Memory Efficient Calculation of Path Probabilities in Large Structured Markov Chains

Paolo Ballarini
Centre for Computational and Systems Biology
Università di Trento
Trento, Italy
Email: ballarini@cosbi.eu

András Horváth
Dipartimento di Informatica
Università di Torino
Torino, Italy
Email: horvath@di.unito.it

***Abstract*—The problem we deal with is the analysis of a class of large structured Markov chains. In particular we assume that the whole state space can be partitioned into disjoint sets (called macro states) in which the process corresponds to the parallel execution of independent jobs. Petri nets and process algebras with phase type (PH) distributed execution times give rise to this kind of model. These models are subject to the phenomenon of state space explosion. It is known that the infinitesimal generator of such models can be handled in a memory efficient way by storing only the "structure" of the infinitesimal generator as Kronecker expressions or decision diagrams. Less is known instead on how to perform the analysis of the model in a memory efficient manner because in case of most of the available methods the vector of transient or steady state probabilities are stored in an explicit manner.**

In this paper we consider the calculation of measures connected to the probability that the process passes through a given series of macrostates. We show that such measures can be calculated in a memory efficient manner by Laplace transform techniques. The method is illustrated by numerical examples.

I. Introduction

Model based, stochastic performance evaluation of distributed systems is usually based on a high level description. Modelling formalisms for the descriptions of the model under study are, for example, stochastic Petri nets [1] or stochastic process algebra [23], [6], [17], [21]. In order to obtain a tractable stochastic model, it is usual to assume that all durations of the system are exponentially distributed. In this case the underlying model is a continuous time Markov chain (CTMC). In many situations, however, the exponential distribution is not a satisfactory approximation of the durations of the real system. In these cases it is possible to make the model more realistic by approximating the actual durations by phase type distributions [24]. For surveys on fitting methods the reader is referred to [20], [18], [5].

It is clear that if many simultaneously active events of the system are modeled by phase type distributions, then the state space of the model explodes. The structure of the CTMC, however, allows for very efficient storage of the state space and the infinitesimal generator through expressions in Kronecker algebra and/or decision diagrams techniques [25], [26], [22], [2]. Solution methods based on these techniques are also available [15], [11], [12], [8], [10]. There remains, however, the problem of storing the vector containing the transient or steady state probabilities of the states. To overcome this problem, in [7] and [13] approximate stationary measures are computed based on aggregation while [9] proposes a compact Kronecker representation for the vector which leads to an approximate solution.

In this paper we consider a class of structured Markov chains. We assume that the state space of the model can be partitioned into macrostates in such a way that

- inside a macrostate the process is described by the parallel execution of independent tasks,
- transition from a macrostate to another happens when a tasks finishes its activity (during the transitions between macrostates active tasks can become inactive and inactive ones can become active).

For what concerns the infinitesimal generator, the description of the process inside a macrostate is given by the Kronecker sum of small matrices while the transition between two macrostates is given by Kronecker products of small matrices. Petri nets with PH distributed firing times [26] and process algebras with PH distributed event durations give rise to a Markov chain of this kind.

We show that in such models measures connected to the probability of passing through a given series of macrostates can be calculated by Laplace transform techniques in a memory efficient manner, i.e., in a way that does not necessitate the storage of a vector with as many entries as many states we have in the series of macrostates. To the best of our knowledge this is the first attempt to tackle the state space explosion problem by Laplace transform techniques.

The rest of paper is organised as follows. In Section II the considered model class is described. In Section III we

provide the details of the calculations. Since the calculations make use of the Jordan normal form of the infinitesimal generator of the PH distributions, in Section IV we present the Jordan normal form for two classes of PH distributions. In Section V we discuss issues concerning the implementation of the algorithm. In Section VI we illustrate the approach by numerical examples. Section VII concludes the paper.

II. Considered model

The considered model consists of N macrostates and describes the interaction of A activities (also called jobs or tasks) denoted by $a_i, 1 \leq i \leq A$.

The duration of the activities are described by PH distributions. PH distributions are given by the distribution of time to absorption in a Markov chain [24]. The number of phases, the row vector of the initial probabilities and the infinitesimal generator of the PH distribution associated to activity a_i is denoted by n_i, $\mathbf{b}_i$ and $\mathbf{T}_i$, respectively. Further, the column vector containing the rates from transient states to the absorbing one is denoted by $\mathbf{f}_i$, i.e., $\mathbf{f}_i = -\mathbf{T}_i\mathbf{e}$ where $\mathbf{e}$ is the column vector of ones. In the following we call $\mathbf{f}_i$ the finishing vector of job a_i. In case of preemption of an activity, the amount of work already done is either lost or maintained. In the second case, at restart of the preempted activity the associated PH distribution has to be restarted in the phase in which it was at the instant of preemption.

Consequently, in a given macrostate a given activity is either

- active, i.e., it is under execution;
- suspended, i.e., its PH distribution will be restarted in the phase in which it was when the preemption occured;
- inactive, i.e., it is neither under execution nor suspended.

The set of active, suspended and inactive activities in macrostate i is denoted by A_i^A, A_i^S and A_i^I, respectively. Change of macrostate occurs when the execution of an activity ends. When execution of a job $a_i \in A_j^A$ ends in macrostate j then the next macrostate is macrostate k with probability $p_i^{(j,k)}$.

In order to describe the Markov chain underlying the process we follow [26]. The infinitesimal generator of the Markov chain has the block structure shown in Table I. The blocks in the diagonal are given by a Kronecker sum of matrices while those off-diagonal by a Kronecker product of matrices. A given block of $\mathbf{Q}$ will be denoted as $\mathbf{Q}^{(i,j)}, 1 \leq i, j \leq N$.

A block in the diagonal describes transitions inside a macrostate:

$$\mathbf{Q}_k^{(j,j)} = \begin{cases} \mathbf{T}_k & \text{if } a_k \in A_j^A \\ \mathbf{0}_{k,k} & \text{if } a_k \in A_j^S \\ 0 & \text{if } a_k \in A_j^I \end{cases} \tag{1}$$

where $\mathbf{0}_{i,i}$ is a matrix of 0s of size $n_i \times n_i$. Cases in (1) are:

- if a job is active, it evolves according to its infinitesimal generator matrix;
- if a job is suspended, then it cannot change phase but it contributes in the state space of the macrostate;
- an inactive job does not contribute to the description of the macrostate (i.e., its contribution to the Kronecker sum is a scalar 0).

