A Decomposition Approach for Stochastic Petri Net Models

Gianfranco Ciardo

Software Productivity Consortium
Herndon, VA 22070, USA

Kishor S. Trivedi

Duke University
Durham, NC 27706, USA

Abstract

We present a decomposition approach for the solution of large stochastic Petri nets (SPNs). 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 parametrized SPN 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 is normally cyclic, so the solution method is based on fixed point iteration. We apply our technique to the analysis of a flexible manufacturing system.*

1 Introduction

Stochastic Petri nets of various ilk are found to be powerful in modeling performance and dependability of computer and communications systems. If the Petri net is restricted to be Markovian, then a wealth of numerical solution algorithms of Markov models [1] can be used to solve SPN models. 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. We can avoid the generation of a large state space if we resort to Monte-Carlo simulation. In addition, simulation allows us to treat non-Markovian SPNs as well. The major drawback of simulation is the potentially large execution time needed.

An attractive alternative to the costly approaches above is the use of model decomposition. We visualize the scenario in which the system has been decomposed into subsystems and an SPN submodel for each subsystem is developed. The modeler then composes an overall model from the constituent submodels, carefully taking into account various interactions among subsystems. Our approach then consists of taking advantage of this natural decomposition by solving subsystems in isolation and exchanging solution components as and when required. 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 to resolve such cyclic situations [2, 3].

Three important issues arise: how to decompose a net into subnets, what kinds of quantities to pass from one subnet to another, and 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 SPN 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 intercommunication among subnets for the net structure types that we define. Convergence of the iterative schemes is discussed elsewhere [3].

In Section 2, we introduce the concepts of near-independence and near-decomposition. In Section 3, we introduce our approach based on near-independence. In Section 4, we illustrate our approach using a model of a flexible-manufacturing system. In Section 5, we present some concluding remarks. We concentrate on the class of Markovian stochastic Petri nets known as Generalized Stochastic Petri Nets (GSPNs) throughout this paper [4].

2 Decomposability and independence

Courtois [5] 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 chain is the main limitation to the study of GSPN, so an analysis approach that modifies the model, rather than the Markov matrix, is extremely desirable. Examples of approaches that do not require the generation of the Markov chain in its entirety are state truncation [6], hierarchical decomposition [7], behavioral decomposition [8], and folding [9]. These methods have proved to be difficult to generalize but are very effective, when applicable.

Consider the GSPN $\mathbf{C}$, obtained by composing two independent GSPNs $\mathbf{A}$ and $\mathbf{B}$. Construct $R^{\mathbf{C}}$ and $S^{\mathbf{C}}$, the transition rate matrix of the underlying continuous-time Markov chain (CTMC) and the tangible reachability set of the GSPN $\mathbf{C}$, and compare them to $R^{\mathbf{A}}$, $S^{\mathbf{A}}$, $R^{\mathbf{B}}$, and $S^{\mathbf{B}}$, the transition rate matrices and the tangible reachability sets of $\mathbf{A}$ and $\mathbf{B}$ (we 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

TH0386-3/91/0000/0074/$01.00 © 1991 IEEE

$$R^{\mathbf{C}} = \left[\begin{array}{c|c|c|c} R^{\mathbf{B}} & \begin{array}{c} R^{\mathbf{A}}_{1,2} \\ \cdots \\ R^{\mathbf{A}}_{1,2} \end{array} & \cdots & \begin{array}{c} R^{\mathbf{A}}_{1,m} \\ \cdots \\ R^{\mathbf{A}}_{1,m} \end{array} \\ \hline \begin{array}{c} R^{\mathbf{A}}_{2,1} \\ \cdots \\ R^{\mathbf{A}}_{2,1} \end{array} & R^{\mathbf{B}} & \cdots & \begin{array}{c} R^{\mathbf{A}}_{2,m} \\ \cdots \\ R^{\mathbf{A}}_{2,m} \end{array} \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline \begin{array}{c} R^{\mathbf{A}}_{m,1} \\ \cdots \\ R^{\mathbf{A}}_{m,1} \end{array} & \begin{array}{c} R^{\mathbf{A}}_{m,2} \\ \cdots \\ R^{\mathbf{A}}_{m,2} \end{array} & \cdots & R^{\mathbf{B}} \end{array}\right] \quad (1)$$

