# Approximate Computation of Transient Results for Large Markov Chains

Peter Buchholz*
Informatik IV, Universität Dortmund
D-44221 Dortmund, Germany
Email: peter.buchholz@udo.edu

William H. Sanders†
Department of Electrical and Computer Engineering and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana IL, U.S.A.
Email: whs@uiuc.edu

## Abstract

*This paper presents a new approach for the computation of transient measures in large continuous time Markov chains (CTMCs). The approach combines the randomization approach for transient analysis of CTMCs with a new representation of probability vectors as Kronecker products of small component vectors. This representation is an approximation that allows an extremely space- and time-efficient computation of transient vectors. Usually, the resulting approximation is very good and introduces errors that are comparable to those found with existing approximation techniques for stationary analysis. By increasing the space and time requirements of the approach, we can represent parts of the solution vector in detail and reduce the approximation error, yielding exact solutions in the limiting case.*

## 1. Introduction

Transient analysis is often needed for performance and dependability analysis. In contrast to stationary analysis, for which several efficient analysis methods for specific models are available, transient analysis typically requires numerical analysis of the CTMC or discrete event simulation. Both approaches have their drawbacks, since numerical analysis suffers from state space explosion, and simulation can estimate results only up to a certain level of accuracy, since it relies on observations of a stochastic process. One way to enlarge the state spaces of numerically solvable models is to represent the generator matrix of the CTMC in a compact form as a Kronecker product of small component matrices. This approach, which dates back to the pioneering work of Plateau [23], has been mainly used for stationary analysis [6, 10], but can also be applied for transient analysis when combined with the randomization approach [18, 21]. However, even if the use of a compact matrix representation permits the representation of very large matrices, it cannot eliminate the problem of state space explosion. The reason is that vectors of the size of the state space are still needed and thus memory and time requirements are still determined by the size of the state space. Several attempts have been made to represent vectors in a more compact form [9, 17], but have not been successful at significantly increasing the size of CTMCs that can be handled.

Thus, for the analysis of larger models, more efficient methods that avoid computation in the complete, combinatorially growing state space must be found. For stationary analysis, a large number of approximation methods are available; the most popular are the so-called *fixed-point approaches*, in which the model is decomposed into components and components are analyzed in partial isolation such that other components are represented by parameters that are updated in an iteration process. The approach has been successfully applied (for example, see [14, 20]). More recently, these fixed-point approaches have been combined with structured solution techniques such that a complete description of the state space and generator matrix is available, but aggregated vectors are computed [13, 4]. Furthermore, in [8] a method was developed to combine an exact and approximate computation of probabilities by using detailed vectors for parts of the state space, and a compact and approximate representation as a Kronecker product of small vectors for the rest of the state space.

For transient analysis, very few approximation methods are available. Most approximation methods for transient analysis rely on the aggregation of some unimportant states [2] or can be applied only for models with a specific structure [11], but no generally applicable method, compa-

* This research is partially supported by the DFG, as part of SFB 358

† This material is based upon work supported by Pioneer Hi-Bred and by the National Science Foundation under Grant No. 0086096. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

rable to the widely used fixed-point approximation for stationary analysis, has been published yet. In this paper we develop a new approximation method for transient analysis. The approach extends the stationary analysis technique from [8] in which solution and iteration vectors are represented as Kronecker products of small component vectors. We show that this extended approach yields a very efficient, and for most applications sufficiently accurate, approximation method. By representing parts of the vector in detail and other less important parts as Kronecker products of component vectors, we increase the accuracy of the method by putting more time and memory effort into the computation of the vectors. We propose the approach for a hierarchical Kronecker representation of the generator matrix [3, 5, 10], but it can also be applied for other representations, like matrix diagrams (as used in [13]).

The paper is organized as follows. In the next section, some basic definitions and transient analysis of CTMCs are briefly introduced. Afterwards, the model class and the underlying structured representation of the state space and generator matrix are introduced. Section 4 presents the compositional representation of distribution vectors and the basic operations involved in the realization of a transient analysis using this vector representation. The following section extends the approach to an adaptive version. Finally, the quality of the approximation method is shown by means of some examples. In Section 7 some general remarks about the quality of the approximation are given.

## 2. Transient Analysis

We consider a finite CTMC with state space $\mathcal{RS} = \{0, \ldots, n-1\}$, generator matrix $\mathbf{Q}$, initial distribution $\mathbf{p}_0$, and reward vector $\mathbf{r}$. Let $\mathbf{p}_t$ be the transient distribution at time $t$, which is computed as

$$\mathbf{p}_t = \mathbf{p}_0 e^{\mathbf{Q}t} = \sum_{k=0}^{\infty} \mathbf{p}_0 \mathbf{P}^k \beta(\alpha t, k) \tag{1}$$

where $\alpha = \max_x(|\mathbf{Q}(x,x)|)$, $\mathbf{P} = \mathbf{Q}/\alpha + \mathbf{I}$ and $\beta(\alpha t, k) = e^{-\alpha t}(\alpha t)^k/k!$ is the probability that a Poisson process with rate $\alpha t$ will make $k$ jumps in the interval $(0, 1]$. The latter representation of $\mathbf{p}_t$ as an infinite sum is called the *randomization method* (e.g., [24]) and is the most popular method for transient analysis of CTMCs. After a finite truncation of the infinite sum, upper and lower bounds for the probabilities $\mathbf{p}_t(x)$ can be easily computed using standard means. The instantaneous reward at time $t$ is then given by

$$E[R_t] = \mathbf{p}_0 \left( \sum_{k=0}^{\infty} \mathbf{P}^k \beta(\alpha t, k) \right) \mathbf{r} \,. \tag{2}$$

Bounds after a finite truncation of the infinite sum are computed as

$$\mathbf{p}_0 \left( \sum_{k=0}^{K} \mathbf{P}^k \beta(\alpha t, k) \right) \mathbf{r} \le E[R_t] \le \mathbf{p}_0 \left( \sum_{k=0}^{K} \mathbf{P}^k \beta(\alpha t, k) \right) \mathbf{r} + (1 - \sum_{k=0}^{K} \beta(\alpha t, k)) \max_x(\mathbf{r}(x)) \tag{3}$$

The distribution of instantaneous reward can also be computed as

$$F[R_t \le X] = \sum_{x=0}^{n-1} \delta(\mathbf{r}(x) \le X) \mathbf{p}_t(x) \tag{4}$$

where $\delta(b) = 1$ for $b = true$ and 0 otherwise. Bounds on the distribution can be computed from the bounds on $\mathbf{p}_t$.

In addition to the instantaneous reward, the reward accumulated during the interval $[0, t)$ is of interest. The expectation of accumulated reward can be computed as (see [16])

$$E[AR_t] = \int_0^t \mathbf{p}_\tau \mathbf{r} d\tau = \mathbf{p}_0 \left( \sum_{k=0}^{\infty} \mathbf{P}^k \frac{1 - \beta(\alpha t, k)}{\alpha} \right) \mathbf{r} \,. \tag{5}$$

Similarly, bounds on $E[AR_t]$ can be computed from a finite truncation of the sum.