An off-diagonal block describes transitions from a macrostate to another. It is built as a sum of Kronecker products of matrices, $\mathbf{Q}_{k,l}^{(i,j)}$, which describes what happens to activity a_k when the process arrives to macrostate j from macrostate i as a consequence of finishing of activity a_l. These matrices are

$$\mathbf{Q}_{k,l}^{(i,j)} = \begin{cases} \mathbf{f}_k & \text{if } a_k \in A_i^A \text{ and } k = l \text{ and } a_k \in A_j^I \\ \mathbf{f}_k\mathbf{b}_k & \text{if } a_k \in A_i^A \text{ and } k = l \text{ and } a_k \notin A_j^I \\ \mathbf{1}_{k,1} & \text{if } a_k \notin A_i^I \text{ and } k \neq l \text{ and } a_k \in A_j^I \\ \mathbf{I}_k & \text{if } a_k \notin A_i^I \text{ and } k \neq l \text{ and } a_k \notin A_j^I \\ \mathbf{b}_k & \text{if } a_k \in A_i^I \text{ and } a_k \notin A_j^I \\ 1 & \text{if } a_k \in A_i^I \text{ and } a_k \in A_j^I \end{cases} \tag{2}$$

with $i \neq j$ and where $\mathbf{I}_k$ denotes an identity matrix of size n_k while $\mathbf{1}_{n,m}$ denotes a matrix of 1s of size $n \times n$. Cases in (2) are

- if a job is active in macrostate i, it ends and it is not active in macrostate j, then its contribution is given by its finishing vector;
- if a job is active in macrostate i, it ends and it is active again in macrostate j, then its contribution is given by the product of its finishing vector and its initial probability vector;
- if a job is not inactive in macrostate i, it does not end and it is inactive in macrostate j, then its contribution is given by a vector 1s, i.e. we can simply "forget" the phase of this job;
- if a job is not inactive in macrostate i, it does not end and it is not inactive in macrostate j, then its contribution is given by an identity matrix, i.e. the phase of this job is maintained in the new macrostate;
- if a job is inactive in macrostate i and it is not inactive in macrostate j, then its starting phase is determined according to its initial probability vector;
- if a job is inactive in macrostate i and it is inactive in macrostate j, then it does not contribute to the block.

III. Computing path probabilities

Given a series of macrostates of length L, $i_1, i_2, \ldots, i_L$, $1 \leq i_j \leq N$, $1 \leq j \leq L$, let us denote by $W_{i_1,i_2,\ldots,i_L}(t)$ the probability that the process, started in macrostate i_1, walks through the series of macrostates arriving in macrostate i_L in less than t time units. Naturally, the process does not follow a given series of macrostates with probability 1. The probability that the process goes through the series without considering time will be denoted by $W_{i_1,i_2,\ldots,i_L}(\infty)$. The derivative of $W_{i_1,i_2,\ldots,i_L}(t)$ according to t, which will be denoted by $w_{i_1,i_2,\ldots,i_L}(t)$, provides the possibly defective probability density function of the time of arriving to macrostate i_L walking through the series $i_1, i_2, \ldots, i_L$. The nth moment of the time needed to go through the series on condition that the model goes through

$$\mathbf{Q} = \begin{vmatrix} \oplus_{i=1}^{A} \mathbf{Q}_i^{(1,1)} & \sum_{j:a_j \in A_1^A} p_j^{(1,2)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(1,2)} & \cdots & \sum_{j:a_j \in A_1^A} p_j^{(1,N)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(1,N)} \\ \sum_{j:a_j \in A_2^A} p_j^{(2,1)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(2,1)} & \oplus_{i=1}^{A} \mathbf{Q}_i^{(2,2)} & \cdots & \sum_{j:a_j \in A_2^A} p_j^{(2,N)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j:a_j \in A_N^A} p_j^{(N,1)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(N,1)} & \sum_{j:a_j \in A_N^A} p_j^{(N,2)} \otimes_{i=1}^{A} \mathbf{Q}_{i,j}^{(N,2)} & \cdots & \oplus_{i=1}^{A} \mathbf{Q}_i^{(N,N)} \end{vmatrix}$$

TABLE I
BLOCK STRUCTURE OF THE INFINITESIMAL GENERATOR OF THE CONSIDERED CLASS OF MARKOV CHAINS

the series will be denoted by $W^{(n)}_{i_1,i_2,\ldots,i_L}$, i.e.,

$$W^{(n)}_{i_1,i_2,\ldots,i_L} = \frac{1}{W_{i_1,i_2,\ldots,i_L}(\infty)} \int_0^\infty t^n w_{i_1,i_2,\ldots,i_L}(t)dt \ .$$

Our aim is to find a memory efficient way for the calculation of the above introduced quantities.

In the following, in Section III-A, we describe how to perform the calculation in a straightforward manner in time domain, based on uniformisation. This way of calculation requires the storage of possibly huge transient probability vectors. Then, in Section III-B, we suggest a transform domain procedure (i.e., based on calculating the Laplace transform of the measure of interest) which can instead be implemented in a memory efficient manner.

A. Calculation in time domain

In order to perform the calculations in time domain, we construct a Markov chain, $\mathcal{M}_{i_1,i_2,\ldots,i_L}$, that corresponds to the series of macrostates $i_1, i_2, \ldots, i_{L-1}$ and contains a single absorbing state in the place of the states of macrostate i_L. The block structure of the infinitesimal generator of $\mathcal{M}_{i_1,i_2,\ldots,i_L}$ is depicted in Table II. Note that a macrostate can be present more than once in the series of macrostates $i_1, i_2, \ldots, i_L$ and, as a consequence, its states can be present more than once in the state space of $\mathcal{M}_{i_1,i_2,\ldots,i_L}$. The initial probability vector of $\mathcal{M}_{i_1,i_2,\ldots,i_L}$ is

$$\mathbf{b}_{i_1,i_2,\ldots,i_L} = \left| \left(\otimes_{i:a_i \notin A^I_{i_1}} \mathbf{b}_i \right) \ \ 0 \ \ldots \ 0 \right|, \tag{3}$$

i.e., the process is started in macrostate i_1 with initial probability vector that corresponds to the combination of the initial probability vectors of the jobs that are not inactive in i_1.

The probability that $\mathcal{M}_{i_1,i_2,\ldots,i_L}$ is in its last state at time t is equal to $W_{i_1,i_2,\ldots,i_L}(t)$, i.e., in order to calculate $W_{i_1,i_2,\ldots,i_L}(t)$ we need to perform transient analysis of $\mathcal{M}_{i_1,i_2,\ldots,i_L}$. The quantities $w_{i_1,i_2,\ldots,i_L}(t)$ and $W^{(n)}_{i_1,i_2,\ldots,i_L}$ can also be computed based on transient analysis. This can be done in time domain by uniformisation. For what concerns memory, the chain itself is "cheap" because it can be stored as Kronecker expressions of small matrices. The transient probability vector instead has to be stored explicitly which requires the storage of a vector with as many entries as many states we have in $\mathcal{M}_{i_1,i_2,\ldots,i_L}$.

B. Calculation in transform domain

The following theorem provides an expression for the Laplace transform of $W_{i_1,i_2,\ldots,i_L}(t)$ in terms of the blocks of the matrix given in Table II.