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

in their diagonal: $Q_{i,i} = -\sum_{j \neq i} Q_{i,j}$, while $R_{i,i} = 0$). If $|S^{\mathbf{A}}| = m$ and $|S^{\mathbf{B}}| = n$, then $R^{\mathbf{A}}$ is $m \times m$ and $R^{\mathbf{B}}$ is $n \times n$. $S^{\mathbf{C}}$ is the cross-product $S^{\mathbf{A}} \times S^{\mathbf{B}}$. $R^{\mathbf{C}}$ can be expressed as the Kronecker sum [10]

$$R^{\mathbf{C}} = R^{\mathbf{A}} \oplus R^{\mathbf{B}} = R^{\mathbf{A}} \otimes I_n + I_m \otimes R^{\mathbf{B}}$$

where I_i is the $i \times i$ identity matrix. We recall that the Kronecker product "$\otimes$" of two matrices E, $e_r \times e_c$, and F, $f_r \times f_c$, is the $e_r f_r \times e_c f_c$ matrix

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

$R^{\mathbf{C}}$ consists of m^2 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 $R^{\mathbf{A}}_{i,j}$.

The Kronecker sum and product are noncommutative operators, but $R^{\mathbf{A}} \oplus R^{\mathbf{B}}$ and $R^{\mathbf{B}} \oplus R^{\mathbf{A}}$ differ only in the ordering of the indices.

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 because the corresponding matrices are visually simpler.

Assuming a direct method is used, the steady-state solution of $\mathbf{C}$ requires $O(m^3 n^3)$ 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 (iterative methods offer analogous savings).

We call this property **independence** and we stress that it is completely orthogonal to **decomposability**. This is apparent when comparing the concepts of **near-independence** and **near-decomposability**.

A transition rate matrix is near-decomposable if it can be reordered and partitioned so that the entries of the off-diagonal blocks are small. The diagonal blocks are not required to have anything in common, they correspond to disjoint subsets of states for a system. By contrast, 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 of (1), but they are not at all required to have small entries. Each diagonal block corresponds to the same set of "partial states" (obtained by considering only some of the components in the description of the states of $\mathbf{C}$).

$$R^{\mathbf{C}^a} = \left[\begin{array}{ccc|c} - & \mu_3 & & \mu_1(1) - \\ \mu_4 & - & & -\,\mu_1(0) \\ \hline \mu_2 & - & - & \mu_3 \\ - & \mu_2 & \mu_4 & - \end{array}\right]$$

$$R^{\mathbf{C}^b} = \left[\begin{array}{cc|cc} - & \mu_3 & \mu_1 & \mu_5 \\ \mu_4 & - & - & \mu_1 \\ \hline \mu_2 & - & - & \mu_3 \\ - & \mu_2 & \mu_4 & - \end{array}\right]$$

$$R^{\mathbf{C}^c} = \left[\begin{array}{cc|cc} - & \mu_3 & \mu_1 & - \\ \mu_4 & - & - & - \\ \hline \mu_2 & - & - & \mu_3 \\ - & \mu_2 & \mu_4 & - \end{array}\right]$$

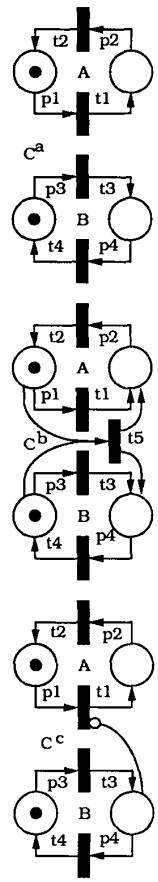

Figure 2: Near-independent CTMC structures.

2.1 Near-independent CTMC structures

The level 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 level 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 attempt a qualitative characterization by describing the three basic near-independent structures that can arise in the CTMC

underlying a near-independent GSPN.

Rate-dependence. Figure 2 contains three GSPNs, $\mathbf{C}^a$, $\mathbf{C}^b$, and $\mathbf{C}^c$ (assume that the rate of transition t_i is μ_i), each with two subnets, **A** and **B**, composed in different ways. In $\mathbf{C}^a$, **A** and **B** are structurally disconnected. If $\forall i, \mu_i$ is a constant, **A** and **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 value of some elements varies. This requires that at least one rate in one subnet be a function of the marking of another subnet. For example, if μ_1 is a function of $\#(p_3)$, the number of tokens in p_3, the corresponding $R^{\mathbf{C}^a}$ is in figure 2, assuming that the order of the GSPN markings is ((1010),(1001),(0110),(0101)). The smaller the difference between $\mu_1(1)$ and $\mu_1(0)$ 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 GSPN, such an entry describes the simultaneous change of marking in two or more subnets. Since simultaneous firings of timed transitions have probability zero, the change must be caused by the firing of a synchronizing transition such as t_5 in GSPN $\mathbf{C}^b$, with inputs or output places in **A** and **B**. The smaller μ_5 is with respect to μ_1 and μ_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 GSPN, 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 (**A**) depending on the marking of another subnet (**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 μ_1 is with respect to μ_4 (the rate at which the inhibiting arc becomes irrelevant), the closer $R^{\mathbf{C}^c}$ is to be independent.