## 3. A Class of Markovian Models

The basic approach for transient analysis of CTMCs starts with the flat matrix $\mathbf{Q}$ without imposing any structure. However, most CTMCs resulting from models in reliability or performance analysis are specified using some high-level specification like queuing networks, stochastic Petri nets, or networks of communicating automata, and have significant structure. If this structure can be exploited for solution purposes, then usually analysis becomes more efficient.

In the numerical analysis of CTMCs, the structure of the model can be introduced by representing matrix $\mathbf{Q}$ in a structured form as a sum of Kronecker products of small component matrices. Components result from the specification of the model. The approach, which dates back to the work of Plateau [23], has been extended in different directions, and several very sophisticated data structures to represent extremely large matrices in a compact form have been proposed [5, 12]. The compact matrix representation can then be used in iterative solution techniques, including randomization. However, although the approach significantly reduces the memory required for the matrix, it does not eliminate state space explosion, since vectors of size $n$ still have to be stored.

We consider models which are decomposed into $J$ components numbered 1 through $J$. Super- or subscripts $i, j \in \{1, \ldots, J\}$ are used to indicate components. Components result from the specification of the model using some high-level formalism. Examples of such components include subnets of a queuing network or Petri net or an automaton in an

automata network. Let $\mathcal{RS}_i = \{0, \ldots, n_i\}$ be the set of states of component $i$. In the simplest form of the compositional representation, every reachable state in a component can be combined with every reachable state in any other component, and $\mathcal{RS} = \times_{i=1}^{J} \mathcal{RS}_i$. Since the state space is described by the states of components, each state has a $J$-dimensional description as a vector $\mathbf{x}$ such that $\mathbf{x}(i)$ is the state of component $i$ in global state $\mathbf{x}$. This $J$-dimensional representation can be linearized via $x = \sum_{i=1}^{J} \mathbf{x}(i) \cdot n_{i+1}^{J}$ where $n_j^i = \prod_{k=j}^{i} n_k$ and empty products are 1.

It has also been noticed that the basic approach that uses the cross product of the component state spaces to represent $\mathcal{RS}$ often does not work because the potential state space $\mathcal{PS} = \times_{j=1}^{J} \mathcal{RS}_i$ is significantly larger than the reachable state space $\mathcal{RS}$. An example that explains this behavior is the model of a single class closed queuing network that is cut into two submodels. If the system contains $N$ customers, then each subset $\mathcal{RS}_i$ ($i = 1, 2$) contains all states with populations 0 through $N$. Set $\mathcal{PS}$ contains all combinations of those states. However, only those states are reachable in which the first component contains $M$ ($M \leq N$) customers and the second component contains the remaining $N - M$ customers. This idea of defining subsets of the component state spaces and combining these subsets at some higher level naturally defines a hierarchical specification of $\mathcal{RS}$. Hierarchical representations can either be directly derived from the model specification [3, 10] or be computed at the state level [5].

For a hierarchical specification, assume that each subset $\mathcal{RS}_i$ is decomposed into $\tilde{n}_i$ disjoint subsets and that $\mathcal{RS}_i[x]$ denotes subset number $x$. Furthermore $\mathcal{RS}$ is decomposed into $\tilde{n}$ subsets and each subset is characterized by a $J$-dimensional vector $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}}(i) \in \{0, \ldots, \tilde{n}_i - 1\}$ and

$$\mathcal{RS}[\tilde{\mathbf{x}}] = \times_{i=1}^{J} \mathcal{RS}_i[\tilde{\mathbf{x}}(i)] \; . \tag{6}$$

$\widetilde{\mathcal{RS}}$ denotes the set of subsets of $\mathcal{RS}$, and each subset is characterized by a vector $\tilde{\mathbf{x}}$. The complete state space is then represented as

$$\mathcal{RS} = \bigcup_{\tilde{\mathbf{x}} \in \widetilde{\mathcal{RS}}} \times_{i=1}^{J} \mathcal{RS}_i[\tilde{\mathbf{x}}(i)] \; . \tag{7}$$

According to this structure of the state space, matrix $\mathbf{Q}$ is block-structured into $\tilde{n}^2$ submatrices $\mathbf{Q}[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}]$, which contain all transitions between states in $\mathcal{RS}[\tilde{\mathbf{x}}]$ and $\mathcal{RS}[\tilde{\mathbf{y}}]$.

We assume that the components interact via events or transitions from a set $\mathcal{T}_S$. Furthermore, a component has some local behavior described by local transitions. Let $\mathbf{Q}_l^{(i)}$ be the matrix of local transition rates of component $i$. $\mathbf{Q}_l^{(i)}$ is the generator matrix of a CTMC with state space $\mathcal{RS}_i$. For each component $i$ and synchronized transition $t \in \mathcal{T}_S$ let $\mathbf{E}_t^{(i)}$ be a matrix containing the transitions due to the occurrence of $t$ in the state space of $\mathcal{RS}_i$. Thus, $\mathbf{E}_t^{(i)}(x, y)$ is the weight of transition $t$ between local states $x$ and $y$ of component $i$. If component $i$ is not involved in the occurrence of $t$, then $\mathbf{E}_t^{(i)} = \mathbf{I}_{n_i}$, the identity matrix of order $n_i$. The rate of transition $t \in \mathcal{T}_S$ starting in state $\mathbf{x}$ ending in state $\mathbf{y}$ is given by $\lambda_t \prod_{j=1}^{J} \mathbf{E}_t^{(i)}(\mathbf{x}(j), \mathbf{y}(j))$. We assume that rates of transitions $t \in \mathcal{T}_S$ are scaled such that $\max_{x \in \mathcal{RS}_i}(\sum_{y \in \mathcal{RS}_i} \mathbf{E}_t^{(i)}(x, y)) = 1.0$. In other words, matrices $\mathbf{E}_t^{(i)}$ have row sums between 0 and 1. Each block of matrix $\mathbf{Q}$ can now be represented as

$$\mathbf{Q}[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}] = \sum_{i=1}^{J} \mathbf{Q}_{l_i}[\mathbf{x}, \mathbf{y}] + \sum_{t \in \mathcal{T}_S} \mathbf{Q}_t[\mathbf{x}, \mathbf{y}] \tag{8}$$

where $\mathbf{Q}_{l_i}[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}] =$

$$\begin{cases} \mathbf{I}_{n_1^{i-1}} \bigotimes \mathbf{Q}_l^{(i)}[\tilde{\mathbf{x}}(i), \tilde{\mathbf{y}}(i)] \bigotimes \mathbf{I}_{n_{i+1}^{J}} & \text{if } \tilde{\mathbf{y}} \in \mathcal{N}(\tilde{\mathbf{x}}) \; , \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{9}$$

$\mathbf{e}_i$ is a vector with 1 in position 1 and 0 elsewhere, $\otimes$ is the Kronecker product and $\mathcal{N}(\tilde{\mathbf{x}}) = \{\mathbf{y} | \mathbf{y} = \mathbf{x} \vee \exists i : |\tilde{\mathbf{x}} - \tilde{\mathbf{y}}| \leq (\tilde{\mathbf{x}}(i) - \tilde{\mathbf{y}}(i))\mathbf{e}_i\}$. Similarly, the matrices due to synchronized transitions can be represented as $\mathbf{Q}_t[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}] =$

$$\lambda_t (\bigotimes_{i=1}^{J} \mathbf{E}_t^{(i)}[\tilde{\mathbf{x}}(i), \tilde{\mathbf{y}}(i)] - \delta(\tilde{\mathbf{x}} = \tilde{\mathbf{y}}) \bigotimes_{i=1}^{J} \mathbf{D}_t^{(i)}[\tilde{\mathbf{x}}(i), \tilde{\mathbf{y}}(i)]) \tag{10}$$

where $\mathbf{D}_t^{(i)} = diag(\mathbf{E}_t^{(i)} \mathbf{e}^T)$, a diagonal matrix with the row sums of $\mathbf{E}_t^{(i)}$ in the main diagonal and $\lambda_t$ is the basic rate of transition $t$.