Theorem 1. *Let us denote the Laplace transform of $W_{i_1,i_2,\ldots,i_L}(t)$ by $W^*_{i_1,i_2,\ldots,i_L}(s)$, i.e.,*

$$W^*_{i_1,i_2,\ldots,i_L}(s) = \int_0^\infty e^{-st} W_{i_1,i_2,\ldots,i_L}(t)dt \ .$$

$W^*_{i_1,i_2,\ldots,i_L}(s)$ can be computed as

$$\begin{aligned} W^*_{i_1,i_2,\ldots,i_L}(s) = s^{-1} & \left(\otimes_{i:a_i \notin A^I_{i_1}} \mathbf{b}_i \right) \\ & \left(s\mathbf{I} - \mathbf{Q}^{(i_1,i_1)} \right)^{-1} \mathbf{Q}^{(i_1,i_2)} \\ & \left(s\mathbf{I} - \mathbf{Q}^{(i_2,i_2)} \right)^{-1} \mathbf{Q}^{(i_2,i_3)} \ldots \\ & \left(s\mathbf{I} - \mathbf{Q}^{(i_{L-1},i_{L-1})} \right)^{-1} \mathbf{Q}^{(i_{L-1},i_L)} \mathbf{e} \ . \end{aligned} \tag{4}$$

Proof: By time domain transient analysis of the Markov chain, $\mathcal{M}_{i_1,i_2,\ldots,i_L}$, we have that $W_{i_1,i_2,\ldots,i_L}(t)$ equals the last entry of the vector

$$\mathbf{b}_{i_1,i_2,\ldots,i_L} \exp(\mathbf{Q}_{i_1,i_2,\ldots,i_L} t) \tag{5}$$

where $\exp(\bullet)$ denotes matrix exponential while $\mathbf{b}_{i_1,i_2,\ldots,i_L}$ and $\mathbf{Q}_{i_1,i_2,\ldots,i_L}$ are as given in (3) and in Table II, respectively. The Laplace transform of (5) is

$$\begin{aligned} & \int_0^\infty e^{-st} \mathbf{b}_{i_1,i_2,\ldots,i_L} \exp(\mathbf{Q}_{i_1,i_2,\ldots,i_L} t)dt = \\ & \mathbf{b}_{i_1,i_2,\ldots,i_L} \left(s\mathbf{I} - \mathbf{Q}_{i_1,i_2,\ldots,i_L} \right)^{-1} \end{aligned} \tag{6}$$

where $\mathbf{I}$ denotes the identity matrix of the proper size. By considering the bi-diagonal block structure of $\mathbf{Q}_{i_1,i_2,\ldots,i_L}$ (shown in Table II), the matrix inversion in (6) can be evaluated. Performing (6) and taking its last entry result in (4). ■

Based on basic properties of the Laplace transform we have that the Laplace transform of $w_{i_1,i_2,\ldots,i_L}(t)$, denoted by $w^*_{i_1,i_2,\ldots,i_L}(s)$, is

$$w^*_{i_1,i_2,\ldots,i_L}(s) = sW^*_{i_1,i_2,\ldots,i_L}(s)$$

and $W_{i_1,i_2,\ldots,i_L}(\infty)$ can be calculated as

$$W_{i_1,i_2,\ldots,i_L}(\infty) = \lim_{s \to 0} w^*_{i_1,i_2,\ldots,i_L}(s) \tag{7}$$

$$\mathbf{Q}_{i_1,i_2,\ldots,i_L} = \begin{vmatrix} \mathbf{Q}^{(i_1,i_1)} & \mathbf{Q}^{(i_1,i_2)} & \mathbf{0} & \cdots & \\ \mathbf{0} & \mathbf{Q}^{(i_2,i_2)} & \mathbf{Q}^{(i_2,i_3)} & \mathbf{0} & \cdots \\ & & \ddots & \ddots & \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{Q}^{(i_{L-1},i_{L-1})} & \mathbf{Q}^{(i_{L-1},i_L)}\mathbf{e} \\ 0 \ \ldots \ 0 & \cdots & \cdots & 0 \ \ldots \ 0 & 0 \end{vmatrix}$$

TABLE II
INFINITESIMAL GENERATOR OF THE MARKOV CHAIN FOR THE COMPUTATION OF THE PROBABILITY OF WALKING THROUGH MACROSTATES $i_1, i_2, \ldots, i_L$

while the moments, $W^{(n)}_{i_1,i_2,\ldots,i_L}$, can be calculated based on the derivative of $w^*_{i_1,i_2,\ldots,i_L}(s)$ as

$$W^{(n)}_{i_1,i_2,\ldots,i_L} = \frac{(-1)^n}{W_{i_1,i_2,\ldots,i_L}(\infty)} \lim_{s\to 0} \frac{d^n w^*_{i_1,i_2,\ldots,i_L}(s)}{ds^n} \ . \tag{8}$$

The following theorem describes the derivative that needs to be evaluated in (8).

Theorem 2. *The nth derivative of $w^*_{i_1,i_2,\ldots,i_L}(s)$ at $s = 0$ is*

$$\lim_{s\to 0} \frac{d^n w^*_{i_1,i_2,\ldots,i_L}(s)}{ds^n} = \left(\otimes_{i:a_i\notin A^I_{i_1}} \mathbf{b}_i\right) \left[\sum_{j_1=1}^{L-1}\sum_{j_2=1}^{L-1}\cdots\sum_{j_n=1}^{L-1}\prod_{j=1}^{L-1}(-1)^{c_{j_1,\ldots,j_n,j}} c_{j_1,\ldots,j_n,j}! \left(-\mathbf{Q}^{(i_j,i_j)}\right)^{-c_{j_1,\ldots,j_n,j}-1} \mathbf{Q}^{(i_j,i_{j+1})}\right]\mathbf{e} \tag{9}$$

where $c_{j_1,\ldots,j_n,j} = \#\{k : j_k = j, 1 \le k \le n\}$, i.e., $c_{j_1,\ldots,j_n,j}$ denotes the number of occurrences of j in the vector $|j_1,\ldots,j_n|$.

Proof: We need to compute the derivative of the expression

$$\left(\otimes_{i:a_i\notin A^I_{i_1}} \mathbf{b}_i\right)\left(s\mathbf{I}-\mathbf{Q}^{(i_1,i_1)}\right)^{-1}\mathbf{Q}^{(i_1,i_2)} \left(s\mathbf{I}-\mathbf{Q}^{(i_2,i_2)}\right)^{-1}\mathbf{Q}^{(i_2,i_3)}\ldots \left(s\mathbf{I}-\mathbf{Q}^{(i_{L-1},i_{L-1})}\right)^{-1}\mathbf{Q}^{(i_{L-1},i_L)}\mathbf{e} \ .$$

The result presented in (9) can be verified based on the fact that

$$\left.\frac{d\left(s\mathbf{I}-\mathbf{Q}\right)^{-1}}{ds^n}\right|_{s=0} = (-1)^n n!\left(s\mathbf{I}-\mathbf{Q}\right)^{-n-1}\Big|_{s=0} = (-1)^n n!(-\mathbf{Q})^{-n-1}$$

(where $|_{s=0}$ denotes the value of the function at $s = 0$) and by applying rules of derivative of a product and taking into account that matrix multiplication is not commutative. ■

In (4) the quantities like

$$\left(s\mathbf{I}-\mathbf{Q}^{(j,j)}\right)^{-1} \tag{10}$$

corresponds to the Laplace domain description of the process inside a given macrostate. The matrix $\mathbf{Q}^{(j,j)}$ is given as the Kronecker sum of the matrices that describe the tasks in macrostate j, i.e., $\mathbf{Q}^{(j,j)} = \oplus_{i=1}^{A}\mathbf{Q}_i^{(j,j)}$. The following theorem provides the possibility of computing the entries of (10) based on computations with the matrices $\mathbf{Q}_i^{(j,j)}$.