Multiple near-dependent 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.
- 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, and then changing it back to zero in all but one block.

In many models, all three interactions are present and it might be difficult to recognize them individually. In addition, the state-space itself may be incomplete. For example, define GSPN $\mathbf{C}^d$, obtained by adding an inhibitor arc from p_2 to t_3 in GSPN $\mathbf{C}^c$ of figure 2. **A** and **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 $S^{\mathbf{C}^d}$ 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 GSPNs **A** and **B** is not out of the question, but trying to obtain the probability of events relating the markings of **A** and **B** is difficult. If the GSPN is completely independent the answer is just the product of the probabilities of the each submarking in each individual subnet, but for $\mathbf{C}^d$, this is false. 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 $S^{\mathbf{C}^d}$ (when ambiguities could arise, we add the name of the particular GSPN as a subscript to the "#" function).

3 GSPN decomposition

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

1. **Decomposition.** Given a GSPN **A**, generate GSPNs $\mathbf{A}_1, \ldots \mathbf{A}_k$. Often, $\mathbf{A}_i$ is not a subnet of **A**, but it shares common subnets with it. These GSPNs are parametrized: 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 SPN $\mathbf{A}_i$, fix the value of its parameters using *imports* from $\mathbf{A}_j, 1 \leq j \leq k$. Also, specify the quantities exported from $\mathbf{A}_i$ after its solution. If $\mathbf{A}_i$ imports a quantity from $\mathbf{A}_j$, we write $\mathbf{A}_j \succ \mathbf{A}_i$. We denote the transitive closure of the import relation with the symbol $\twoheadrightarrow$. The graph describing the import relation may be cyclic (the case $\mathbf{A}_i \succ \mathbf{A}_i$, a cycle of length one, may occur).
3. **Iteration.** If the import graph is acyclic, it implicitly defines a (partial) order for the solution of the GSPNs $\mathbf{A}_i$. If $\mathbf{A}_i \twoheadrightarrow \mathbf{A}_j$, $\mathbf{A}_i$ must be studied before $\mathbf{A}_j$. If neither $\mathbf{A}_i \twoheadrightarrow \mathbf{A}_j$ nor $\mathbf{A}_j \twoheadrightarrow \mathbf{A}_i$, 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. Conver-

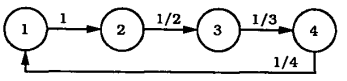

Figure 3: Example for the computation of μ and δ.

gence is reached when all the imports remain (almost) constant between successive iterations.

3.1 Primitives for the exchange of data

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

Given a GSPN **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:

$\mu_{\mathbf{A}}(c)^{-1} = E[\text{time } c \text{ remains } ON \text{ in steady-state}]$

$Pr_{\mathbf{A}}\{c\} = Pr\{\text{c is } ON \text{ in steady-state}\}$

$\delta_{\mathbf{A}}(c)^{-1} = E[\text{time until } c \text{ is } ON \text{ in steady-state}]$

Partition the tangible reachability set S of **A** into S_c and $S_{\overline{c}}$, where condition c is respectively *ON* or *OFF*. If $\underline{\pi}$ is the steady-state probability vector for the underlying CTMC,

$$\mu_{\mathbf{A}}(c)^{-1} = (\sum_{k \in S_c} \pi_k)/(\sum_{i \in S_c, j \in S_{\overline{c}}} \pi_i Q_{i,j})$$

That is, $\mu_{\mathbf{A}}(c)$ 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 GSPN is ergodic and neither S_c nor $S_{\overline{c}}$ are empty.

$$Pr_{\mathbf{A}}\{c\} = \sum_{k \in S_c} \pi_k$$

$Pr_{\mathbf{A}}\{c\}$ is simply the sum of the steady-state probability of each marking in S_c.