The above representation is very compact since all matrices are of dimension $n_i \times n_i$ rather than $n \times n$. The initial vector can also be composed from initial vectors of the components. Let $\mathbf{p}_0^{(i)}$ be the initial vector of component $i$ which can be decomposed into subvectors according to the decomposition of $\mathcal{RS}_i$ and let $\mathbf{p}_0^{(0)}$ be the initial distribution of the macro states. The initial distribution of the complete model is then given as

$$\mathbf{p}_0[\tilde{\mathbf{x}}] = \mathbf{p}_0^{(0)}(\tilde{\mathbf{x}}) \bigotimes_{i=1}^{J} \mathbf{p}_0^{(i)}[\tilde{\mathbf{x}}(i)]. \tag{11}$$

Similarly, let $\mathbf{r}^{(i)}$ be the reward vector for component $i$ and $\mathbf{r}^{(0)}$ be the reward vector of the macro states then,

$$\mathbf{r}[\tilde{\mathbf{x}}] = \mathbf{r}^{(0)}(\tilde{\mathbf{x}}) \bigoplus_{i=1}^{J} \mathbf{r}^{(i)}[\tilde{\mathbf{x}}(i)] \text{ or } \mathbf{r}[\tilde{\mathbf{x}}] = \mathbf{r}^{(0)}(\tilde{\mathbf{x}}) \bigotimes_{i=1}^{J} \mathbf{r}^{(i)}[\tilde{\mathbf{x}}(i)] \tag{12}$$

is the reward vector of the complete system. Whether the reward vector is built by the Kronecker sum or product depends on the type of reward we consider.

The basic step for randomization is the computation of the iteration $\mathbf{p}^k = \mathbf{p}^{k-1} + \mathbf{p}^{k-1}\mathbf{Q}/\alpha$ where $\mathbf{p}^0 = \mathbf{p}_0$ and the vector matrix product computation is realized as a repeated multiplication of a vector (or subvector) with a Kronecker product of matrices at the submatrix or block level of $\mathbf{Q}$ [3, 6, 21].

## 4. Compact Representations of Vectors

The memory and time requirements of a numerical solution will be drastically reduced if the iteration vector, like the generator matrix, can be represented as a Kronecker product of component vectors such that the storage requirement for the vector and the computational effort for an iteration step are in $O(\sum_{i=1}^{J} n_i)$. Unfortunately, such a representation for the stationary or transient distribution does not exist for most models. Nevertheless, the idea of computing the solution vector independently for the components is commonly applied in approximate solution techniques based on a fixed-point approach, and has been integrated in the proposed Kronecker representation in [8]. Here we extend the approach to transient analysis by introducing a different representation of the randomized transition matrix and by integration of the new iteration step into randomization. However, the major contribution of the paper is a detailed experimental analysis of the approximation quality of the new transient analysis approach.

We assume that the solution vector can be approximated as a Kronecker product of distribution vectors for the components. Since for the introduction of the following steps it is more convenient to use normalized vectors, we define for a non-negative vector $\mathbf{a}$ the normalized version as $\bar{\mathbf{a}} = \mathbf{a}/(\mathbf{a}\mathbf{e}^T)$. For $\mathbf{a} = \mathbf{0}$ let $\bar{\mathbf{a}} = \mathbf{0}$. Furthermore, let $\Pr(\mathbf{a}) = \mathbf{a}\mathbf{e}^T$, and for notational convenience, let $\Pr_t(\tilde{\mathbf{x}}) = \Pr(\mathbf{p}_t[\tilde{\mathbf{x}}]) = \mathbf{p}_t[\tilde{\mathbf{x}}]\mathbf{e}^T$. If $\bar{\mathbf{a}}$ and $\Pr(\mathbf{a})$ are known, vector $\mathbf{a}$ can be recreated. For the following representation, it should be noted that the decomposition of component vectors considers macro states from $\widetilde{\mathcal{RS}}$ rather than $\widetilde{\mathcal{RS}}_i$. Thus, $\mathbf{p}_t^{(i)}[\tilde{\mathbf{x}}]$ and $\mathbf{p}_t^{(i)}[\tilde{\mathbf{y}}]$ are different vectors for $\tilde{\mathbf{x}} \neq \tilde{\mathbf{y}}$ even if $\tilde{\mathbf{x}}(i) = \tilde{\mathbf{y}}(i)$. The basic assumption is that each transient vector can be represented approximately as a Kronecker product of component vectors

$$\mathbf{p}_t[\tilde{\mathbf{x}}] \approx \Pr_t(\tilde{\mathbf{x}}) \cdot \bigotimes_{i=1}^{J} \mathbf{p}_t^{(i)}[\tilde{\mathbf{x}}] \tag{13}$$

where $\mathbf{p}_t^{(i)} \in \mathbb{R}^{n_i(\tilde{\mathbf{x}}(i))}$. Similarly, the accumulated values can then be approximated as

$$\int_0^T \mathbf{p}_t[\tilde{\mathbf{x}}]dt \approx \int_0^T \Pr_t(\tilde{\mathbf{x}}) \cdot \bigotimes_{i=1}^{J} \mathbf{p}_t^{(i)}[\tilde{\mathbf{x}}]dt\ . \tag{14}$$

A vector representation that represents vectors by a Kronecker product of component vectors (see eq. (13)) is denoted as an *aggregated representation*, in contrast to a *detailed representation* in which each state corresponds to one vector element.

Define for component $i$ $\lambda_{l_i} = \max_{x \in \mathcal{RS}_i}(|\mathbf{Q}_l^{(i)}(x,x)|)$ as the maximum rate of local transitions leaving a state. Let $\Lambda_l = \sum_{i=1}^{J} \lambda_{l_i}$, $\Lambda_t = \sum_{t \in \mathcal{T}_S} \lambda_t$ and $\Lambda = \Lambda_l + \Lambda_t$ be the local, synchronized, and overall transition rate bounds, respectively. With the maximum rates, transition probability matrices of DTMCs can be computed at the component level. For local transitions we have

$$\mathbf{P}_l^{(i)} = \mathbf{Q}_l^{(i)}/\lambda_{l_i} + \mathbf{I} \tag{15}$$

which describes transition probabilities of component $i$.

