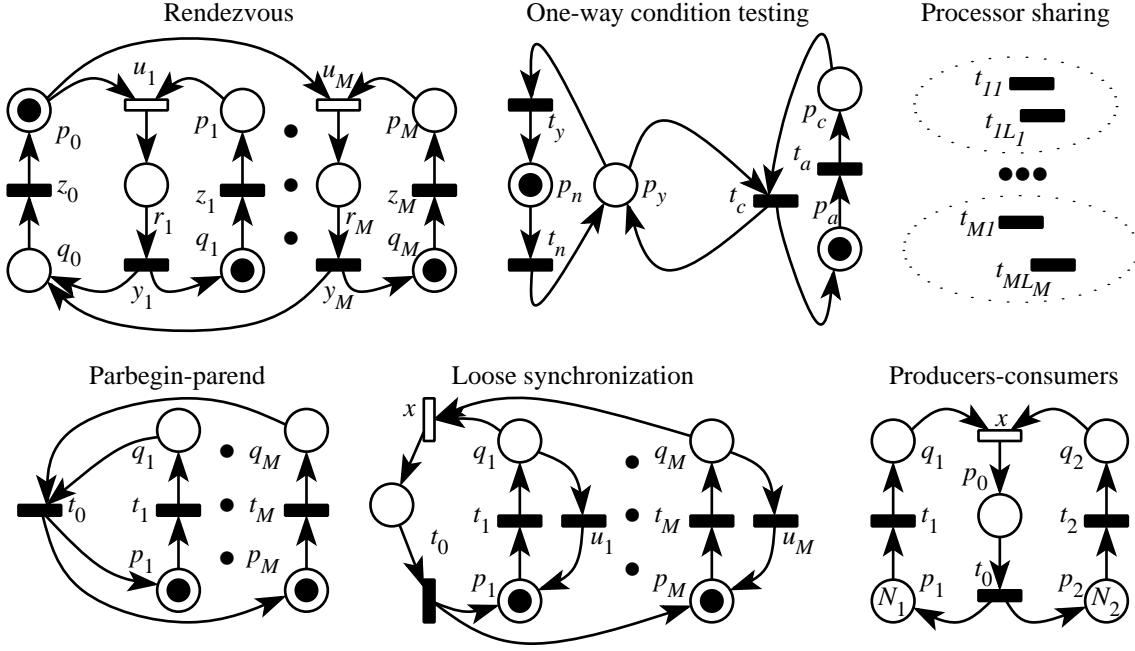


Figure 5: Some nearly-independent SRN structures.

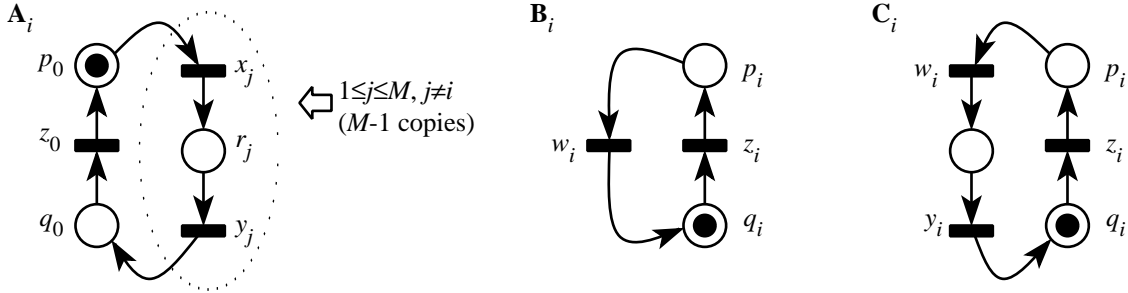


Figure 6: The decomposition of the rendezvous SRN.

$Pr\{c\}$ can be derived from $\Lambda(c)$ and $\Lambda(\bar{c})$. First, observe that the conservation of flow ensures that rate at which \mathcal{T}_c and $\mathcal{T}_{\bar{c}}$ are entered is the same in steady-state, so we can write

$$\sum_{i \in \mathcal{T}_c, j \in \mathcal{T}_{\bar{c}}} \pi_i Q_{i,j} = \sum_{i \in \mathcal{T}_{\bar{c}}, j \in \mathcal{T}_c} \pi_i Q_{i,j} = Q_c$$

Then, $\Lambda(c) = Q_c / Pr\{c\}$ and $\Lambda(\bar{c}) = Q_c / Pr\{\bar{c}\}$ and we can compute $Pr\{c\}$ as

$$\frac{\Lambda(\bar{c})}{\Lambda(c) + \Lambda(\bar{c})} = \frac{Q_c / Pr\{\bar{c}\}}{Q_c / Pr\{c\} + Q_c / Pr\{\bar{c}\}} = Pr\{c\}$$

The same is not true for the conditional version of these quantities: in general, $Pr\{c|b\}$ cannot be derived from $\Lambda(c|b)$ and $\Lambda(\bar{c}|b)$.

4.2 Near-independent SRN structures

Figure 5 contains some nearly-independent SRN structures which can be decomposed using our approach. Unfilled rectangles represent transitions having $\lambda = \infty$ in any marking. In GSPN terminology, they are “immediate transitions”. The other transitions are general SRN transitions. Furthermore, they could have an arbitrary phase-type distribution with marking-dependent parameters instead of a simple exponential distribution. This corresponds to substituting a transition with an entire subnet. The decomposition approach still applies, but the quality of the approximation differs. See [10, 12] for further details.

We will consider the decomposition of one of them, modeling a rendezvous, in detail and only sketch the approach for the remaining ones.

Rendezvous. The rendezvous structure shown in Figure 5 derives its name from the interaction between Ada tasks, which it models closely. The same structure, though, arises when modeling contention for hardware or software resources which must be used in exclusive mode as opposed to processor-sharing.

The token in p_0 represents a callee task X_0 . The tokens in q_i represent M caller tasks X_i ($1 \leq i \leq M$). A token in place r_i signifies that the callee is in rendezvous with caller X_i and the portion of code represented by y_i corresponds to the operations performed during the rendezvous.