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

$$\delta_{\mathbf{A}}(c)^{-1} = \sum_{k \in S_{\overline{c}}} x_k^{\overline{c}} \quad \text{where} \quad \underline{x}^{\overline{c}} Q^{\overline{c}} = -\underline{\pi}^{\overline{c}}$$

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

Both $\mu(c)$ and $\delta(\overline{c})$ refer to sojourns in markings where c holds, but they have different values. For example, consider the CTMC in figure 3, where the sojourn time in state i has exponential distribution with parameter i^{-1}. The steady-state probability vector is $\underline{\pi} = [0.1, 0.2, 0.3, 0.4]$. If $S_c = \{1, 2\}$ and $S_{\overline{c}} = \{3, 4\}$,

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

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

Then,

$$\mu(c)^{-1} = \frac{0.1 + 0.2}{0.2\frac{1}{2}} = 3 \quad \neq \quad \delta(\overline{c})^{-1} = 0.1 + 0.6 = 0.7$$

$$\mu(\overline{c})^{-1} = \frac{0.3 + 0.4}{0.4\frac{1}{4}} = 7 \quad \neq \quad \delta(c)^{-1} = 0.9 + 2.8 = 3.7$$

If we only know that condition b holds in the set of markings we are interested, 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

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

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

$$\mu(c|b)^{-1} = (\sum_{k \in S_c} \pi_k^b)/(\sum_{i \in S_c, j \in S_{\overline{c}}} \pi_i^b Q_{i,j})$$

The denominator in the equation for $\mu(c|b)^{-1}$ 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 $\mu(c|b)$ to be zero, regardless of the value of the numerator, since this corresponds to an event which cannot happen.

$$Pr\{c|b\} = \sum_{k \in S_c} \pi_i^b$$

$$\delta(c|b)^{-1} = \sum_{k \in S_{\overline{c}}} x_k^{\overline{c}|b} \quad \text{where} \quad \underline{x}^{\overline{c}|b} Q^{\overline{c}} = -\underline{\pi}^{\overline{c}|b}$$

and $\underline{\pi}^{\overline{c}|b}$ is the restriction of $\underline{\pi}^b$ to the states in $S_{\overline{c}}$.

$Pr\{c\}$ can be derived from $\mu(c)$ and $\mu(\overline{c})$. First, observe that, in steady-state, the rate at which S_c and $S_{\overline{c}}$ are entered is the same (conservation of flow), so we can write

$$\sum_{i \in S_c, j \in S_{\overline{c}}} \pi_i Q_{i,j} = \sum_{i \in S_{\overline{c}}, j \in S_c} \pi_i Q_{i,j} = Q_c$$

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

$$\frac{\mu(\overline{c})}{\mu(c) + \mu(\overline{c})} = \frac{Q_c / Pr\{\overline{c}\}}{Q_c / Pr\{c\} + Q_c / Pr\{\overline{c}\}} = Pr\{c\}$$

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

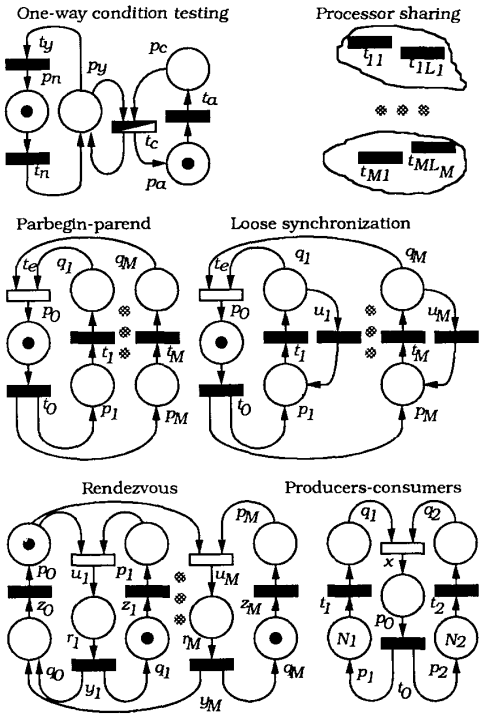

Figure 4: Some near-independent GSPN structures.

3.2 Near-independent GSPN structures

Figure 4 contains some near-independent GSPN structures which can be decomposed using our approach. Timed transitions are represented with black filled rectangles. In these structures, each timed transition could have an arbitrary phase-type distribution instead of a simple exponential distribution. The decomposition approach will still apply, but the quality of the approximation will differ. See [3, 11] for further details.

One-way condition testing. Transition t_c is drawn only half filled because it can be either timed or immediate. 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 portion of the GSPN describes the change of the condition, and is completely independent of the right portion, which must occasionally test the condition before it can continue.

For example, the left and right portions could represent a high and 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 (timed) represent the use of the resource, while transitions t_y and t_a represent periods during which the shared resource is not needed.

The right portion must import from the left portion information such as $Pr\{\#(p_y) = 1\}$ and $\delta(\#(p_y) = 1)$. The left portion is unaffected by the right portion, so it has no imports. This is an example of external-dependence.