As for matrix $\mathbf{Q}$ in (8-10), we derive a Kronecker representation for the stochastic transition matrix $\mathbf{P}$ of the randomized Markov chain. Also as for $\mathbf{Q}$, we distinguish local and synchronized transitions, which are both described by stochastic matrices $\mathbf{P}_l$ and $\mathbf{P}_t$. In contrast to the continuous time case, we cannot simply add the matrices, because each involved matrix is a stochastic matrix of a DTMC describing the evolution of the process under the condition that a transition of the specific type has occurred. However, each transition occurs according to a Poisson process with a fixed rate. This implies that the occurrence of an arbitrary transition follows a Poisson process with rate $\Lambda$, and that a transition is with probability $\Lambda_l/\Lambda$ a local transition in some component and with probability $\lambda_t/\Lambda$ a synchronized transition of type $t \in \mathcal{T}_S$. As is usual in randomization, in both cases pseudo-transition may occur. Local transitions can be collected in one stochastic matrix $\mathbf{P}_l$, and synchronized transitions can be collected in stochastic matrices $\mathbf{P}_t$. Matrix $\mathbf{P}$ can be represented as

$$\mathbf{P} = \sum_{i=1}^{J} \frac{\lambda_{l_i}}{\Lambda} \mathbf{P}_{l_i} + \sum_{t \in \mathcal{T}_S} \frac{\lambda_t}{\Lambda} \mathbf{P}_t\ . \tag{16}$$

The behavior of component $i$ according to local transitions can be described as follows. Local transitions in component $i$ occur according to a Poisson process with rate $\lambda_{l_i}$, and if a local transition occurs in component $i$, then the state changes according to the transitions of the DTMC with transition probability matrix $\mathbf{P}_{l_i}$. This corresponds to a local state change in component $i$ as described by matrix $\mathbf{P}_l^{(i)}$. As usual in randomization, $\mathbf{P}_l^{(i)}(x,x) > 0$ implies a pseudo-transition starting and ending in state $x$. This gives the following Kronecker representation for matrix $\mathbf{P}_{l_i}$, which corresponds to $\mathbf{Q}_{l_i}/\Lambda_{l_i} + \mathbf{I}$.

$$\mathbf{P}_{l_i}[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}] = \begin{cases} \mathbf{I}_{n_1^{i-1}} \bigotimes \mathbf{P}_l^{(i)}[\tilde{\mathbf{x}}(i), \tilde{\mathbf{y}}(i)] \bigotimes \mathbf{I}_{n_{i+1}^J} & \text{if } \tilde{\mathbf{y}} \in \mathcal{N}(\tilde{\mathbf{x}}), \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{17}$$

The derivation of matrix $\mathbf{P}_t$ is slightly more complex, since a pseudo-transition in one component means that the synchronized transition is not enabled and does not occur. However, if the transition does not occur, then the state of all

components remains unchanged. Thus we obtain

$$\mathbf{P}_t[\tilde{\mathbf{x}},\tilde{\mathbf{y}}] = \underbrace{\bigotimes_{i=1}^{J}\mathbf{E}_t^{(i)}[\tilde{\mathbf{x}}(i),\tilde{\mathbf{y}}(i)]}_{\text{synchronized-transition :=}\mathbf{P}_t^{+}[\tilde{\mathbf{x}},\tilde{\mathbf{y}}]} + \underbrace{\delta(\tilde{\mathbf{x}}=\tilde{\mathbf{y}})\left(\bigotimes_{i=1}^{J}\mathbf{I}_{n_i} - \bigotimes_{i=1}^{J}\mathbf{D}_t^{(i)}[\tilde{\mathbf{x}}(i),\tilde{\mathbf{y}}(i)]\right)}_{\text{pseudo-transition :=}\mathbf{P}_t^{-}[\tilde{\mathbf{x}},\tilde{\mathbf{y}}]} \tag{18}$$

The first Kronecker product describes the occurrence of the synchronized transition, whereas the second and third Kronecker products describe a pseudo-transition. Obviously, $\mathbf{P}_t$ is a stochastic matrix. To distinguish pseudo from real transitions, we define $\mathbf{P}_t^{+}$ as the matrix resulting from the first part of the sum in (18) and $\mathbf{P}_t^{-}$ as the matrix resulting from the second part of the sum and $\mathbf{P}_t = \mathbf{P}_t^{+} + \mathbf{P}_t^{-}$.

We now consider the multiplication of $\mathbf{P}_{l_i}$ and $\mathbf{P}_t$ with a Kronecker product of vectors as used for the aggregated representation of vectors. Thus, assume that

$$\mathbf{p}[\tilde{\mathbf{x}}] = \bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]$$

$$\text{then } \left(\bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]\right)\mathbf{P}_{l_i}[\tilde{\mathbf{x}},\tilde{\mathbf{y}}] = \begin{cases} \left(\bigotimes_{j=1}^{i-1}\mathbf{p}^{(j)}[\tilde{\mathbf{x}}]\right)\otimes\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]\mathbf{P}_l^{(i)}[\tilde{\mathbf{x}}(i),\tilde{\mathbf{y}}(i)]\otimes \\ \qquad\qquad \left(\bigotimes_{j=i+1}^{J}\mathbf{p}^{(j)}[\tilde{\mathbf{x}}]\right) \\ \qquad\qquad \text{if } \tilde{\mathbf{y}}\in\mathcal{N}(\tilde{\mathbf{x}}), \\ \mathbf{0} \qquad\qquad \text{otherwise.} \end{cases} \tag{19}$$

The computation follows from the properties of the Kronecker product [24] and implicitly shows that the multiplication of a Kronecker product of vectors with a Kronecker sum of $J$ matrices yields as a result a sum of $J$ Kronecker products of vectors. The situation is slightly more complex for Kronecker products, because pseudo-transitions have to be considered explicitly. If a pseudo-transition occurs in one component that is involved in a synchronization, then the synchronized transition is blocked for all others. Thus, the vector after the synchronized transition occurred is given by

$$\begin{aligned} &\left(\bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]\right)\mathbf{P}_t[\tilde{\mathbf{x}},\tilde{\mathbf{y}}] = \\ &\left(\bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]\right)\mathbf{E}_t^{(i)}[\tilde{\mathbf{x}}(i),\tilde{\mathbf{y}}(i)] + \\ &\delta(\tilde{\mathbf{x}}=\tilde{\mathbf{y}})\left(\bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}] - \left(\bigotimes_{i=1}^{J}\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]\mathbf{D}_t^{(i)}[\tilde{\mathbf{x}}(i),\tilde{\mathbf{y}}(i)]\right)\right) \end{aligned} \tag{20}$$

The resulting vector can be represented by three Kronecker products. Unfortunately, the last Kronecker product is subtracted from the first two such that we have to work with negative and positive vectors, even if all elements of the resulting vector are non-negative [8].