We can study each X_i ($1 \leq i \leq M$) individually by estimating the amount of time each of them waits for a rendezvous (in p_i), but the study of X_0 is more complex. Our decomposition makes use of the SRNs in Figure 6, where the value of λ_{x_i} and λ_{w_i} must be determined. When caller X_i is ready to rendezvous, it deposits a token in p_i . The expected amount of time before X_0 accepts the call must then be estimated. SRN \mathbf{A}_i is used for this purpose, it represents the possible states of X_0 when the token arrives in p_i . X_0 may be executing locally, in q_0 ; it may be in rendezvous with X_j ($1 \leq j \leq M, j \neq i$), in r_j ; or it may be waiting for a call, in p_0 . The value of $\lambda_{x_j}^{-1}$ represents the expected time elapsing from the arrival of a token in p_0 to the arrival of a token in p_j , when X_i is executing locally: it is estimated from \mathbf{B}_j as

$$\Delta_{\mathbf{B}_j}(\#(p_j) = 1)$$

From \mathbf{A}_i , we can compute

$$\Delta_{\mathbf{A}_j}(\#(p_0) = 1)$$

representing only the time until a token arrives in p_0 when X_i is waiting in p_i ; once the token arrives in p_0 , other producers may be waiting for it in addition to X_i , so the waiting of X_i may not be over; X_0 may decide to rendezvous with other producers before satisfying X_i . If M is sufficiently large, we can assume that N , the number of times X_i is not chosen, has a distribution $\sim MODGEOM(\alpha)$ where

$$\alpha = \frac{1}{1 + \sum_{j=1, j \neq i}^M Pr_{\mathbf{C}_j}\{\#(p_j) = 1\}}$$

since X_i is waiting with probability one while producer X_j ($j \neq i$) is waiting with probability $Pr_{\mathbf{C}_j}\{\#(p_j) = 1\}$, if we assume it is in steady-state. \mathbf{B}_i and \mathbf{C}_i need to know the expected to-

tal time before X_i can rendezvous, $\lambda_{w_i}^{-1}$, computed as

$$\begin{aligned} \lambda_{w_i}^{-1} &= \underbrace{\Delta_{\mathbf{A}_i}(\#(p_0) = 1)}_{E[\text{time before } \# \mathbf{A}_i(p_0)=1]} + \underbrace{\frac{1 - \alpha}{\alpha}}_{E[N]} \cdot \frac{\sum_{j=1, j \neq i}^M Pr_{\mathbf{C}_j} \{\#(p_j) = 1\} (\lambda_{y_j}^{-1} + \lambda_{z_0}^{-1})}{\underbrace{\sum_{j=1, j \neq i}^M Pr_{\mathbf{C}_j} \{\#(p_j) = 1\}}_{E[\text{length of a rendezvous with a task other than } X_i]}} \\ &= \Delta_{\mathbf{A}_i}(\#(p_0) = 1) + \sum_{j=1, j \neq i}^M Pr_{\mathbf{C}_j} \{\#(p_j) = 1\} (\lambda_{y_j}^{-1} + \lambda_{z_0}^{-1}) \end{aligned}$$

We use $Pr_{\mathbf{C}_j} \{\#(p_j) = 1\}$, the probability of X_j being waiting in \mathbf{C}_j , not in \mathbf{B}_j . The difference between the two is the presence of y_j , corresponding to the rendezvous of X_j with X_0 . We use \mathbf{B}_j to obtain $\Delta_{\mathbf{B}_j}(\#(p_j) = 1)$, needed by \mathbf{A}_i , because X_j cannot be in rendezvous with X_0 when a token is in p_0 ; now the rendezvous is possible, so we use \mathbf{C}_j .

These three sets of SRNs constitute the entire decomposition of the rendezvous SRN. Iteration is needed because

$$\begin{aligned} \forall i, 1 \leq i \leq M, \mathbf{A}_i \succ \mathbf{B}_i \\ \forall i, 1 \leq i \leq M, \mathbf{A}_i \succ \mathbf{C}_i \\ \forall i, 1 \leq i \leq M, \forall j, 1 \leq j \leq M, j \neq i, \mathbf{C}_j \succ \mathbf{C}_i \\ \forall i, 1 \leq i \leq M, \forall j, 1 \leq j \leq M, j \neq i, \mathbf{C}_j \succ \mathbf{B}_i \\ \forall i, 1 \leq i \leq M, \forall j, 1 \leq j \leq M, j \neq i, \mathbf{B}_j \succ \mathbf{A}_i \end{aligned}$$

One-way condition testing. The token residing in places p_y or p_n represents a condition that can be either true (token in p_y) or false (token in p_n). The left side of the SRN describes the change of the condition, and is completely independent of the right side, which must occasionally test the condition before it can continue.

For example, the left and right sides could represent a high and a low priority job, respectively, sharing a resource. The condition is then “high priority job is not using the CPU” and transitions t_n and t_c represent the use of the resource, while transitions t_y and t_a represent periods during which the shared resource is not needed.

The decomposition uses two SRN. \mathbf{A}_l contains places p_y and p_n and transitions t_y and t_n . \mathbf{A}_r contains places p_a and p_c and transitions t_a and t_c . \mathbf{A}_r must import from \mathbf{A}_l the quantities $Pr_{\mathbf{A}_l} \{\#(p_y) = 1\}$ and $\Delta_{\mathbf{A}_l}(\#(p_y) = 1)$. \mathbf{A}_l is unaffected by \mathbf{A}_r , so it has no imports. This is an example of external-dependence.

Processor-sharing. A SRN composed of M structurally disconnected (sub)SRNs with transitions sharing resources in processor-sharing mode is one of the most intuitive cases of near-independence. In SRN \mathbf{A}_i , transitions t_{i1}, \dots, t_{iL_i} require to use a resource whenever they are enabled. If K transitions needing the resource are enabled in a marking, each of them receives a fraction $1/K$ of it (their actual firing rates are decreased by a factor K).