Processor-sharing. A GSPN composed of M structurally disconnected (sub)GSPNs with transitions sharing resources in processor-sharing mode is one of the most intuitive cases of near-independence. In GSPN i, transitions $t_{i1}, \ldots, t_{iL_i}$ 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 $r_{ij} \min\{1, R/\sum_{k,l} r_{kl}\}$. 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 GSPN i must know from each other GSPN k information about its resource usage, such as the expected number of requests in steady state or, even better, its distribution:

$$\forall k, 1 \le k \le M, k \ne i, \forall r, 0 \le r \le T_k, Pr\{\sum_{1 \le l \le L_k} r_{kl} = r\}$$

where T_k is the maximum number of requests to the resource that can be originated by the transitions of GSPN 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 once, to compute $T_i = \delta(\#(q_i) = 1 | \#(p_i) = 1)^{-1}$, 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 very complex subnet). Then, the original GSPN can be studied assuming that the firing time for t_i is exponentially distributed with parameter T_i^{-1}. 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 $(C - T_i)^{-1}$, since $C - T_i$ represent the average amount of time thread i is not in use.

This decomposition still requires the generation of large reachability graphs (the original GSPN where all the transitions t_i have exponential distributions generates $2^M + 1$ markings), but it provides large savings if the transitions correspond to complex timed activities (complex subnets). Furthermore, the last step where each individual thread is solved again with the addition of transition w_i is particularly useful when the threads are complex and have an internal state. The subnet representing thread i could in fact have internal activities which are near-independent of

the main execution thread. In this case, then, the initial marking for the subnet might affect the computation of T_i, so iterations would be needed.

Loose synchronization. The structure of a loose-synchronization GSPN differs from that of the parbegin-parend in one important aspect. In both GSPNs, 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_e. 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. Indeed, the decomposition of the loose synchronization structure results in better approximations than that of tight synchronization.

Rendezvous. The rendezvous structure shown in figure 4 derives its name from the interaction between Ada tasks [12], 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, or a resource. The tokens in q_i ($1 \leq i \leq M$) represent M caller tasks, or potential users for the resource. A token in place r_i signifies that the callee is in rendezvous with caller i, or that the resource has been assigned to user i.

Producers-consumers. GSPNs modeling apparently different phenomena may be very similar: the parbegin-parend GSPN with $M = 2$ has exactly the same structure as the rendezvous GSPN with $M = 1$, and both of them differ from the producer-consumer GSPN only in the initial marking.

This last structure can model N_1 producer tasks interacting with N_2 consumer tasks, for example, or the assemblage of two different types of parts in an automated assembly line. Parts of type 1 and 2 are machined separately (transitions t_1 and t_2), then they are assembled together (transition t_0). The number of parts in the system is kept constant by introducing one new raw part of each type whenever an assembled product exits the line.

Tight and loose synchronization, rendezvous, and producers-consumers are examples of synchronization-dependence coupled with external-dependence (e.g., in the parbegin-parend, the serial thread "inhibits" the M concurrent threads).

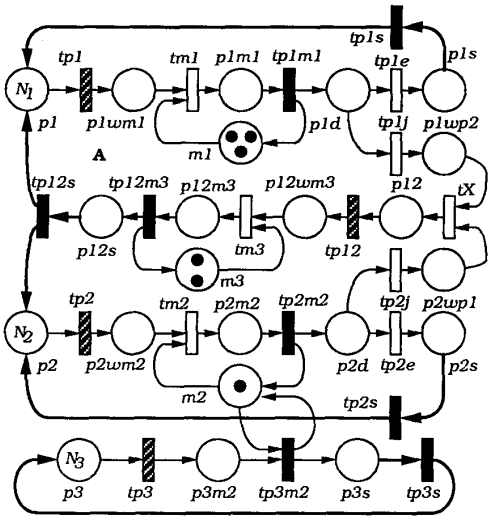

Figure 5: A FMS with three machines.

4 An application

We apply our decomposition approach to a system model exhibiting several of the behaviors of the previous section. The SPN **A** in figure 5 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 P_1, up to three at a time. Machine M_2 normally processes parts of type P_2, one at a time, but it can also process parts of type P_3, when no part of type P_2 is ready to be processed.