Theorem 3. *Given the Jordan normal form of the matrices $\mathbf{Q}_i^{(j,j)}, 1 \le j \le N, 1 \le i \le A$ such that*

$$\mathbf{Q}_i^{(j,j)} = \mathbf{V}_{j,i}\mathbf{J}_{j,i}\mathbf{V}_{j,i}^{-1}$$

the terms in (4) corresponding to sojourns in macrostates can be calculated as

$$\left(s\mathbf{I}-\mathbf{Q}^{(j,j)}\right)^{-1} = \left(s\mathbf{I}-\oplus_{i=1}^{A}\mathbf{Q}_i^{(j,j)}\right)^{-1} = \left(\otimes_{i=1}^{A}\mathbf{V}_{j,i}\right)\left(s\mathbf{I}-\oplus_{i=1}^{A}\mathbf{J}_{j,i}\right)^{-1}\left(\otimes_{i=1}^{A}\mathbf{V}_{j,i}^{-1}\right) \ . \tag{11}$$

Proof: In order to simplify the notation we will omit the reference to the macrostate (index j in (11)). Furthermore, $\mathbf{I}$ will denote the identity matrix whose size corresponds to the size of the macrostate (i.e., the identity matrix in (11)) while the identity matrices with different size will be denoted differently. Applying the Jordan normal form and the definition of Kronecker sum $\mathbf{Q}$ can be written as

$$\begin{aligned}\mathbf{Q} = \oplus_{i=1}^{A}\mathbf{Q}_i = \oplus_{i=1}^{A}\left(\mathbf{V}_i\mathbf{J}_i\mathbf{V}_i^{-1}\right) = \\ \left(\mathbf{V}_1\mathbf{J}_1\mathbf{V}_1^{-1}\right)\otimes\mathbf{I}_2\otimes\cdots\otimes\mathbf{I}_A + \\ \mathbf{I}_1\otimes\left(\mathbf{V}_2\mathbf{J}_2\mathbf{V}_2^{-1}\right)\otimes\mathbf{I}_3\otimes\cdots\otimes\mathbf{I}_A + \\ \mathbf{I}_1\otimes\cdots\otimes\mathbf{I}_{A-1}\otimes\left(\mathbf{V}_A\mathbf{J}_A\mathbf{V}_A^{-1}\right)\end{aligned} \tag{12}$$

which, by writing $\mathbf{I}_i$ as $\mathbf{V}_i\mathbf{I}_i\mathbf{V}_i^{-1}$ and applying compatibility of ordinary and Kronecker product, becomes

$$\begin{aligned}\left(\mathbf{V}_1\mathbf{J}_1\mathbf{V}_1^{-1}\right)\otimes\left(\mathbf{V}_2\mathbf{I}_2\mathbf{V}_2^{-1}\right)\otimes\cdots\otimes\left(\mathbf{V}_A\mathbf{I}_A\mathbf{V}_A^{-1}\right) + \\ \left(\mathbf{V}_1\mathbf{I}_1\mathbf{V}_1^{-1}\right)\otimes\left(\mathbf{V}_2\mathbf{J}_2\mathbf{V}_2^{-1}\right)\otimes\left(\mathbf{V}_3\mathbf{I}_3\mathbf{V}_3^{-1}\right)\otimes\ldots \\ \otimes\left(\mathbf{V}_A\mathbf{I}_A\mathbf{V}_A^{-1}\right) + \cdots + \\ \left(\mathbf{V}_1\mathbf{I}_1\mathbf{V}_1^{-1}\right)\otimes\cdots\otimes\left(\mathbf{V}_{A-1}\mathbf{I}_{A-1}\mathbf{V}_{A-1}^{-1}\right)\otimes\left(\mathbf{V}_A\mathbf{J}_A\mathbf{V}_A^{-1}\right) = \\ \left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(\mathbf{J}_1\otimes\mathbf{I}_2\otimes\cdots\otimes\mathbf{I}_A\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) + \\ \left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(\mathbf{I}_1\otimes\mathbf{J}_2\otimes\mathbf{I}_3\otimes\cdots\otimes\mathbf{I}_A\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) + \cdots + \\ \left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(\mathbf{I}_1\otimes\cdots\otimes\mathbf{I}_{A-1}\otimes\mathbf{J}_A\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right).\end{aligned} \tag{13}$$

The identity matrix, $\mathbf{I}$, can be written as a Kronecker product of identity matrices whose size corresponds to the sizes of

$\mathbf{Q}_i$, $1 \leq i \leq A$, as $\mathbf{I}_1 \otimes \mathbf{I_2} \otimes \cdots \otimes \mathbf{I_A}$. Applying again $\mathbf{I}_i = \mathbf{V}_i\mathbf{I}_i\mathbf{V}_i^{-1}$ and compatibility of ordinary and Kronecker product, $s\mathbf{I}$ can be written as

$$\begin{aligned} &s\mathbf{I} = s\mathbf{I}_1 \otimes \mathbf{I}_2 \otimes \cdots \otimes \mathbf{I}_A = \\ &s\mathbf{V}_1\mathbf{I}_1\mathbf{V}_1^{-1} \otimes \mathbf{V}_2\mathbf{I}_2\mathbf{V}_2^{-1} \otimes \cdots \otimes \mathbf{V}_A\mathbf{I}_A\mathbf{V}_A^{-1} = \\ &s\left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(\mathbf{I}_1 \otimes \mathbf{I}_2 \otimes \cdots \otimes \mathbf{I}_A\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) \ . \end{aligned} \tag{14}$$

By considering (13) and (14), $s\mathbf{I} - \mathbf{Q}$ can be written as

$$\begin{aligned} &\left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(s\mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_A - \mathbf{J}_1 \otimes \mathbf{I}_2 \otimes \cdots \otimes \mathbf{I}_A - \right. \\ &\quad \mathbf{I}_1 \otimes \mathbf{J}_2 \otimes \mathbf{I}_3 \otimes \cdots \otimes \mathbf{I}_A - \cdots - \\ &\quad \left.\mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_{A-1} \otimes \mathbf{J}_A\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) = \\ &\left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(s\mathbf{I} - \oplus_{i=1}^{A}\mathbf{J}_i\right)\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) \ . \end{aligned} \tag{15}$$

For two matrices, $\mathbf{A}$ and $\mathbf{B}$, the inverse of ordinary and Kronecker product have the properties

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad \text{and} \quad (\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$

and hence the inverse of $s\mathbf{I} - \mathbf{Q}$ given in (15) becomes

$$\begin{aligned} &\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right)^{-1}\left(s\mathbf{I} - \oplus_{i=1}^{A}\mathbf{J}_i\right)^{-1}\left(\otimes_{i=1}^{A}\mathbf{V}_i\right)^{-1} = \\ &\left(\otimes_{i=1}^{A}\mathbf{V}_i\right)\left(s\mathbf{I} - \oplus_{i=1}^{A}\mathbf{J}_i\right)^{-1}\left(\otimes_{i=1}^{A}\mathbf{V}_i^{-1}\right) \end{aligned}$$

which concludes the proof. ■

Since the matrices $\mathbf{J}_{j,i}$ are resulting from Jordan decompositions, they are of an almost diagonal special form. Therefore, the entries of

$$\left(s\mathbf{I} - \oplus_{i=1}^{A}\mathbf{J}_{j,i}\right)^{-1}$$

in (11) can be computed on the fly. Consequently, applying (11) to the expression given in (4) (and also the one given in (9) with $s = 0$ in (11)) can be evaluated based on computations with small matrices without the necessity of storing large vectors.

IV. Jordan normal form for special classes of PH distributions

In many applications special classes of PH distributions are used. For this reason, in the following two subsection, we describe the Jordan normal form for Erlang distributions and for acyclic PH distributions.