Several extensions are possible. Each enabled transition t_{ij} could require r_{ij} resources (r_{ij} could be equal to the number of tokens in an input place, representing customers waiting for concurrent service at t_{ij}). R equivalent copies of the resource could exist. In this case, t_{ij} is granted a fraction $r_{ij} \min\{1, R/\sum_{k,l} r_{kl}\}$. The allocation of resources to requests could even follow a more complex algorithm, as long as no transition is completely disabled because of resource contention.

In any case, this is an example of rate-dependence. Each SRN \mathbf{A}_i must know from each SRN \mathbf{A}_k , $k \neq i$, information about its resource usage, such as the expected number of requests in steady state or, even better, its distribution:

$$\forall k, 1 \leq k \leq M, k \neq i, \forall r, 0 \leq r \leq T_k, Pr \left\{ \sum_{1 \leq l \leq L_k} r_{kl} = r \right\}$$

where T_k is the maximum number of requests to the resource that can be originated by the transitions of SRN k in any marking.

It is also possible to have several types of resources, as long as each transition issues requests to at most one type of resource in each marking.

Tight synchronization (parbegin-parend). Assume we have an infinite loop containing a sequential thread followed by M concurrent threads. Each thread from p_i to q_i can be studied in isolation, to compute $\lambda_{t_i}^{-1} = \Delta(\#(q_i) = 1 | \#(p_i) = 1)^{-1}$, the mean time to absorption (MTTA) for thread i starting from the marking(s) where $\#(p_i) = 1$ (remember that transition t_i can represent a complex subnet). Then, the original SRN can be studied assuming that the firing time for t_i is exponentially distributed with parameter λ_{t_i} . The cycle time C for the token is then computed as the inverse of the throughput of t_0 . Finally, each thread i is studied again, this time in steady state, after adding a transition w_i with input q_i , output p_i , and rate $(C - 1/\lambda_{t_i})^{-1}$, since $C - 1/\lambda_{t_i}$ represents the average amount of time thread i is not in use.

This decomposition still requires the generation of large reachability graphs, since the original SRN where all the transitions t_i have exponential distributions generates $2^M + 1$ markings. Large savings are obtained, though, if the transitions correspond to complex subnets. The last step where each individual thread is solved again with the addition of transition w_i is particularly useful when the threads have an internal state. The subnet representing thread i could in fact have internal activities which are nearly-independent of the main execution thread. In this case, the initial marking for the subnet might affect the computation of λ_{t_i} , so iterations would be needed.

Loose synchronization. The structure of a loose-synchronization SRN differs from that of the parbegin-parend in one important aspect. In both SRNs, when concurrent thread i ($1 \leq i \leq M$) completes the execution of t_i , a token is put in q_i . With tight synchronization, though, the token can only wait in q_i for the firing of t_0 . With loose synchronization, instead, transition u_i can remove the token from q_i . In other words, the thread can “give up waiting” and execute for some more time, before attempting to synchronize again. Synchronization only happens when all the M threads happen to be waiting at the same time.

The decomposition of this structure is somewhat similar to that of the tight synchronization, although the imports are different (this time, $Pr\{\#(q_i) = 1\}$ is of interest, not $\Delta(\#(q_i) = 1 | \#(p_i) = 1)$).

Intuitively, the individual threads are much less independent with tight synchronization than with loose synchronization. The throughputs of the timed transitions, $\tau(t_i)$ ($0 \leq i \leq M$) are the same in the first case, while they only satisfy the constraint $\tau(t_0) \leq \min_{1 \leq i \leq M} \{\tau(t_i)\}$ in the second case.

The SRN \mathbf{A} in Figure 7 models a flexible manufacturing system (FMS) with three machines: M_1 , M_2 , and M_3 . Machine M_1 is triplicated and processes parts of type $P1$, up to three at a time. Machine M_2 normally processes parts of type $P2$, one at a time, but it can also process parts of type $P3$, when no part of type $P2$ is ready to be processed.

Finished parts of type $P1$ and $P2$ may also be assembled together (one part of each type) into a new part type, $P12$; machine M_3 , duplicated, is used for this purpose. When finished parts of type $P1$, $P2$, $P3$, and $P12$ are shipped, the same number of rough parts enters the system, to maintain a constant inventory. The time required to ship and replace parts is constant and independent of the number of parts involved. The “flushing” arcs connected to the corresponding transitions, t_{P1s} , t_{P2s} , t_{P3s} , and t_{P12s} have a marking-dependent multiplicity equal to the number of tokens in the input place for the transition. For example, when t_{P12s} fires, $\#(P12s)$ tokens are removed from place $P12s$, which becomes empty, and $\#(P12s)$ tokens are added to both place $P1$ and $P2$.

Rough parts of any type are moved one at a time to the appropriate machine using pallets. Finished parts of type $P1$ and $P2$ to be assembled by M_3 also need to be moved, but two, one of each type, share one pallet.

The total number of parts of type $P1$, $P2$, and $P3$ in the system is N_1 , N_2 , and N_3 , respectively. The total number of pallets is $N_p = \lfloor (N_1 + N_2 + N_3)/2 \rfloor$.

The meaning of each place and transition is explained in Table 1. Table 2 gives the transition rates or weights. The pallets are a shared resource; for the purpose of illustration, we assume that the contention for them can be approximated using the processor-sharing policy, although we realize that this might be unrealistic, especially with a small number of pallets. In Table 2, the quantity $r = \#(P1) + \#(P2) + \#(P3) + \#(P12)$ represents the number of pallets requested in any given marking. Transitions t_{P1} , t_{P2} , t_{P3} , and t_{P12} are drawn with a grey pattern to indicate that they share a resource (the pallets).

We study a single measure, ψ , the “productivity” of the FMS:

$$\psi = 400\phi_1 + 600\phi_2 + 100\phi_3 + 1100\phi_{12}$$

where ϕ_x , $x \in \{1, 2, 3, 12\}$ is the throughput (in min^{-1}) for parts of type x and the multiplicative constants are the net gain when producing a part of the corresponding type.