Finished parts of type P_1 and P_2 may also be assembled together (one part of each type) into a new part type, P_{12}; machine M_3, duplicated, is used for this purpose. When finished parts of type P_1, P_2, P_3, and P_{12} 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}, are drawn thicker; they have a marking-dependent cardinality 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 P_1 and P_2 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 P_1, P_2, and P_3 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 probabilities. The pallets are a shared resource; for the purpose

Place	Meaning
$m1$	idle instances of M_1
$m2$	idle M_2 (or M_2 processing a rough P_3)
$m3$	idle instances of M_3
$p1$	rough P_1
$p2$	rough P_2
$p3$	rough P_3
$p12$	finished (P_1, P_2) pairs
$p1wm1$	rough P_1 waiting for M_1
$p2wm2$	rough P_2 waiting for M_2
$p3m2$	rough P_3 waiting for, or processing at, M_2
$p12wm3$	finished (P_1, P_2) pairs waiting for M_3
$p1m1$	rough P_1 processing at M_1
$p2m2$	rough P_2 processing at M_2
$p12m3$	finished (P_1, P_2) pairs assembling at M_3
$p1d$	finished P_1 deciding whether to join a P_2
$p2d$	finished P_2 deciding whether to join a P_1
$p1wp2$	finished P_1 waiting for P_2
$p2wp1$	finished P_2 waiting for P_1
$p1s$	finished P_1 waiting to be shipped
$p2s$	finished P_2 waiting to be shipped
$p3s$	finished P_3 waiting to be shipped
$p12s$	finished P_{12} waiting to be shipped
Transition	**Meaning**
t_{p1}	a rough P_1 goes to M_1 on a pallet
t_{p2}	a rough P_2 goes to M_2 on a pallet
t_{p3}	a rough P_3 goes to M_2 on a pallet
t_{p12}	a finished (P_1, P_2) pair goes to M_3 on a pallet
t_{m1}	M_1 starts processing a rough P_1
t_{m2}	M_2 starts processing a rough P_2
t_{m3}	M_3 starts assembling a (P_1, P_2) pair into a P_{12}
t_{p1m1}	M_1 ends processing a P_1
t_{p2m2}	M_2 ends processing a P_2
t_{p3m2}	M_2 ends processing a P_3
t_{p12m3}	M_3 ends assembling a (P_1, P_2) pair into a P_{12}
t_{p1s}	finished P_1 ship, rough P_1 arrive
t_{p2s}	finished P_2 ship, rough P_2 arrive
t_{p3s}	finished P_3 ship, rough P_3 arrive
t_{p12s}	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_{p1e}	a finished P_1 decides to exit (not to join a P_2)
t_{p1j}	a finished P_1 decides to join a P_2
t_{p2e}	a finished P_2 decides to exit (not to join a P_1)
t_{p2j}	a finished P_2 decides to join a P_1

Table 1: Places and transitions in the FMS model.

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.

We study 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.

4.1 Decomposition of the model

A can be studied only for small values of N_1, N_2, and N_3, since its reachability set soon becomes excessively large.

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	**Probability**
t_{p1e} vs. t_{p1j}	0.8 vs. 0.2
t_{p2e} vs. t_{p2j}	0.6 vs. 0.4

Table 2: Rates and probabilities in the FMS model.

We then decompose **A** into three GSPNs: $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ (figure 6). $\mathbf{A}_1$ describes the flow of P_1 parts, including their assemblage with P_2 parts, $\mathbf{A}_2$ describes the flow of P_2 parts, including their assemblage with P_1 parts, and $\mathbf{A}_3$ describes the flow of P_3 parts. Place lim in $\mathbf{A}_1$ and $\mathbf{A}_2$ is used to limit the total number of tokens in the subnet assembling P_1 and P_2 parts to $N_{12} = \min\{N_1, N_2\}$. In any marking, $N_{12} - \#(lim) = \#(p12) + \#(p12wm3) + \#(p12m3) + \#(p12s)$.

The reachability sets of the three GSPNs are much smaller than the one for **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:

$$\mu_{\mathbf{A}_1}(t_{p1}) = \sum_{i_2=0}^{N_2}\sum_{i_3=0}^{N_3} u_2[i_2, \#(lim)]u_3[i_3]\#(p1) \min\{1, N_p/(\#(p1)+\#(p12)+i_2+i_3)\}$$

$$\mu_{\mathbf{A}_1}(t_{p12}) = \sum_{i_2=0}^{N_2}\sum_{i_3=0}^{N_3} u_2[i_2, \#(lim)]u_3[i_3]\#(p12) \min\{1, N_p/(\#(p1)+\#(p12)+i_2+i_3)\}$$

$$\mu_{\mathbf{A}_2}(t_{p2}) = \sum_{i_1=0}^{N_1}\sum_{i_3=0}^{N_3} u_1[i_1, \#(lim)]u_3[i_3]\#(p2) \min\{1, N_p/(\#(p2)+\#(p12)+i_1+i_3)\}$$

$$\mu_{\mathbf{A}_2}(t_{p12}) = \sum_{i_1=0}^{N_1}\sum_{i_3=0}^{N_3} u_1[i_1, \#(lim)]u_3[i_3]\#(p12) \min\{1, N_p/(\#(p2)+\#(p12)+i_1+i_3)\}$$

$$\mu_{\mathbf{A}_3}(t_{p3}) = \sum_{j=0}^{N_{12}}\sum_{i_1=0}^{N_1}\sum_{i_2=0}^{N_2} u_{12}[j]u_1[i_1, j]u_2[i_2, j]\#(p3) \min\{1, N_p/(\#(p3)+j+i_1+i_2)\}$$