A. Jordan normal form for Erlang distributions

An Erlang distribution with shape parameter equal to n and rate parameter equal to λ can be represented as a PH distribution with initial probability vector, $\mathbf{b}$, and infinitesimal generator, $\mathbf{T}$, as

$$\mathbf{b} = \left|\begin{array}{cccc|} 1 & 0 & \ldots & 0 \end{array}\right| ,$$

$$\mathbf{T} = \left|\begin{array}{ccccc} -\lambda & \lambda & 0 & \cdots & \\ 0 & -\lambda & \lambda & 0 & \cdots \\ & & \ddots & & \\ & \cdots & 0 & -\lambda & \lambda \\ & & \cdots & 0 & -\lambda \end{array}\right|$$

and the Jordan normal form of the generator is $\mathbf{T} = \mathbf{VJV}^{-1}$ with

$$\mathbf{J} = \left|\begin{array}{ccccc} -\lambda & 1 & 0 & \cdots & \\ 0 & -\lambda & 1 & 0 & \cdots \\ & & \ddots & & \\ & \cdots & 0 & -\lambda & 1 \\ & & \cdots & 0 & -\lambda \end{array}\right| ,$$

$$\mathbf{V} = \left|\begin{array}{ccccc} 1 & 0 & \cdots & & \\ 0 & \lambda^{-1} & 0 & & \cdots \\ & & \ddots & & \\ \cdots & 0 & \lambda^{-n+2} & & 0 \\ & \cdots & 0 & & \lambda^{-n+1} \end{array}\right| , \tag{16}$$

$$\mathbf{V}^{-1} = \left|\begin{array}{ccccc} 1 & 0 & \cdots & & \\ 0 & \lambda^{1} & 0 & & \cdots \\ & & \ddots & & \\ \cdots & 0 & \lambda^{n-2} & & 0 \\ & \cdots & 0 & & \lambda^{n-1} \end{array}\right| .$$

In the following we provide the Laplace domain description of a macrostate in which all the tasks are of Erlang distributed duration. In particular, consider a macrostate in which we have E active Erlang distributed jobs and there are not suspended activities. Let n_i, λ_i and $\mathbf{R}_i$ with $1 \leq i \leq E$ denote the number of phases, the parameter and the infinitesimal generator of the Erlang distributions, respectively. In order to describe the process inside the macrostate in transform domain we need to calculate the entries of the matrix $\left(s\mathbf{I} - \otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$. This matrix is clearly of size $\prod_{i=1}^{E} n_i$. In order to describe it, we refer to its entries by the so-called mixed based numbering scheme, that is, a given row (or column) of the matrix is identified by a vector $|i_1, i_2, \ldots, i_E|$ with $0 \leq i_j \leq n_j - 1, 1 \leq j \leq E$. Precisely, the vector $|i_1, i_2, \ldots, i_E|$ refers to row $i_E + n_E(i_{E-1} + n_{E-1}(i_{E-2} + n_{E-2}(i_{E-3} + \ldots(i_2 + i_1 n_1)\ldots)))+1$. Based on (16) and (11), by rather cumbersome but basic algebra, it can be shown that the entry of $\left(s\mathbf{I} - \otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$ in row $|i_1, i_2, \ldots, i_E|$ and column $|j_1, j_2, \ldots, j_E|$ is

$$\begin{cases} 0 & \\ & \text{if } \exists k : 1 \leq k \leq E \text{ and } i_k > j_k \\ \dfrac{\left(\sum_{k=1}^{E} j_k - i_k\right)! \prod_{k=1}^{E} \lambda_k^{j_k - i_k}}{\left(\prod_{k=1}^{E}(j_k - i_k)!\right)\left(s + \sum_{k=1}^{E}\lambda_k\right)^{1+\sum_{k=1}^{E} j_k - i_k}} & \\ & \text{otherwise.} \end{cases}$$

Entries of the matrix $\left(s\mathbf{I} - \otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$ can be computed on the fly based on the parameters of the Erlang distributions. Moreover, $\left(s\mathbf{I} - \otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$ is strongly structured with a high number of repeated entries which can be exploited in the computations. In order to show this feature, let us consider the case $E = 2$ with $n_1 = 2, n_2 = 3$. In Table III $\left(s\mathbf{I} - \otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$ is given.

B. Jordan normal form for acyclic PH distribution

Acyclic PH distributions are frequently applied in modelling because they have a canonical form with a relatively small

$$\left|\begin{array}{cccccc}
\frac{1}{s+\lambda_1+\lambda_2} & \frac{\lambda_2}{(s+\lambda_1+\lambda_2)^2} & \frac{\lambda_2^2}{(s+\lambda_1+\lambda_2)^3} & \frac{\lambda_1}{(s+\lambda_1+\lambda_2)^2} & \frac{2\lambda_1\lambda_2}{(s+\lambda_1+\lambda_2)^3} & \frac{3\lambda_1\lambda_2^2}{(s+\lambda_1+\lambda_2)^4} \\
0 & \frac{1}{s+\lambda_1+\lambda_2} & \frac{\lambda_2}{(s+\lambda_1+\lambda_2)^2} & 0 & \frac{\lambda_1}{(s+\lambda_1+\lambda_2)^2} & \frac{2\lambda_1\lambda_2}{(s+\lambda_1+\lambda_2)^3} \\
0 & 0 & \frac{1}{s+\lambda_1+\lambda_2} & 0 & 0 & \frac{\lambda_1}{(s+\lambda_1+\lambda_2)^2} \\
0 & 0 & 0 & \frac{1}{s+\lambda_1+\lambda_2} & \frac{\lambda_2}{(s+\lambda_1+\lambda_2)^2} & \frac{\lambda_2^2}{(s+\lambda_1+\lambda_2)^3} \\
0 & 0 & 0 & 0 & \frac{1}{s+\lambda_1+\lambda_2} & \frac{\lambda_2}{(s+\lambda_1+\lambda_2)^2} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{s+\lambda_1+\lambda_2}
\end{array}\right|$$

TABLE III
$\left(s\mathbf{I}-\otimes_{i=1}^{E}\mathbf{R}_i\right)^{-1}$ WITH $E=2$ AND $\mathbf{R}_i$ DESCRIBING ERLANG DISTRIBUTIONS OF SIZE $n_1=2, n_2=3$ WITH PARAMETERS λ_1 AND λ_2

number of parameters [14] which can be exploited in PH fitting problems [4], [3], [19]. In particular, any N-phase APH distribution can be transformed into the form

$$\mathbf{b} = \left|\begin{array}{cccc} b_1 & b_2 & \dots & b_N \end{array}\right|$$
$$\mathbf{T} = \left|\begin{array}{ccccc} -\lambda_1 & \lambda_1 & 0 & \cdots & \\ 0 & -\lambda_2 & \lambda_2 & 0 & \cdots \\ & & \ddots & & \\ & \cdots & 0 & -\lambda_{N-1} & \lambda_{N-1} \\ & & \cdots & 0 & -\lambda_N \end{array}\right| \quad (17)$$

where we must have $\sum_{i=1}^{N} b_i = 1$ and $\lambda_i \leq \lambda_{i+1}, 1 \leq i \leq N-1$. When all λ_is are different, the Jordan decomposition of $\mathbf{T}$ results in $\mathbf{T} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ with

$$\mathbf{J} = \left|\begin{array}{cccc} -\lambda_1 & 0 & \cdots & \\ 0 & -\lambda_2 & 0 & \cdots \\ & & \ddots & \\ \cdots & 0 & -\lambda_{N-1} & 0 \\ & \cdots & 0 & -\lambda_N \end{array}\right|$$

and entries of $\mathbf{V}$ and $\mathbf{V}^{-1}$ given as

$$[\mathbf{V}]_{i,j} = \begin{cases} 0 & \text{if } i > j \\ 1 & \text{if } i = j \\ \prod_{k=0}^{j-i-1} \frac{\lambda_{i+k}}{\lambda_{i+k}-\lambda_j} & \text{otherwise} \end{cases}$$

$$\left[\mathbf{V}^{-1}\right]_{i,j} = \begin{cases} 0 & \text{if } i > j \\ 1 & \text{if } i = j \\ \prod_{k=0}^{j-i-1} (-1)^{i+j} \frac{\lambda_{i+k}}{\lambda_i-\lambda_{i+k+1}} & \text{otherwise.} \end{cases}$$