5.1 Decomposition of the model

\mathbf{A} can be studied only for small values of N_1 , N_2 , and N_3 , since its reachability set soon becomes excessively large. We then decompose \mathbf{A} into three SRNs: \mathbf{A}_1 , \mathbf{A}_2 , and \mathbf{A}_3 (Figure 8). \mathbf{A}_1 describes the flow of $P1$ parts, including their assemblage with $P2$ parts, \mathbf{A}_2 describes the flow of $P2$ parts, including their assemblage with $P1$ parts, and \mathbf{A}_3 describes the flow of $P3$ parts. Places $P2$ in \mathbf{A}_1 and $P1$ in \mathbf{A}_2 are kept to enforce a limit of $N_{12} = \min\{N_1, N_2\}$ on the total number of tokens in the subnet assembling $P1$ and $P2$ parts.

The reachability sets of the three SRNs are much smaller than the one for \mathbf{A} , so it is possible to study the behavior of the FMS for larger values of N_1 , N_2 , and N_3 .

\mathbf{A}_1 , \mathbf{A}_2 , and \mathbf{A}_3 interact in several ways. They share the common pool of pallets (processor-sharing

interaction); the rates of t_{P1} , t_{P2} , t_{P3} , and t_{P12} are:

$$\begin{aligned}\lambda_{t_{P1}}^{\mathbf{A}_1} &= \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} u_2[i_2, N_2 - \#(P2)] u_3[i_3] \#(P1) \min \{1, N_p / (\#(P1) + \#(P12) + i_2 + i_3)\} \\ \lambda_{t_{P12}}^{\mathbf{A}_1} &= \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} u_2[i_2, N_2 - \#(P2)] u_3[i_3] \#(P12) \min \{1, N_p / (\#(P1) + \#(P12) + i_2 + i_3)\} \\ \lambda_{t_{P2}}^{\mathbf{A}_2} &= \sum_{i_1=0}^{N_1} \sum_{i_3=0}^{N_3} u_1[i_1, N_1 - \#(P1)] u_3[i_3] \#(P2) \min \{1, N_p / (\#(P2) + \#(P12) + i_1 + i_3)\} \\ \lambda_{t_{P2}}^{\mathbf{A}_2} &= \sum_{i_1=0}^{N_1} \sum_{i_3=0}^{N_3} u_1[i_1, N_1 - \#(P1)] u_3[i_3] \#(P12) \min \{1, N_p / (\#(P2) + \#(P12) + i_1 + i_3)\} \\ \lambda_{t_{P3}}^{\mathbf{A}_3} &= \sum_{j=0}^{N_{12}} \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} u_{12}[j] u_1[i_1, j] u_2[i_2, j] \#(P3) \min \{1, N_p / (\#(P3) + j + i_1 + i_2)\}\end{aligned}$$

where $\forall i_1, 1 \leq i_1 \leq N_1, \forall i_2, 1 \leq i_2 \leq N_2, \forall i_3, 1 \leq i_3 \leq N_3, \forall j, 0 \leq j \leq N_{12}$,

$$\begin{aligned}u_1[i_1, j] &= Pr_{\mathbf{A}_1} \{\#(P1) = i_1 | N_2 - \#(P2) = j\} \\ u_2[i_2, j] &= Pr_{\mathbf{A}_2} \{\#(P2) = i_2 | N_1 - \#(P1) = j\} \\ u_3[i_3] &= Pr_{\mathbf{A}_3} \{\#(P3) = i_3\} \\ u_{12}[j] &= (Pr_{\mathbf{A}_1} \{\#(P12) = j\} + Pr_{\mathbf{A}_2} \{\#(P12) = j\}) / 2\end{aligned}$$

The last quantity represents the pmf of the number of tokens in $P12$. Both \mathbf{A}_1 and \mathbf{A}_2 contain this place, so this information is needed only by \mathbf{A}_3 . Since the estimates for this pmf obtained from \mathbf{A}_1 and \mathbf{A}_2 might differ, we average them.

The expressions $N_2 - \#(P2)$ and $N_1 - \#(P1)$ in the above equations refer to the total number of tokens in places $P12$, $P12wM3$, $P12M3$, and $P12s$ as seen by \mathbf{A}_1 and \mathbf{A}_2 , respectively. Since they represent the same quantity, they are equated when the two SRNs exchange data. So, for example, the rate for transition t_{P1} in \mathbf{A}_1 when $N_2 - \#(P2) = j$ is computed using the pmf of the number of pallets used by t_{P2} when $N_1 - \#(P1) = j$ in \mathbf{A}_2 .

\mathbf{A}_1 and \mathbf{A}_2 contain parts that must occasionally wait for each other to be assembled (producers-consumers interaction); t_x then becomes a timed transition in \mathbf{A}_1 and \mathbf{A}_2 , with rate:

$$\lambda_{t_x}^{\mathbf{A}_1} = d_2[N_2 - \#(P2)] \quad (1)$$

$$\lambda_{t_x}^{\mathbf{A}_2} = d_1[N_1 - \#(P1)] \quad (2)$$

where $\forall j, 0 \leq j \leq N_{12}$,

$$\begin{aligned}d_1[j] &= \Delta_{\mathbf{A}_1} (\#(P1wP2) > 0 | N_2 - \#(P2) = j) \\ d_2[j] &= \Delta_{\mathbf{A}_2} (\#(P2wP1) > 0 | N_1 - \#(P1) = j)\end{aligned}$$

That is, $d_2[j]^{-1}$ is computed as the expected time until $P2wP1$ becomes not-empty in \mathbf{A}_2 starting from steady-state, given that $N_1 - j$ tokens are in $P1$, that is, given that j tokens are in $P12$, $P12wM3$, $P12M3$, and $P12s$. Then, the rate of t_x in \mathbf{A}_1 in a marking where $N_2 - \#(P2) = j$ is simply equated to the inverse of this time.

Finally, parts $P3$ in \mathbf{A}_3 need M_2 , but M_2 can process them only when it has no requests from \mathbf{A}_2 (one-way condition testing interaction); the rate of transition t_{P3M2} is then:

$$\lambda_{t_{P3M2}}^{\mathbf{A}_3} = \left(\frac{2}{Pr_{\mathbf{A}_2}\{\#(P2M2) = 0\}} + \frac{1}{\Delta_{\mathbf{A}_2}(\#(P2M2) = 0)} \right)^{-1}$$

That is, the rate of t_{P3M2} is the inverse of the sum of two times. The first time is obtained dividing the time that would be required in isolation to machine a part $P3$, 2, by a “rarefaction” quantity equal to the probability of finding machine M_2 not in use by $P2$ parts (shadow CPU approximation [25]). The second time is a “delay” portion equal to the expected time before M_2 can become available to $P3$ parts, starting from steady-state. See [10] for a detailed discussion of the approximations obtained by taking into account only the rarefaction, only the delay, or both, which is the approach we use in this example, since it usually results in the best approximation.