Since (20) contains addition and subtraction of Kronecker products of vectors, we have to represent both operations. We start with the addition operation, which represents a sum of $K$ Kronecker products of vectors with a single Kronecker product. We assume that all vectors are non-negative and have norm 1. Furthermore $\psi_k$ is the non-negative weight of the $k$-th vector.

$$\begin{aligned} &\sum_{k=1}^{K}\psi_k\bigotimes_{i=1}^{J}\mathbf{a}^{(i,k)} \approx \\ &\left(\sum_{k=1}^{K}\psi_k\right)\left(\bigotimes_{i=1}^{J}\left(\sum_{k=1}^{k}\frac{\psi_k}{\sum_{k=1}^{K}\psi_k}\mathbf{a}^{(i,k)}\right)\right) \end{aligned} \tag{21}$$

The resulting vector can afterwards be renormalized by an appropriate choice of the weight such that all vectors have norm 1. The representation of the difference of two Kronecker products of vectors is defined under some restrictions. Again, we assume that all vectors are non-negative and have norm 1. Then we obtain

$$\begin{aligned} &\psi_1\bigotimes_{i=1}^{J}\mathbf{a}^{(i)} - \psi_2\bigotimes_{i=1}^{J}\mathbf{b}^{(i)} \approx \\ &\tfrac{1}{\psi_1-\psi_2}\bigotimes_{i=1}^{J}\left(\psi_1\mathbf{a}^{(i)} - \psi_2\mathbf{b}^{(i)}\right) = \psi_3\bigotimes_{i=1}^{J}\mathbf{c}^{(i)} \end{aligned} \tag{22}$$

The operation is defined only for $\psi_1 > \psi_2$ and non-negative values in the resulting vectors. Thus, $\mathbf{c}^{(i)} \geq \mathbf{0}$ has to hold. If the requirement holds, vector $\mathbf{c}^{(i)}$ can be renormalized to get a representation using vectors with unit norm. As shown in [8], these conditions are observed if the operation is applied to (20).

With the operations (21) and (22), a sum of Kronecker products of vectors can be represented with a single Kronecker product. Obviously, this representation is an approximation, but as shown in [8] for stationary analysis, the approximation error is often small. An iteration step to compute $\mathbf{p}^k[\tilde{\mathbf{x}}]$ from $\mathbf{p}^{k-1}[\tilde{\mathbf{y}}]$ using the iterations defined in (19) and (20) is followed by an approximation (21) and (22). With these steps the randomization approach can be realized and only vectors of length $O(n_i)$ are required. In each step, for each component $i$ and each subset $\tilde{\mathbf{x}}$, a vector $\mathbf{p}^{(i)}[\tilde{\mathbf{x}}]$ of length $O(n_i)$ has to be multiplied by up to $2|\mathcal{T}_S|+1$ matrices, and the resulting vectors have to be combined using (21) and (22).

## 5. Mixing Vector Representations

As shown in [8], it is easy to mix a detailed representation for some vectors $\mathbf{p}^k[\tilde{\mathbf{x}}]$ and an aggregated representation of some other vectors $\mathbf{p}^k[\tilde{\mathbf{y}}]$. Detailed and aggregated representations can be easily transformed. A detailed representation is computed from an aggregated representation using the following relation.

$$\mathbf{p}^k[\tilde{\mathbf{x}}] = \overset{(k)}{\Pr}(\tilde{\mathbf{x}})\cdot\bigotimes_{i=1}^{J}\mathbf{p}^{(i),k}[\tilde{\mathbf{x}}] \tag{23}$$

Thus, if $\mathbf{p}^k[\tilde{\mathbf{x}}]$ has an aggregated representation and $\mathbf{p}^k[\tilde{\mathbf{y}}]$ has a detailed representation, then $\mathbf{p}^k[\tilde{\mathbf{x}}]\mathbf{P}[\tilde{\mathbf{x}},\tilde{\mathbf{y}}]$ is computed, resulting in an aggregated representation of the vector, which can then be transformed into a detailed representation using (23). Similarly, a detailed representation can be transformed into an aggregated version using the following relation.

$$\begin{aligned}&\Pr^{(k)}(\tilde{\mathbf{x}}) = \mathbf{p}^k[\tilde{\mathbf{x}}]\mathbf{e}^T \text{ and}\\ &\mathbf{p}^{(i),k}[\tilde{\mathbf{x}}] = \mathbf{p}^k[\tilde{\mathbf{x}}] \cdot \left(\left(\mathbf{e}_{n_1^{i-1}(\tilde{\mathbf{x}})}\right)^T \bigotimes \mathbf{I}_{n_i[\tilde{\mathbf{x}}(i)]} \bigotimes \left(\mathbf{e}_{n_{i+1}^J(\tilde{\mathbf{x}})}\right)^T\right) / \left(\mathbf{p}^k[\tilde{\mathbf{x}}]\mathbf{e}^T\right)\end{aligned} \quad (24)$$

where $\mathbf{e}_{n_1^{i-1}(\tilde{\mathbf{x}})}$ is the unit vector of length $n_1^{i-1}(\tilde{\mathbf{x}}(i))$. If $\mathbf{p}^k[\tilde{\mathbf{x}}]$ has a detailed representation and $\mathbf{p}^k[\tilde{\mathbf{y}}]$ has an aggregated representation, then $\mathbf{p}^k[\tilde{\mathbf{x}}]\mathbf{P}[\tilde{\mathbf{x}},\tilde{\mathbf{y}}]$ is computed, resulting in a detailed representation of the vector, which can then be transformed into an aggregated representation using (24).

An algorithm exploiting different representations either defines a priori the macro states that are represented in detail or computes those macro states during iteration. In the latter case, it is best to use a detailed representation for the states with the highest probabilities $\Pr(\tilde{\mathbf{x}})$. Since probabilities change during the iteration, the sets of states to be represented in detail are recomputed after each iteration, and the representation changes according to the flow of probability.

## 6. Examples

We analyze the quality of the approximation approach by means of several examples, most are taken from the literature. The state spaces of all examples are of a relatively small size such that exact values can be computed and the quality of the approximation can be experimentally tested. However, the run times of the approximate technique show that the approach can be applied to much larger models. All experiments were performed on a PC with a $2.4$ MHz processor and $512$ MB of main memory. Programs were written in C, and run under Linux. In all cases the stopping criterion for the iteration resulting from the Poisson probabilities [24] equals $\epsilon = 10^{-8}$. We used this criterion for the exact and approximate cases, to compare the results, although a larger $\epsilon$, such as $10^{-6}$, would usually be sufficient for approximate analysis.

### 6.1. Courier Protocol

The Courier protocol is an example that was originally proposed in [25] and has been used as a benchmark example several times. We consider a version with window size $3$ that results in a state space with $419,400$ states. Transition rates and the initial state are chosen as in the original model. As performance measures, we consider the throughput of packets at time $t \in \{0.004, 0.02, 0.1\}$ and in steady state. Furthermore, the average throughput in the interval $[0,t)$ with $t$ chosen as before is presented as an accumulated measure. For the transient measures, randomization is used to compute the result, and for stationary analysis a multi-level solution method as recently proposed in [7] is used. For approximate analysis we use the method introduced in this paper for the transient cases and the method of [8] for steady state analysis. In both cases all vectors are represented in an aggregate form such that it is necessary to store only $905$ probabilities instead of nearly half a million probabilities. Observe that transition rates of the model are in the range of $10^4$, so that more than $5,000$ iterations are necessary to determine results for $t = 0.1$ with the randomization approach.