(When some of the λ_is are equal, matrix $\mathbf{J}$ is not diagonal but, naturally, it is a Jordan diagonal block matrix.) The above expressions clearly suggest the numerical difficulties connected to the Jordan normal form of acyclic PH distributions. The matrix $\mathbf{V}$ can become stiff for relatively small sizes as well. We investigate this issue in practice in Section VI.

V. IMPLEMENTATION ISSUES

The complexity of computing the Jordan normal form depends on the applied algorithm. It is at most $O(n^4)$. Since the PH distributions describing event durations are small and/or structured and we consider large Markov chains, this part of the computation is negligible for what concerns execution times. As it is illustrated in Section VI, the calculation of the Jordan normal form is an ill-conditioned problem if the eigenvalues are close to each other. This problem can be tackled by an implementation that allows for computation with arbitrary precision.

There are several algorithms available to perform Kronecker products and sums, see [10] for a comparison. In order to follow the process inside a macrostate, we have to multiply a vector by an expression of the form of (11). This expression contains two Kronecker products composed of matrices that are either sparse or dense depending on the PH distributions that represent task durations. In case of dense matrices the fastest approach to perform the Kronecker product is the perfect shuffles algorithm [25] while for sparse matrices other algorithms presented in [10] perform better. The remaining part of (11) involves calculations with almost diagonal matrices and has a negligible effect on the execution time as it will be illustrated in Section VI as well.

As described at the end of Section III-B, the computation of the measures connected to a series of macrostates can be performed storing only the matrices that describe the PH distributions of the tasks (including their Jordan normal form). This requires the application of (11) in (4) which requires the implementation of several nested cycles. Proceeding this way the execution times would become intolerable. We suggest instead the storage of one vector initialised as

$$\mathbf{v} = s^{-1}\left(\otimes_{i:a_i \notin A_{i_1}^I} \mathbf{b}_i\right) \quad (18)$$

and perform $L-1$ times

$$\mathbf{v} \leftarrow \mathbf{v}\left(s\mathbf{I} - \mathbf{Q}^{(i_k,i_k)}\right)^{-1} \quad (19)$$

$$\mathbf{v} \leftarrow \mathbf{v}\mathbf{Q}^{(i_k,i_{k+1})} \quad (20)$$

with $1 \leq k \leq L-1$ where (19) is performed in three parts as suggested by (11). Hereinafter, the first and third part of (11) will be referred to as Phase I of the computation, while

the second part of (11) as Phase II. Phase III instead refers to the computation of (20). Finally, the value of (4) is given by $\mathbf{ve}$. Proceeding this way, the peak length of $\mathbf{v}$, i.e., the peak memory consumption of the computation, is given by the number of states of the largest macrostate in the series.

The inversion of the Laplace transform requires the evaluation of (4) for a set of values of s. This requires to perform (18-20) several times. A significant part of the computation for different values of s are common. Therefore, the computations can be sped up but the memory consumption increases since we need to store a vector $\mathbf{v}$ for every values of s. We illustrate this possibility in Section VI.

VI. Numerical experiments

In this section we illustrate the numerical properties and the execution times of the proposed approach.

Instead of considering a model describing a real phenomenon, we apply the calculations to a model that illustrate the numerical properties and the execution times well and whose parameters are easy to change in order to see their effect. Because of its simplicity, it is possible that the model we consider hereinafter can be evaluated with other methods in less time. However, the method we present in this paper can be used to solve more complicated problems and the simple model we use to illustrate its properties has the same (or higher) computational complexity as a more complicated model would have. The experiments were performed on a standard portable computer with processor operated at 2GHz and with 2GB of RAM. Our implementation is written in C.

As example I we consider a series of L macrostates in which in every macrostate there are A running activities. The PH distributions representing the duration of the activities are of the same size (denoted by n), general (i.e., the infinitesimal generators, $\mathbf{T}_i, 1 \leq i \leq A$, are full matrices) and randomly generated. Finishing of the first activity takes the model from a macrostate to the following one. In the following macrostate the activity restarts while the others keep on executing. Passing through the series means that the first activity was executed $L-1$ times before any of the other activities finished their task. The process inside a macrostate is described simply by the Kronecker sum of the generators of the activities, i.e.,

$$\mathbf{Q}^{(i,i)} = \oplus_{k=1}^{A} \mathbf{T}_k.$$

The transition from a macrostate to another is described by

$$\mathbf{Q}^{(i,i+1)} = \mathbf{f}_1 \mathbf{b}_1 \otimes \left(\otimes_{k=2}^{A} \mathbf{I}_k \right), \tag{21}$$

i.e., the first activity terminates and restarts while the others remain in the phase they were. The initial probability vector in the first macrostate is given by

$$\otimes_{k=1}^{A} \mathbf{b}_k.$$

We compute the probability of passing through the series which is given by (7) and (4) as

$$\left(\otimes_{k=1}^{A} \mathbf{b}_k \right) \left(\prod_{i=1}^{L-1} \left(\left(-\mathbf{Q}^{(i,i)} \right)^{-1} \mathbf{Q}^{(i,i+1)} \right) \right) \mathbf{e}. \tag{22}$$

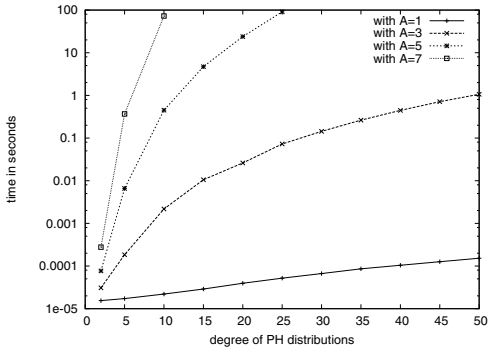

Fig. 1. Time of computation in case of jobs represented by PH distributions of the same size (example I)

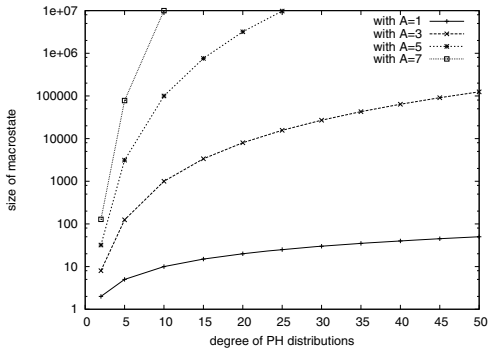

Fig. 2. Size of the macrostate in case of jobs represented by PH distributions of the same size (example I)