We observe that the above rates, with the exception of $\lambda_{t_{P3M2}}^{\mathbf{A}_3}$, are marking-dependent. For example, the rates of t_{P1} and t_{P12} in \mathbf{A}_1 depend on the number of tokens in places $P2$, $P1$, and $P12$ of \mathbf{A}_1 , the rates of t_{P2} and t_{P12} in \mathbf{A}_2 depend on the number of tokens in places $P1$, $P2$, and $P12$ of \mathbf{A}_2 , and so on.

The analysis of \mathbf{A}_1 , \mathbf{A}_2 , and \mathbf{A}_3 using our decomposition schema gives estimates for ϕ_1 , ϕ_2 , ϕ_3 , and ϕ_{12} . The estimates of ϕ_{12} obtained from \mathbf{A}_1 and \mathbf{A}_2 , $\phi_{12}^{(1)}$ and $\phi_{12}^{(2)}$ do not coincide. We could simply average them, as we did for the computation of u_{12} , but we can use a better heuristic.

In the exact model, the throughput ϕ_{12} is mostly determined by the “bottleneck” subnet, the slowest subnet in producing tokens for the synchronization in t_x . In general, we would like to characterize the “level of credibility” of $\phi_{12}^{(1)}$ and $\phi_{12}^{(2)}$. For example, if parts $P1$ are the bottleneck, $P1wP2$ is almost always empty because $P2wP1$ almost always contains tokens waiting to synchronize. Hence, both $\alpha_2 = Pr_{\mathbf{A}_2}\{\#(P2wP1) > 0\}$ and $(1 - \alpha_1) = Pr_{\mathbf{A}_1}\{\#(P1wP2) = 0\}$ measure the credibility of $\phi_{12}^{(1)}$, while $\alpha_1 = Pr_{\mathbf{A}_1}\{\#(P1wP2) > 0\}$ and $(1 - \alpha_2) = Pr_{\mathbf{A}_2}\{\#(P2wP1) = 0\}$ measure the credibility of $\phi_{12}^{(2)}$.

We must then determine our “best estimate” for the value of ϕ_{12} , as a function of $\phi_{12}^{(1)}$, $\phi_{12}^{(2)}$, α_1 , and α_2 . Out of many possibilities, the most natural is the weighted average

$$\phi_{12} = \frac{\alpha_2 \phi_{12}^{(1)} + \alpha_1 \phi_{12}^{(2)}}{\alpha_1 + \alpha_2}$$

The comparison of the exact and approximate results (possible for values of $N = N_1 = N_2 = N_3 \leq 5$) suggests that, in this model, our approximation computes a lower bound of the exact value (Table 3). We define the percent error as $100(\psi_{approx} - \psi_{exact})/\psi_{exact}$. The exact and approximate values for the productivity ψ as a function of $N = N_1 = N_2 = N_3$ are plotted in Figure 9.

Table 4 shows the complexity of the analysis for the exact and the approximate solution. $|\mathcal{T}^{\mathbf{A}}|$, $\eta^{\mathbf{A}}$, and $n^{\mathbf{A}}$ represent the number of tangible markings, nonzero CTMC entries (excluding the diagonal), and iterations for the numerical solution of the exact model. The optimal Successive Over-Relaxation (SOR) method is used in all cases [14, 11]. Column “Total_e” is equal to $\eta^{\mathbf{A}} n^{\mathbf{A}}$ and represents the total number of floating point operations. The large size of $\mathcal{T}^{\mathbf{A}}$ and $\eta^{\mathbf{A}}$ indicates how the number of integer operations and the memory requirements are extremely large as well. For the approximate results, K is the number of decomposition-level iteration, that is, the number of times \mathbf{A}_1 , \mathbf{A}_2 , and \mathbf{A}_3 are solved (we choose to stop the decomposition-level iterations when the imports remain stable between

iterations, up to four significant digits). Column “Total_a” is equal to $(\eta^{\mathbf{A}_1}n^{\mathbf{A}_1} + \eta^{\mathbf{A}_2}n^{\mathbf{A}_2} + \eta^{\mathbf{A}_3}n^{\mathbf{A}_3})K$. The entries $n^{\mathbf{A}_1}$, $n^{\mathbf{A}_2}$, and $n^{\mathbf{A}_3}$ refer to the last decomposition-level iteration. Column $|\mathcal{T}^{\mathbf{A}_2}|$ is missing because it is the same as $|\mathcal{T}^{\mathbf{A}_1}|$. The execution time and memory requirements savings are substantial.

6 Conclusion

The decomposition approach that we discussed applies to a large class of SRNs and appears to be extendible to other SRN structures. We stress that our decomposition approach iteratively modifies the CTMC itself; furthermore, only three primitives are needed to define the imports: $\Lambda(c|b)$, $Pr\{c|b\}$, and $\Delta(c|b)$.

Two important issues remain open: the computation of bounds and a general theory on the convergence behavior. Empirically, we have found that the approximation is generally acceptable, at times extremely good, and that the convergence is reached in just a few iterations.

The reduction of the state space when applying near-independence is substantial. If K iterations are needed at the decomposition level, if M SRNs need to be analyzed at each iteration, and if, at the k -th decomposition-level iteration ($1 \leq k \leq K$), SRN \mathbf{A}_m ($1 \leq m \leq M$) has an underlying infinitesimal generator $Q_m^{(k)}$ with $\eta_m^{(k)}$ nonzero entries requiring $n_m^{(k)}$ iterations of the numerical method for its solution, the total cost is $\sum_{m=1}^M \sum_{k=1}^K \eta_m^{(k)} \cdot n_m^{(k)}$. Two observations can be made at this point. The SRNs

\mathbf{A}_m are often very simple. If they correspond to a product form queueing network or to a combinatorial model, for example, there is no need to study them with standard SRN solution methods. Sometimes the SRN can be captured by an equation, where, for each value of the import(s), a value for the export is obtained.