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

$$u_1[i_1, j] = Pr_{\mathbf{A}_1}\{\#(p1) = i_1 | \#(lim) = j\}$$
$$u_2[i_2, j] = Pr_{\mathbf{A}_2}\{\#(p2) = i_2 | \#(lim) = j\}$$
$$u_3[i_3] = Pr_{\mathbf{A}_3}\{\#(p3) = i_3\}$$
$$u_{12}[j] = (Pr_{\mathbf{A}_1}\{\#(p12) = j\} + Pr_{\mathbf{A}_2}\{\#(p12) = j\})/2$$

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 $\mathbf{A}_3$. Unfortunately, the estimate for this pmf obtained from $\mathbf{A}_1$ is different than the one obtained from $\mathbf{A}_2$, so we average them.

$\mathbf{A}_1$ 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:

$$\mu_{\mathbf{A}_1}(t_X) = d_2[\#(lim)], \quad \mu_{\mathbf{A}_2}(t_X) = d_1[\#(lim)]$$

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

$$d_1[j] = \delta_{\mathbf{A}_1}(\#(p1wp2) > 0 | \#(lim) = j)$$
$$d_2[j] = \delta_{\mathbf{A}_2}(\#(p2wp1) > 0 | \#(lim) = j)$$

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 j tokens are in lim ($N_{12} - j$ tokens are in $p12$, $p12wm3$, $p12m3$, and $p12s$). Then, the the rate of t_X in $\mathbf{A}_1$ in a marking where $\#(lim) = j$ is simply equated to the inverse of this time.

Finally, parts P_3 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:

$$\mu_{\mathbf{A}_3}(t_{p3m2}) = \left(\frac{2}{Pr_{\mathbf{A}_2}\{\#(p2m2)=0\}} + \frac{1}{\delta_{\mathbf{A}_2}(\#(p2m2)=0)}\right)^{-1}$$

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, 2, by a "rarefaction" quantity equal to the probability of finding machine M_2 not in use by P_2 parts (shadow CPU approximation [13]). The second time is a "delay" portion equal to the expected time before M_2 can become available to P_3 parts, starting from steady-state. See [3] for a detailed discussion of the approximations obtained by taking into account only the rarefaction, only the delay, or both (which we use in this example, since it is usually the best).

All the above rates, with the exception of $\mu_{\mathbf{A}_3}(t_{p3m2})$, are functions of the marking. For example, the rates of t_{p1} and t_{p12} in $\mathbf{A}_1$ depend on the number of tokens in places lim, $p1$, 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 lim, $p2$, and $p12$ of $\mathbf{A}_2$, and so on.

The analysis of $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ using our iteration 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.

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

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 $\phi_{12}^{(2)}$. If, for example, $\mathbf{A}_1$ is 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 [3], we use:

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

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 7.

Table 4 shows the complexity of the analysis for the exact and the approximate solution. For the computation of the numerical results, we simplified the GSPNs by manually eliminating the immediate transitions. This results in a smaller reachability set and graph, but it does not affect the size of the CTMC or the numerical results.

In table 4, $|S^{\mathbf{A}}|$, $\eta^{\mathbf{A}}$, 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. Column "Total$_e$" is equal to $\eta^{\mathbf{A}} n^{\mathbf{A}}$ and represents the total number of FLOPS. The large size of $S^{\mathbf{A}}$ 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 none of the imports, approximated to four significant digits, varies from the previous iteration). Column "Total$_a$" 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 $|S^{\mathbf{A}_2}|$ is missing because it is the same as $|S^{\mathbf{A}_1}|$. The execution time and memory requirements savings are substantial.

5 Conclusion

The decomposition approach that we discussed applies to a large class of GSPNs and appears to be extendable to other GSPN structures. We stress that our decomposition approach iteratively modifies the CTMC itself; furthermore, only three primitives are needed to define the imports: μ, Pr, and δ.