By computing the Jordan normal form of the infinitesimal generators of the PH distributions, $\mathbf{T}_j = \mathbf{V}_j \mathbf{J}_j \left(\mathbf{V}_j\right)^{-1}$, according to Theorem 3 we have that

$$\left(-\mathbf{Q}^{(i,i)} \right)^{-1} = \otimes_{j=1}^{A} \mathbf{V}_j \left(-\oplus_{j=1}^{A} \mathbf{J}_j \right)^{-1} \otimes_{j=1}^{A} \left(\mathbf{V}_j\right)^{-1}.$$

In the following we report on execution times for $L = 2$, i.e., we have only one factor in the product in (22). For other values of L, the computation times can simply be achieved by multiplying the reported numbers by $L-1$ (e.g., for $L = 3$ one has to compute the reported numbers by 2). The reported execution times do not include the Jordan decomposition because for large models its effect is negligible. Figure 1 depicts the total time needed to perform (22) for different values of A and different degrees of the PH distributions representing the tasks. On Figure 2 we show the size of one macrostate of the series which is equal to n^A. The size of one macrostate corresponds to the length of the vector that has to be stored in order to perform the computation. Note that the amount of memory needed does not depend on L. The curves finish at a given point for higher values of A because there is not enough RAM in our computer to go further.

For what concerns other measures, computing the nth moment of the time needed to pass through the series is L^n times the time indicated in Figure 1.

In order to stress the strength of the proposed method, let us compare it against the time domain approach. With the time domain approach one should store a vector with as many entries as many states are involved in the series of macrostates.

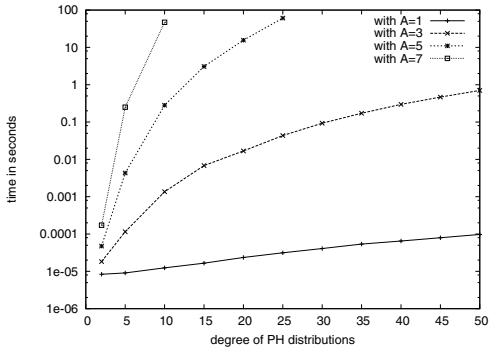

Fig. 3. Time of computation in phase I in case of jobs represented by PH distributions of the same size (example I)

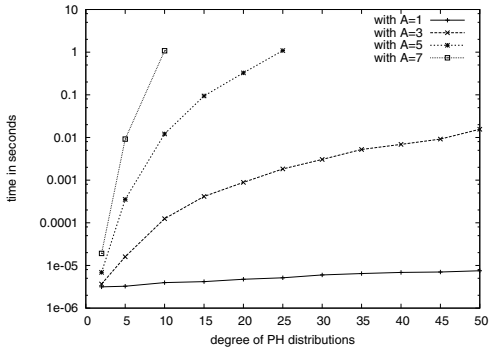

Fig. 4. Time of computation in phase II in case of jobs represented by PH distributions of the same size (example I)

For example I, this means that one needs a vector with Ln^A entries. With our approach we need a vector of size n^A. As a consequence, the probability of performing the series, for example, with $L = 100$, $n = 15$ and $A = 5$ cannot be computed in reasonable amount of time in time domain (it would require storing entries of the transient probability vector on the hard disk) but can be computed with the proposed method in about 150 seconds.

In Figures 3-5 we report on computation times connected to different phases of the computation. Phases I and II refer to the calculations inside a macrostate. In particular, phase I refers to multiplication by $\otimes_{i=1}^{A}\mathbf{V}_{j,i}$ and $\otimes_{i=1}^{A}\left(\mathbf{V}_{j,i}\right)^{-1}$ while phase II to multiplication by $\left(-\oplus_{j=1}^{A}\mathbf{J}_j\right)^{-1}$. Phase III refers instead to the transition from one macrostate to the following one, i.e., to the multiplication by the expression given in (21). One can observe that performing phase II is the lightest part of the computation. The execution time of phase III is approximately the half of the execution time of phase I. This is due to the fact that in our implementation we use the same function to perform phase I and phase III (since both correspond to multiplication by a Kronecker product of matrices). Note, however, that while in phase I the matrices involved in the Kronecker product are dense (they are the result of the Jordan decomposition of $\mathbf{T}_i, 1 \leq i \leq A$), the matrices involved in phase III are not. With an implementation that exploit this characteristics of the computation, phase III could be made drastically faster, speeding up the total computation by about 25%.

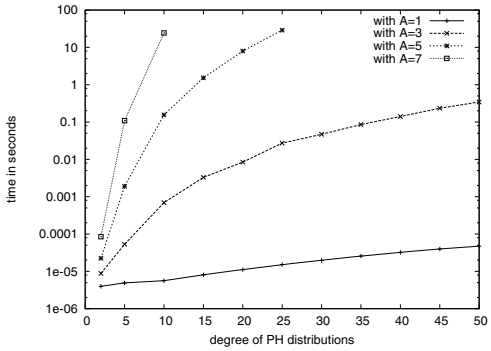

Fig. 5. Time of computation in phase III in case of jobs represented by PH distributions of the same size (example I)

size	A	composition	phase I	phase II	phase III
2^8	4	$2^2 2^2 2^2 2^2$	2.53e-04	2.63e-05	1.13e-04
2^8	2	$2^4 2^4$	3.45e-04	2.88e-05	1.60e-04
2^8	2	$2^2 2^6$	6.70e-04	2.86e-05	3.27e-04
2^8	3	$2^2 2^2 2^4$	2.80e-04	2.89e-05	1.34e-04
2^{12}	6	$2^2 2^2 2^2 2^2 2^2 2^2$	5.70e-03	4.04e-04	2.62e-03
2^{12}	4	$2^3 2^3 2^3 2^3$	5.85e-03	4.08e-04	2.82e-03
2^{12}	3	$2^4 2^4 2^4$	7.69e-03	4.33e-04	3.67e-03
2^{12}	2	$2^6 2^6$	3.10e-02	4.19e-04	8.38e-03
2^{12}	4	$2^2 2^2 2^4 2^4$	6.85e-03	4.30e-04	3.41e-03
2^{12}	4	$2^2 2^2 2^2 2^6$	1.25e-02	4.17e-04	6.41e-03
2^{12}	5	$2^2 2^2 2^2 2^2 2^4$	6.69e-03	4.10e-04	3.03e-03
2^{20}	5	$2^4 2^4 2^4 2^4 2^4$	4.72e+00	1.02e-01	2.33e+00
2^{20}	4	$2^5 2^5 2^5 2^5$	6.14e+00	1.09e-01	3.05e+00
2^{20}	5	$2^2 2^3 2^4 2^5 2^6$	6.27e+00	1.10e-01	3.12e+00
2^{20}	5	$2^3 2^3 2^4 2^5 2^5$	5.39e+00	1.09e-01	2.73e+00

TABLE IV
EXECUTION TIMES OF EXAMPLE II WITH DIFFERENT COMPOSITIONS OF THE CONSIDERED MACROSTATE

As example II we consider the same model but with PH distributions of different size. In Table IV execution times are reported. The first column gives the size of one macrostate, the second the number of active running tasks and the third the sizes of the PH distributions. The remaining three columns give the time needed to perform the different phases of the computation. One can observe that the more activities are present in a macrostate, the faster is the execution of phase I and III. Also, the more homogeneous the sizes of the active tasks are, the faster will phase I and III be performed. Execution time of Phase II instead does not depend on the structure of the macrostate.