Even when the generation and the solution of a CTMC is required, the efficiency of the analysis might still be improved. The reachability graph for \mathbf{A}_m must be generated only once, since it does not change from k to $k+1$. Even the differences between $Q_m^{(k)}$ and $Q_m^{(k+1)}$ are usually minor, restricted to a small change in some of the rates; in particular, the pattern of zero and nonzero entries does not change, so $\eta_m^{(k)}$ is the same for all k . It is possible to exploit this property in two ways. First, $Q_m^{(\cdot)}$ should be generated once, leaving “symbolic” entries in correspondence to the imports for \mathbf{A}_m . Then, at step k , $Q_m^{(k)}$ could be generated from $Q_m^{(\cdot)}$ by binding the symbolic entries using the current values of the imports.

Furthermore, it is arguable that $\underline{\pi}_m^{(k)}$, the numerical steady-state solution of $\underline{\pi}_m^{(k)}Q_m^{(k)} = 0$, is only slightly different from $\underline{\pi}_m^{(k+1)}$, the solution of $\underline{\pi}_m^{(k+1)}Q_m^{(k+1)} = 0$, since $Q_m^{(k)}$ is close to $Q_m^{(k+1)}$. If we use $\underline{\pi}_m^{(k)}$ as the initial iterate when solving $\underline{\pi}_m^{(k+1)}Q_m^{(k+1)} = 0$, few iterations are needed, especially when convergence at the decomposition level is nearly achieved. While the first improvement requires the ability to store the transition rate matrix in symbolic format, the second improvement only requires storing a set of M probability vectors, from the k -th to the $(k+1)$ -th decomposition-level iteration.

Appendix

We briefly recall the definition of Kronecker sum and Kronecker product. More information on Kronecker algebra can be found in [5, 2].

Given two matrices E , $e_r \times e_c$, and F , $f_r \times f_c$, the Kronecker product of E and F is given by the $e_r f_r \times e_c f_c$ matrix

$$E \otimes F = \begin{bmatrix} E_{1,1}F & \cdots & E_{1,e_c}F \\ \cdots & \cdots & \cdots \\ E_{e_r,1}F & \cdots & E_{e_r,e_c}F \end{bmatrix}$$

Assuming that the row and column indexes of the matrices start at zero, the entry of $E \otimes F$ in row r and column c is:

$$(E \otimes F)_{r,c} = E_{\text{div}(r,f_r),\text{div}(c,f_c)} F_{\text{mod}(r,f_r),\text{mod}(c,f_c)}$$

The Kronecker sum is instead defined only for square matrices. Given two matrices A , $a \times a$, and B , $b \times b$, the Kronecker sum of A and B is given by the $ab \times ab$ matrix

$$A \oplus B = A \otimes I_b + I_a \otimes B$$

where I_i is the $i \times i$ identity matrix. The entry of $A \oplus B$ in row r and column c is:

$$(A \oplus B)_{r,c} = \delta_{\text{div}(r,b)=\text{div}(c,b)} B_{\text{mod}(r,b),\text{mod}(c,b)} + \delta_{\text{mod}(r,b)=\text{mod}(c,b)} A_{\text{div}(r,b),\text{div}(c,b)}$$

where the ‘‘Kronecker delta’’ operator δ is defined to be one if its boolean subscript is true, zero otherwise.

We observe that the Kronecker sum and product are noncommutative operators, but the difference between $A \otimes B$ and $B \otimes A$, or $A \oplus B$ and $B \oplus A$, is only in the ordering of the indices.

References

- [1] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.
- [2] V. Amoia, G. De Micheli, and M. Santomauro. Computer-oriented formulation of transition-rate matrices via Kronecker algebra. *IEEE Transactions on Reliability*, R-30:123–132, June 1981.
- [3] V. A. Barker. Numerical Solution of Sparse Singular Systems of Equations Arising from Ergodic Markov Chains. *Stochastic Models*, 5(3):335–381, 1989.
- [4] M. Becker, C. Dekoninck, J. P. Prost, and B. Verrier. Stochastic Petri net model for the FPS/264. *Computer System Science and Engineering*, 5(2), Apr. 1990.
- [5] J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, CAS-25:772–781, Sept. 1978.
- [6] P. Bucholz. Numerical solution methods based on structured descriptions of Markovian models. In G. Balbo and G. Serazzi, editors, *Computer performance evaluation*. Elsevier Science Publishers B.V. (North Holland), 1991.

- [7] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets and multiprocessor modelling applications. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*, pages 504–530. Springer-Verlag, 1991.
- [8] H. Choi and K. S. Trivedi. Approximate Performance Models of Polling Systems using Stochastic Petri Nets. In *Proceedings of the IEEE INFOCOM 92*, Florence, Italy, May 1992.
- [9] G. Ciardo. Le reti di Petri stocastiche generalizzate: uno strumento per la modellizzazione di sistemi distribuiti. Tesi di Laurea, Istituto di Scienze dell’ Informazione, Università di Torino, Italy, July 1982.
- [10] G. Ciardo. *Analysis of large stochastic Petri net models*. PhD thesis, Duke University, Durham, NC, USA, 1989.
- [11] 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*. Springer-Verlag, Heidelberg, 1992.
- [12] G. Ciardo and K. S. Trivedi. Solution of large GSPN models. In W. J. Stewart, editor, *Numerical Solution of Markov Chains*, pages 565–595. Marcel Dekker, Inc., New York, 1991.
- [13] P. J. Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, New York, 1977.
- [14] G. Dahlquist and A. Björck. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, N.J., 1974.
- [15] S. Donatelli. Superposed Stochastic Automata: a class of stochastic Petri nets amenable to parallel solution. In *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models (PNPM91)*, Melbourne, Australia, Dec. 1991.
- [16] R. A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. John Wiley and Sons, New York, NY, 1971.
- [17] G. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand-Reinhold, New York, 1960.
- [18] V. O. K. Li and J. A. Silvester. Performance Analysis of Networks with Unreliable Components. *IEEE Transactions on Communications*, COM-32(1), Oct. 1984.
- [19] R. R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. In *Proceedings of the 1989 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Berkeley, CA, USA, May 1989.
- [20] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
- [21] C. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, West Germany, 1962.
- [22] B. Plateau and K. Atif. Stochastic Automata Network for modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, Oct. 1991.
- [23] R. A. Sahner and K. S. Trivedi. Performance and reliability analysis using directed acyclic graphs. *IEEE Transactions on Software Engineering*, Oct. 1987.