Table 1 shows the results of the first example. The first column includes the value of $t$ and $t = \infty$ equals steady state analysis. The second and third column contain the solution time (CPU time) in seconds for the exact solution via randomization or the multi-level approach and the approximate technique, respectively. The approximate solution technique is in all cases significantly faster than the exact numerical analysis. The smallest difference between exact and approximate analysis occurs in the steady state case, because for steady state analysis very efficient solvers are available, whereas the approximation approach for steady state analysis uses the Power method and requires much more iterations. For approximate transient analysis, the same number of iterations must be performed, but each iteration is much faster. The results are shown in columns 4-6 and 7-9 of the table. The quality of the results is excellent, since all errors are in the range of 1%. Thus, it is not necessary to mix aggregated and detailed representation to improve the result.

### 6.2. An Availability Model

The second model is a simple example of a fault-tolerant system for which the availability has to be determined. The system consists of 3 computer systems, each with 2 processors, 2 buses, and 4 memory modules. The processors, buses, and memory modules have an exponentially distributed time to failure with failure rates $0.0005$, $0.0004$ and $0.0001$, respectively. We assume that one processor acts as a cold spare and that only active processors can fail. Buses and memory modules are all active and might fail. A computer is available if at least one processor, one bus and one memory module are available. Initially all parts are available. We assume that each failed part is repaired by a single repairman and that repairs are prioritized with preemptive priorities. Repair times are exponentially distributed with a rate of $1.0$ for all components. Parts of the first computer system have a higher priority than components of the second system, which have a higher priority than parts of the third system. Among the parts of each system, the CPUs have the highest priority followed by the buses and the memory modules.

| t | Solution Time exact | approx. | Instantaneous exact | approx. | error | Accumulated exact | approx. | error |
|---|---|---|---|---|---|---|---|---|
| 0.004 | 113 | 0.3 | 6505 | 6493 | -0.2% | 359.0 | 353.2 | -1.6% |
| 0.020 | 480 | 0.6 | 8649 | 8532 | -1.4% | 740.5 | 733.3 | -1.0% |
| 0.100 | 2133 | 2.1 | 8649 | 8533 | -1.3% | 8400 | 8293 | -1.3% |
| $\infty$ | 32 | 0.8 | 8649 | 8533 | -1.3% | 8649 | 8533 | -1.3% |

**Table 1. Throughput of the Courier protocol.**

The system can be described by three components, one for each computer system. Each component has $45$ states, and the complete reachable state space contains $91,125$ states. For the approximate analysis, only three vectors of length $45$ are required to store all state probabilities. Table 2 contains the results and runtimes of the exact transient analysis using randomization and of the approximate approach for the average availability in the interval $[0, t)$. The table contains results only for the components 2 and 3, since the approximate technique yields exact results for component 1 due to the preemptive repair priority, which assures that repairs for component 1 are independent of the state of other components. As in the previous example, the approximation results are excellent; all errors are in the range of $1\%$, and the approximation is much faster than the exact analysis.

### 6.3. A Multi-Server Multi-Queue System

The next example we consider is a multi-server multi-queue system that is often used as a benchmark and was originally described as a Generalized Stochastic Petri Net in [1]. The system describes a token ring with several tokens on it. We consider a configuration with 2 tokens and 5 queues each with a capacity of 5. Service times, switching times, and interarrival times are exponentially distributed with rates $\mu = 1$, $\omega = 10.0$, and $\lambda_i$ for queue $i$. We assume that all queues are initially empty and that both tokens are on their way from the first to the second queue. As result measures, we consider the mean population and the blocking probability (i.e., the probability that the queue is full) of the first queue at time $t \in \{1, 10, 50, 100, 500\}$. We analyze three different configurations of the model. In all configurations $\lambda_1 = 0.4$. In the lightly loaded version, $\lambda_{2-5} = 0.05$; in the medium loaded version, $\lambda_{2-5} = 0.2$; and in the heavily loaded configuration, $\lambda_{2-5} = 0.4$.

For the exact analysis of the system, it is necessary to analyze a Markov chain with $358,560$ states. The approximate method with all states represented in aggregated form requires vectors of length $595$. Results of the example are shown in Table 3. The approximation results are not as good as in the previous examples. Particularly for the blocking probability, errors may become large. For this configuration, the representation of some macro states in detail does not really improve the situation significantly because probabilities are distributed evenly and a subset of more important states cannot be found.

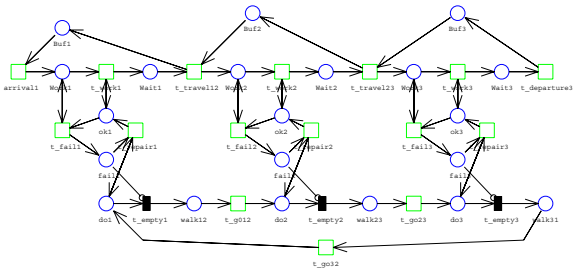


**Figure 1. Kanban example model.**

### 6.4. A Kanban System with Unreliable Machines

The following example is a Kanban system with 3 stages, each containing a single machine. Working machines may fail, and failed machines are repaired by a single repairman who visits and cyclically checks all machines. A Petri net model of the system is shown in Figure 1. The resulting Markov chain has $2,302,911$ states and is stiff, since service rates are in $O(1)$ and failure rates are in $O(0.0001)$. We assume that initially all buffers are filled and all machines are working. Results for the model are shown in Table 4. The exact analysis requires a long runtime, because the state space is large and the Markov chain is stiff. Thus, computation of the result for $t = 1000$ requires more than twelve hours. The approximate technique requires for the same parameters only a few seconds and results are excellent.

### 6.5. An Overflow Queuing System

As our last example, we analyze an overflow queueing system as shown in Figure 2. The system consists of two queues with switched Poisson input process and exponentially service time distributions. Both queues have a finite capacity of $100$ and arriving customers that are blocked by one queue, are routed immediately to the other queue. We analyze the model for the following parameters: $\omega_1 = 0.1$, $\omega_2 = 0.001$, $\nu_1 = 0.01$, $\nu_2 = 0.0001$, $\lambda_1 = \lambda_2 = 5.0$ and $\mu_1 = \mu_2 = 1.0$. The whole system can be described by a CTMC with $40,804$ states and a single macro state is

| t | Component 2 exact | approx. | error | Component 3 exact | approx. | error | Solution Time exact | approx |
|---|---|---|---|---|---|---|---|---|
| 100 | 0.99940 | 0.99938 | 0.0% | 0.99938 | 0.99935 | 0.0% | 1.25 | 0.05 |
| 1000 | 0.99188 | 0.99353 | +0.2% | 0.98727 | 0.99102 | +0.4% | 4.14 | 0.06 |
| 10000 | 0.98709 | 0.99115 | +0.4% | 0.97167 | 0.98610 | +1.5% | 21.5 | 0.06 |
| $\infty$ | 0.98653 | 0.99088 | +0.4% | 0.96955 | 0.98554 | +1.6% | 112 | 0.05 |