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 GSPNs need to be analyzed at each iteration, if, at the k-th iteration ($1 \leq k \leq K$), GSPN $\mathbf{A}_m$ ($1 \leq m \leq M$) 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 GSPNs $\mathbf{A}_m$ are often 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 GSPN solution methods. Sometimes the GSPN 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 $\eta_m^{(k)} = \eta_m^{(k+1)}$. It is possible to exploit this similarity 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.

References

[1] V. Barker, "Numerical Solution of Sparse Singular Systems of Equations Arising from Ergodic Markov Chains," *Stochastic Models*, 1989.

[2] M. Becker, C. Dekoninck, J. Prost, and B. Verrier, "Stochastic Petri net model for the FPS/264," *Comp. System Science and Eng.*, Apr. 1990.

[3] G. Ciardo, *Analysis of large stochastic Petri net models.* PhD thesis, Duke University, Durham, NC, USA, 1989.

[4] M. Ajmone Marsan, G. Balbo, and G. Conte, "A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems," *ACM Trans. on Comp. Systems*, May 1984.

[5] P. Courtois, *Decomposability: Queueing and Computer System Applications.* New York: Academic Press, 1977.

[6] R. Muntz, E. de Souza e Silva, and A. Goyal, "Bounding availability of repairable computer systems," *IEEE Trans. on Comp.* , Dec. 1989.

[7] R. Sahner and K. Trivedi. "Reliability modeling using SHARPE," *IEEE Trans. on Rel.*, June 1987.

[8] K. Trivedi and R. Geist, "Decomposition in reliability analysis of fault-tolerant systems," *IEEE Trans. on Rel.*, Dec. 1983.

[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] V. Amoia, G. De Micheli, and M. Santomauro, "Computer-oriented formulation of transition-rate matrices via Kronecker algebra," *IEEE Trans. on Rel.*, June 1981.

[11] G. Ciardo and K. Trivedi, "Solution of large GSPN models," in *Numerical Solution of Markov Chains*, W. Stewart (ed.), New York: Marcel Dekker, 1991.

[12] U. S. Department of Defense, "Reference Manual for the Ada programming language," ANSI/MIL-STD-1815A-1983, Feb. 1983.

[13] K. Sevcik, "Priority scheduling disciplines in queueing network models of computer systems," in *IFIP Congress Proceedings*, 1977.

N	$\|S^{\mathbf{A}}\|$	$\eta^{\mathbf{A}}$	$n^{\mathbf{A}}$	Totale	$\|S^{\mathbf{A}_1}\|$	$\eta^{\mathbf{A}_1}$	$n^{\mathbf{A}_1}$	$n^{\mathbf{A}_2}$	$\|S^{\mathbf{A}_3}\|$	$\eta^{\mathbf{A}_3}$	$n^{\mathbf{A}_3}$	K	Totala
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
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.

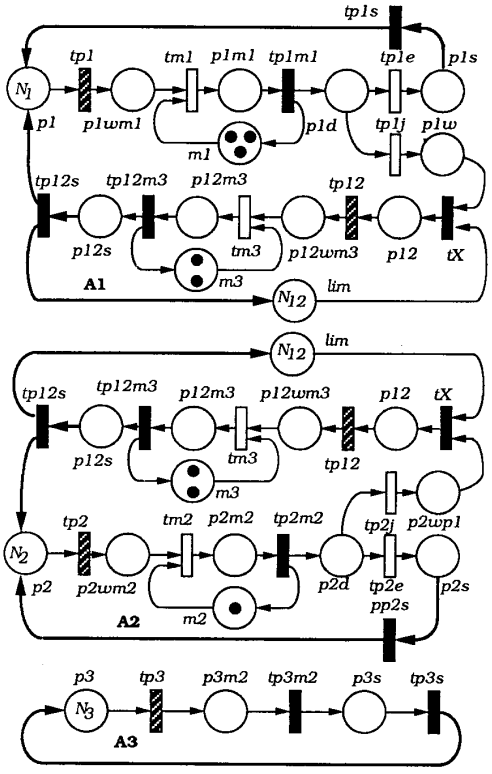

Figure 6: Decomposition of the FMS.

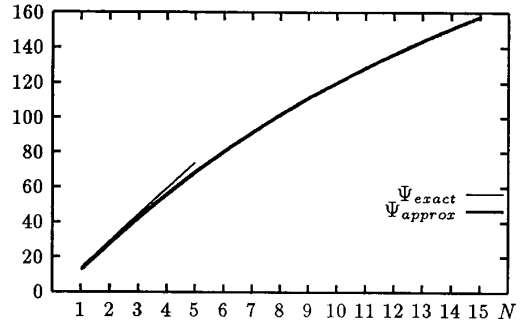

Figure 7: Productivity Ψ as a function of N.