- [24] W. H. Sanders and J. F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, 1991.
- [25] K. C. Sevcik. Priority scheduling disciplines in queueing network models of computer systems. In *1977 IFIP Congress Proceedings*, 1977.
- [26] L. Tomek and K. S. Trivedi. Fixed-Point Iteration in Availability Modeling. In M. Dal Cin, editor, *Informatik-Fachberichte, Vol. 91: Fehlertolerierende Rechensysteme*. Springer-Verlag, Berlin, 1991.
- [27] K. S. Trivedi and R. M. Geist. Decomposition in reliability analysis of fault-tolerant systems. *IEEE Transactions on Reliability*, Dec. 1983.
- [28] N. M. Van Dijk. Truncation of Markov chains with applications to queueing. *Operations Research*, 39(6):1018–1026, July-Aug. 1991.

Place	Meaning
M_1	idle instances of M_1
M_2	idle M_2 (or M_2 processing a rough P_3)
M_3	idle instances of M_3
P_1	rough P_1
P_2	rough P_2
P_3	rough P_3
P_{12}	finished (P_1, P_2) pairs
P_1wM_1	rough P_1 waiting for M_1
P_2wM_2	rough P_2 waiting for M_2
P_3M_2	rough P_3 waiting for, or processing at, M_2
$P_{12}wM_3$	finished (P_1, P_2) pairs waiting for M_3
P_1M_1	rough P_1 processing at M_1
P_2M_2	rough P_2 processing at M_2
$P_{12}M_3$	finished (P_1, P_2) pairs assembling at M_3
P_1d	finished P_1 deciding whether to join a P_2
P_2d	finished P_2 deciding whether to join a P_1
P_1wP_2	finished P_1 waiting for P_2
P_2wP_1	finished P_2 waiting for P_1
P_1s	finished P_1 waiting to be shipped
P_2s	finished P_2 waiting to be shipped
P_3s	finished P_3 waiting to be shipped
$P_{12}s$	finished P_{12} waiting to be shipped
Transition	Meaning
t_{P_1}	a rough P_1 goes to M_1 on a pallet
t_{P_2}	a rough P_2 goes to M_2 on a pallet
t_{P_3}	a rough P_3 goes to M_2 on a pallet
$t_{P_{12}}$	a finished (P_1, P_2) pair goes to M_3 on a pallet
t_{M_1}	M_1 starts processing a rough P_1
t_{M_2}	M_2 starts processing a rough P_2
t_{M_3}	M_3 starts assembling a (P_1, P_2) pair into a P_{12}
$t_{P_1M_1}$	M_1 ends processing a P_1
$t_{P_2M_2}$	M_2 ends processing a P_2
$t_{P_3M_2}$	M_2 ends processing a P_3
$t_{P_{12}M_3}$	M_3 ends assembling a (P_1, P_2) pair into a P_{12}
t_{P_1s}	finished P_1 ship, rough P_1 arrive
t_{P_2s}	finished P_2 ship, rough P_2 arrive
t_{P_3s}	finished P_3 ship, rough P_3 arrive
$t_{P_{12}s}$	finished P_{12} ship, rough P_1 and P_2 arrive
t_x	a (P_1, P_2) pair starts on the assembly path
t_{P_1e}	a finished P_1 decides to exit (not to join a P_2)
t_{P_1j}	a finished P_1 decides to join a P_2
t_{P_2e}	a finished P_2 decides to exit (not to join a P_1)
t_{P_2j}	a finished P_2 decides to join a P_1

Table 1: Places and transitions in the FMS model.

Transition	Rate (min^{-1})
t_{P1}	$\#(P1) \min\{1, N_p/r\}$
t_{P2}	$\#(P2) \min\{1, N_p/r\}$
t_{P3}	$\#(P3) \min\{1, N_p/r\}$
t_{P12}	$\#(P12) \min\{1, N_p/r\}$
t_{P1M1}	$\#(P1M1)/4$
t_{P2M2}	1/6
t_{P3M2}	1/2
t_{P12M3}	$\#(P12M3)$
t_{P1s}	1/60
t_{P2s}	1/60
t_{P3s}	1/60
t_{P12s}	1/60
Transition	Weight
t_{P1e} vs. t_{P1j}	0.8 vs. 0.2
t_{P2e} vs. t_{P2j}	0.6 vs. 0.4

Table 2: Rates and weights in the FMS model.

N	ψ_{exact}	ψ_{approx}	% Error
1	13.853148	13.239658	-4.428524
2	29.154731	28.243110	-3.126837
3	44.443713	42.695356	-3.933868
4	59.551361	56.318113	-5.429344
5	74.373573	69.019672	-7.198660

Table 3: Quality of the approximation.

N	$ \mathcal{T}^A $	η^A	n^A	Total _e	$ \mathcal{T}^{A_1} $	η^{A_1}	n^{A_1}	n^{A_2}	$ \mathcal{T}^{A_3} $	η^{A_3}	n^{A_3}	K	Total _a
1	54	155	68	10,540	7	8	2	2	3	3	1	7	245
2	810	3,699	83	307,017	28	56	9	11	6	9	3	9	10,323
3	6,520	37,394	95	3,552,430	84	224	12	18	10	18	4	8	54,336
4	35,910	237,120	102	24,186,240	210	672	13	23	15	30	4	9	218,808
5	152,712	1,111,482	109	121,151,538	462	1,680	14	29	21	45	4	8	579,360
6	-	-	-	-	924	3,696	14	33	28	63	4	9	1,565,676
7	-	-	-	-	1,716	7,392	15	38	36	84	4	9	3,529,008
8	-	-	-	-	3,003	13,728	15	42	45	108	4	9	7,046,352
9	-	-	-	-	5,005	24,024	16	45	55	135	4	8	11,728,032
10	-	-	-	-	8,008	40,040	17	48	66	165	4	8	20,826,080
15	-	-	-	-	54,264	310,080	18	61	136	360	4	8	195,982,080

Table 4: Complexity of the analysis of the FMS.