**Table 2. Average availability of the computer system in the interval $[0, t)$.**

| t | Population 1 exact | approx. | error | Blocking 1 exact | approx. | error | Solution Time exact | approx. |
|---|---|---|---|---|---|---|---|---|
| Light load | | | | | | | | |
| 1 | 0.3225 | 0.3225 | 0.0% | 4.031e-5 | 4.035e-5 | +0.1% | 4.02 | 0.04 |
| 10 | 0.7806 | 0.7551 | -3.2% | 7.655e-3 | 6.870e-3 | -10.3% | 21.7 | 0.10 |
| 50 | 0.8211 | 0.7819 | -4.8% | 9.610e-3 | 8.073e-3 | -16.0% | 89.1 | 0.31 |
| 100 | 0.8211 | 0.7819 | -4.8% | 9.610e-3 | 8.073e-3 | -16.0% | 168.1 | 0.56 |
| Medium load | | | | | | | | |
| 1 | 0.3268 | 0.3266 | -0.1% | 4.125e-5 | 4.138e-5 | +0.3% | 4.22 | 0.04 |
| 10 | 1.062 | 0.9101 | -14.3% | 1.732e-2 | 9.331e-3 | -46.1% | 23.1 | 0.10 |
| 50 | 1.492 | 1.045 | -25.1% | 5.080e-2 | 1.460e-2 | -71.3% | 93.3 | 0.31 |
| 100 | 1.505 | 1.045 | -30.6% | 5.202e-2 | 1.461e-2 | -71.9% | 176.1 | 0.57 |
| $\infty$ | 1.505 | 1.045 | -30.6% | 5.204e-2 | 1.461e-2 | -71.9% | 11.5 | 0.49 |
| Heavy load | | | | | | | | |
| 1 | 0.3331 | 0.3327 | -0.1% | 4.266e-5 | 4.301e-1 | 0.8% | 4.17 | 0.03 |
| 10 | 1.512 | 1.438 | -4.9% | 3.835e-2 | 2.821e-2 | -26.4% | 23.1 | 0.09 |
| 50 | 3.029 | 3.040 | +0.4% | 2.089e-1 | 2.043e-1 | -2.2% | 94.1 | 0.31 |
| 100 | 3.172 | 3.190 | +0.6% | 2.281e-1 | 2.263e-1 | -0.8% | 175.8 | 0.59 |
| 500 | 3.179 | 3.196 | +0.5% | 2.290e-1 | 2.273e-1 | -0.7% | 813.0 | 2.76 |

**Table 3. Results of the MSMQ system.**

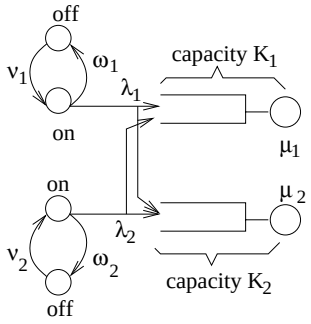


**Figure 2. Overflow queueing system.**

sufficient, since the potential state space equals the reachable state space. For the approximate analysis, only vectors of length $202$ are required. Table 5 contains as result measures the average population and the average probability of a completely filled buffer, for the first queue in the interval $[0, t]$, and for different values of $t$. Initially, the buffers of both queues are empty and both arrival processes are in the *on*-state. For increasing values of $t$, the approximation errors first increase and after $t = 10^3$, the errors slowly decrease. However for $t = 10^3$ or $t = 10^4$ relative errors are still in the range of $30\%$.

For this example, results can be improved by considering some states in detail. In particular those state should be analyzed in detail, for which the queues interact, that means that at least one queue is full. Consequently, we introduce artifically a partition of the state space of each queue, by combining states with population $0$ through $95$ in the first macro state, and combining states with population $96$ through $100$ in the second macro state. For the approximation, state vectors are represented in aggregated form whenever both queues are in the first macro state, and they are represented in detail otherwise. Altogether $4344$ state probabilities have to be stored in this case, compared to $40804$ for an exact solution, and $404$ for an aggregated solution. The results of this aggregation are also shown in Table 5. The results are excellent, since errors almost disappear. However, the solution time is increased too, since the approximate solution is only three times faster than the exact solution. This difference might be improved by a more efficient implementation of the approximation approach.

| t | Throughput exact | approx. | error | Unavailability exact | approx. | error | Solution Time exact | approx. |
|---|---|---|---|---|---|---|---|---|
| 10 | 0.9950 | 0.9950 | 0.0% | 1.052e-9 | 1.052e-9 | 0.0% | 622.4 | 1.04 |
| 100 | 0.9516 | 0.9516 | 0.0% | 1.882e-5 | 1.882e-5 | 0.0% | 4960 | 1.96 |
| 1000 | 0.6322 | 0.6322 | 0.0% | 2.169e-2 | 2.168e-2 | 0.0% | 46184 | 10.6 |

**Table 4. Results for the Kanban system with unreliable machines.**

| t | Population 1 exact | approx. | error | Full 1 exact | approx. | error | Solution Time exact | approx. |
|---|---|---|---|---|---|---|---|---|
| All states aggregated | | | | | | | | |
| 1 | 2.060 | 2.060 | 0.0% | 0.000e+0 | 0.00e+0 | 0.0% | 0.42 | 0.01 |
| 10 | 14.22 | 14.22 | 0.0% | 1.282e-14 | 1.283e-14 | 0.0% | 2.12 | 0.03 |
| 100 | 68.75 | 70.44 | +2.5% | 3.714e-1 | 3.962e-1 | +6.7% | 15.4 | 0.11 |
| 1000 | 69.83 | 96.17 | +37.7% | 4.429e-1 | 5.751e-1 | +29.8% | 134.0 | 0.96 |
| 10000 | 31.43 | 43.04 | +36.9% | 1.316e-1 | 9.661e-2 | -26.6% | 1295 | 9.21 |
| States with population $> 95$ in one queue not aggregated | | | | | | | | |
| 1 | 2.060 | 2.060 | 0.0% | 0.000e+0 | 0.00e+0 | 0.0% | 0.42 | 0.13 |
| 10 | 14.22 | 14.22 | 0.0% | 1.282e-14 | 1.283e-14 | 0.0% | 2.12 | 0.50 |
| 100 | 68.75 | 68.80 | 0.0% | 3.714e-1 | 3.707e-1 | -0.2% | 15.4 | 5.04 |
| 1000 | 69.83 | 69.91 | +0.1% | 4.429e-1 | 4.425e-1 | 0.0% | 134.0 | 44.9 |
| 10000 | 31.43 | 31.45 | 0.0% | 1.316e-1 | 1.316e-1 | 0.0% | 1295 | 384.1 |

**Table 5. Results of the overflow QN example.**

## 7. On the Quality of the Approximation

As for almost all approximation methods, the approximation error of the proposed method cannot be estimated a priori. Unfortunately, approximation errors depend on many factors, and are hard to estimate if the exact solution is not known. However, the errors of the proposed approach for transient analysis are similar to the errors that occur if a fixed-point approach is applied for steady state analysis. Furthermore, the steady state approximation results that are computed here with the method from [8] are similar to those computed with other fixed-point approaches, such as [13, 14, 20]. Thus, a great deal of experience with approximate fixed-point analysis can be taken into account in consideration of models that may introduce large approximation errors.

We know that the method presented here yields exact results for systems with uncoupled components. Thus, if the set $\mathcal{T}_S$ is empty, exact transient probabilities are computed. If the coupling is loose, we can expect good approximations. On the other hand, if the coupling is very tight and synchronized transitions are enabled most of the time, the approximation results are also very good. Another class of models for which we can expect good approximations are models for which state probabilities are concentrated on a small subset of states (as in most reliability models). If this subset is represented in detail, approximation errors become very small. Observe that the method automatically chooses the states with the highest probability of being represented in detail.

If the model does not fall into one of the mentioned classes, the quality of the approximation is unclear. Experience shows that measures like throughput are more robust than measures like populations or blocking probabilities. The MSMQ example indicates the problems and limits of the approach and confirms the general observations. The largest errors can be observed in the medium loaded situation, but even in this case the error for the population is about $30\%$, which is not good, but better than nothing. However, we showed that the errors that can be observed in the MSMQ example are similar to the errors of fixed-point approaches for stationary analysis, which are widely used. Unfortunately, most papers presenting new approximation techniques include only examples for which the proposed approach works well, but do show the limits of the approximation. Nevertheless, for really large models with several hundreds of millions or billions of states, one has the choice between approximation and simulation. The simulative analysis of measures like transient blocking probabilities is often cumbersome and not really reliable, and the approximation approach becomes an efficient alternative to simulation. In the medium-sized examples we presented in this paper, the approximation method is several orders of magnitude faster than numerical analysis and requires only a little memory. Thus, the method can be applied whenever the generator matrix of a model can be represented in compact form.

In addition to developing approximation techniques, we also investigated bounding techniques for transient measures. Most available bounding techniques work only for stationary analysis [22] or can only be used for specific matrix structures [11]. Our first results in computing transient bounds are very promising [15, 19], although the computation of bounds is usually much more expensive than the computation of approximations, and bounds might become loose for some models.

## 8. Conclusions

We present in this paper a new approach for the approximate transient analysis of Markov chains that result from models that consist of interacting components. The algorithm works on small component state spaces only and avoids the state space explosion problem. By means of several examples we have shown that the approach yields good results for many models, and that the solution effort is very small. The approach is much more flexible than most other approximation techniques, since it is possible to improve the quality of the results by spending more effort in terms of memory and solution time.

**Acknowledgements:** We would also like to thank Ms. Jenny Applequist for her editorial assistance.

## References

[1] M. Ajmone-Marsan, S. Donatelli, and F. Neri. GSPN models of Markovian multiserver multiqueue systems. *Performance Evaluation*, 11:227–240, 1990.

[2] A. Bobbio and K. S. Trivedi. An aggregation technique for the transient analysis of stiff Markov chains. *IEEE Trans. on Computers*, 35(9):803–814, 1986.

[3] P. Buchholz. Numerical solution methods based on structured descriptions of Markovian models. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation - Modelling Techniques and Tools*, pages 251–267. Elsevier, 1992.

[4] P. Buchholz. Iterative decomposition and aggregation of labelled GSPNs. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998*, pages 226–245. Springer LNCS 1420, 1998.

[5] P. Buchholz. Hierarchical structuring of superposed GSPNs. *IEEE Trans. on Software Engineering*, 25(2):166–181, 1999.

[6] P. Buchholz. Structured analysis approaches for large Markov chains. *Applied Numerical Mathematics*, 31(4):375–404, 1999.

[7] P. Buchholz. Multi-level solutions for structured Markov chains. *SIAM Journal on Matrix Methods and Applications*, 22(2):342–357, 2000.

[8] P. Buchholz. An adaptive decomposition approach for the analysis of stochastic Petri nets. *Performance Evaluation*, 56(1-4):23–52, 2004.

[9] P. Buchholz and P. Kemper. Compact representations of probability distributions in the analysis of superposed GSPNs. In R. German and B. Haverkort, editors, *Proc. 9th Int. Work. on Petri Nets and Performance Models*, pages 81–90. IEEE CS Press, 2001.

[10] J. Campos, M. Silva, and S. Donatelli. Structured solution of asynchronously communicating stochastic modules. *IEEE Trans. on Software Engineering*, 25(2):147–165, 1999.

[11] J. Carrasco. Computationally efficient and numerically stable realiability bounds for repairable fault-tolerant systems. *IEEE Trans. on Software Engineering*, 51(3):254–268, 2002.

[12] G. Ciardo and A. S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz and M. Silva, editors, *Proc. 8th Int. Work. on Petri Nets and Performance Models*, pages 22–31. IEEE CS Press, 1999.

[13] G. Ciardo, A. S. Miner, and S. Donatelli. Using the exact state space of a model to compute approximate stationary measures. In J. Kurose and P. Nain, editors, *Proc. ACM Sigmetrics*, pages 207–216. ACM Press, 2000.

[14] G. Ciardo and K. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18:37–59, 1994.

[15] D. Daly, P. Buchholz, and W. H. Sanders. An approach for bounding dependability measures of Markov models. Submitted for publication.

[16] E. de Souza e Silva and H. R. Gail. The uniformization method in performability analysis. In B. R. Haverkort, R. Marie, G. Rubino, and K. Trivedi, editors, *Performability Modelling*, pages 31–58. Wiley, 2001.

[17] H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, 56:23–67, 2003.

[18] P. Kemper. Transient analysis of superposed GSPNs. *IEEE Trans. on Software Engineering*, 25(2):182–193, 1999.

[19] V. V. Lam, P. Buchholz, and W. H. Sanders. A structured path-based approach for computing transient rewards of large Markov chains. Accepted for QEST'04 (this volume).

[20] Y. Li and C. M. Woodside. Complete decomposition of stochastic Petri nets. *IEEE Trans. on Computers*, 44(4):577–592, 1995.

[21] A. S. Miner. Computing response time distributions using stochastic Petri nets and matrix diagrams. In *Proc. 10th Int. Work. on Petri Nets and Performance Models*, pages 10–19. IEEE CS Press, 2003.

[22] R. R. Muntz and J. Lui. Computing bounds on steady-state availability of repairable computer systems. *Journal of the ACM*, 41(4):676–707, 1994.

[23] B. Plateau. On the stochastic structure of parallelism and synchronisation models for distributed algorithms. *ACM Performance Evaluation Review*, 13:142–154, 1985.

[24] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

[25] C. M. Woodside and Y. Li. Performance Petri net analysis of communications protocol software by delay equivalent aggregation. In *Proc. 4th Int. Workshop on Petri Nets and Performance Models*, pages 64–73. IEEE CS Press, 1991.