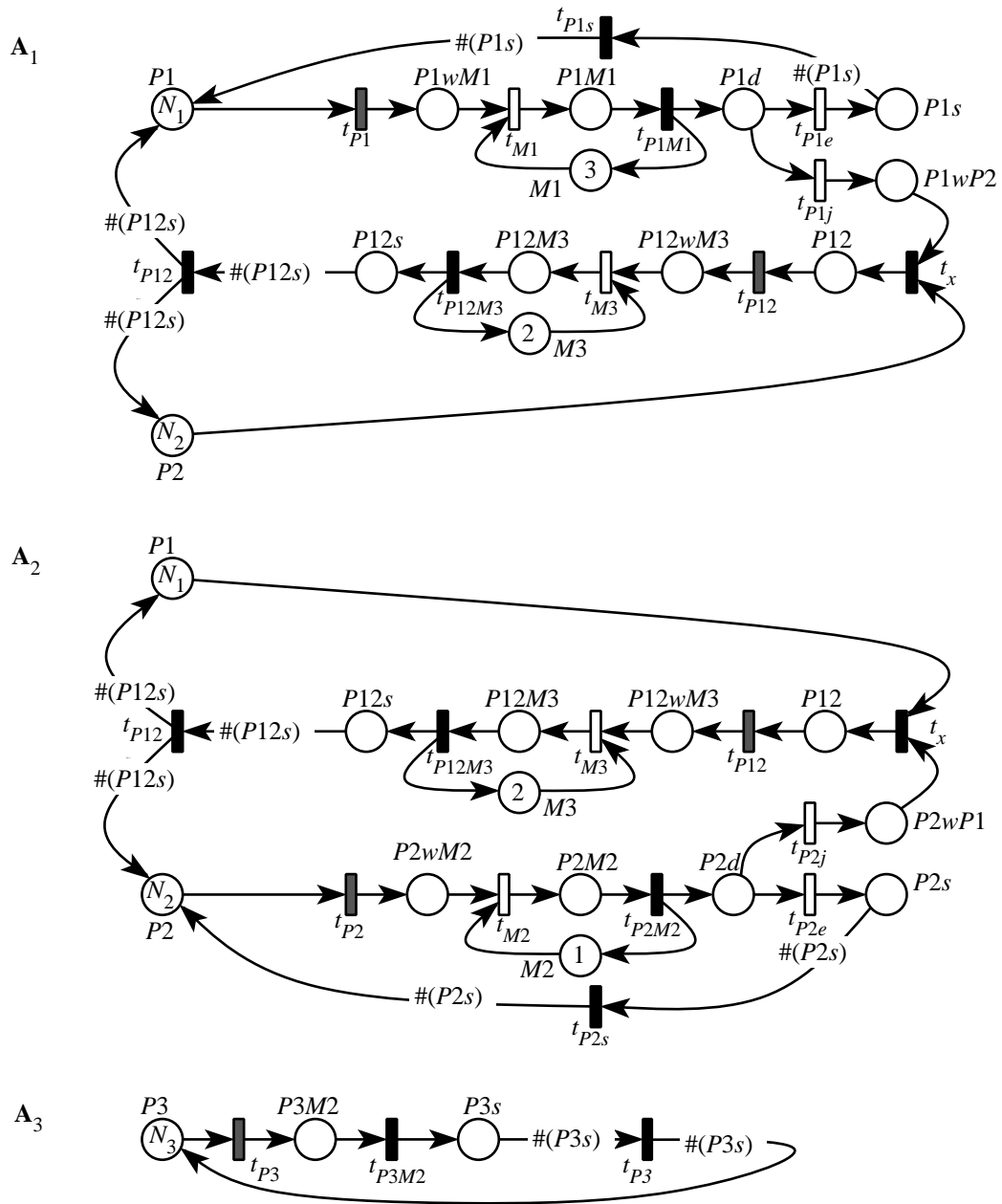


Figure 8: Decomposition of the FMS.

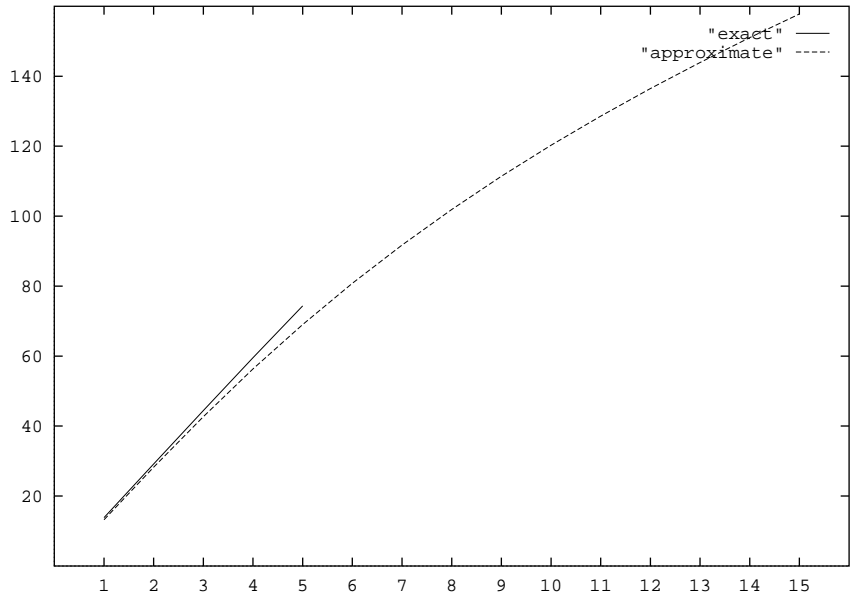


Figure 9: Productivity ψ as a function of N .