In the following we turn back to example I. We illustrate that by applying inverse Laplace transform it is possible to compute the probability that the model passes through the series of macrostates in a given amount of time. In particular, we consider example I with $A = 4$ and $n = 10$ and different values of L. The PH distributions are random but with mean set to 1 for the first activity and to 5 for the remaining three. The results are depicted in Figure 6. Naturally, the probability that the first activity succeeds in finishing it tasks L times before any other activity finishes decreases as L increases.

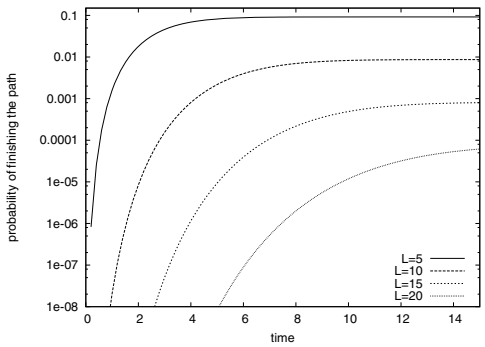

Fig. 6. Probability of performing the path in a given amount of time for example I

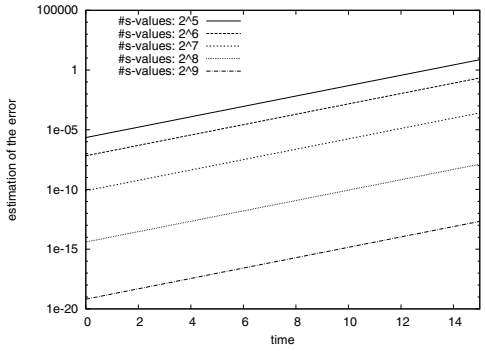

Fig. 7. Estimation of error of the inverse Laplace transform for the probabilities presented in Figure 6

The inverse Laplace transform was performed by applying Weeks' method as provided in [27], [16]. With this method the Laplace transform, $f^*(s)$, has to be evaluated at a set of complex values of s. Then, the function in time domain, $f(t)$, is reconstructed based on these values of $f^*(s)$. It is important that the method uses the same set of complex values of s to determine $f(t)$ for any t, i.e., $f(t)$ can be determined at any number of values of t based on the same set of values of $f^*(s)$. The number of complex values of s has a strong impact on the precision of the inverse transform. The numbers depicted in Figure 6 were generated with 2^9 values of s. Calculating $f^*(s)$ for a given value of s requires the same computational effort as computing the probability of passing through the series (because it is computed at $s = 0$). Calculation of the transient curve in Figure 6 with $L = 5$ takes about 72 seconds when the computations are made in an independent manner for different values of s and requires the storage of a vector of complex numbers of length 10000. As it was discussed briefly in Section V, it is possible to spare computations if the calculations are performed in parallel for every value of s. This requires less time, about 9.5 seconds, but much more memory because we need to store 2^9 vectors of complex numbers of length 10000.

Weeks' method provides the possibility of estimating the error. Figure 7 depicts the estimated error as function of time for different number of values of s for $L = 20$. Other values of L yielded almost precisely the same error estimation.

In order to investigate the numerical problems caused by the necessity of Jordan decomposition, we consider once again example I with $A = 2$ and $L = 2$ and the computation of passing through the series. With Mathematica it is possible to perform the calculations with arbitrary precision (i.e., we can define with how many digits the input parameters are stored and all the calculations are performed according to the chosen precision). Then, having performed the calculation, Mathematica is able to give an estimation of the number of digits that are accurate in the result. We distinguished two cases: PH distributions in canonical form as given in (17) and general PH distributions (with full infinitesimal generator). Moreover, in both categories we generated random infinitesimal generators whose entries (apart from the diagonal) are in a given range. The ranges were $[1, 2]$, $[1, 10]$ and $[1, 100]$. Figure 8 depicts the resulting precisions for PH distributions of size 10 and 25. For general PH distributions neither the range of the parameters nor the size changes the resulting precision. The precision of the output is almost equal to the precision of the input and grows linearly with the precision of the input. On the contrary, for PH distributions in canonical form the precision of the output can be much lower than that of the input and, the narrower the range of the parameters is, the lower is the precision of the output. This is caused by the fact that for PH distributions in canonical form, narrower range of parameters results in an infinitesimal generator whose eigenvalues are close to each other which makes the computation of the Jordan normal form ill-conditioned and the resulting matrices stiff. For example, with 25 phases and parameters in the range $[1, 2]$ we need to work with at least 50 digits precision in order to get a single precise digit in the result. The good news is, however, that precision of the output is growing linearly with precision of the input.

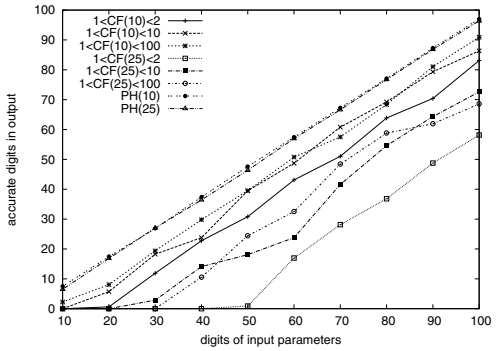

Fig. 8. Estimated number of accurate digits for example I with $A = 2$ and $L = 2$ with PH distributions in canonical form (CF) and general PH distributions (PH)

VII. Conclusion

In this paper we provided a method for the memory efficient calculations of measures connected to paths of macrostates in a class of large structured Markov chains. The technique is based on computing the Laplace transform of the measures of interest. The numerical properties and the execution times of the calculations were discussed based on an illustrative example.

In the future we plan to work on an implementation which allows for control of the precision of the results, to investigate the possibility of computing other measures of interest based on Laplace transform and to apply the technique to real problems.

REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley, 1995.

[2] A. Bobbio and A. Horváth. Petri nets with discrete phase type timing: A bridge between stochastic and functional analysis. In *Proc. of 2nd Workshop on Models for Time-Critical Systems*, volume 52 No. 3 of *Electronic Notes in Theoretical Computer Science*, Aalborg, Denmark, Aug. 2001.

[3] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21:303–326, 2005.

[4] A. Bobbio and M. Telek. A benchmark for PH estimation algorithms: results for Acyclic-PH. *Stochastic Models*, 10:661–677, 1994.

[5] L. Bodrog, A. Heindl, A. Horváth, G. Horváth, and M. Telek. Current results and open questions on PH and MAP characterization. In *Dagstuhl Seminar N.07461 on Numerical Methods for Structured Markov Chains*, 2008.

[6] P. Buchholz. On a Markovian process algebra. Technical report, Technical Report 500, Informatik IV, University of Dortmund, 1994.

[7] P. Buchholz. Iterative decomposition and aggregation of labelled GSPNs. In *Application and Theory of Petri Nets 1998: 19th International Conference, ICATPN'98*, 1998.

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

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

[10] P. Buchholz, G. Ciardo, P. Kemper, and S. Donatelli. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 13(3):203–222, 2000.

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

[12] G. Ciardo and A. S. Miner. A data structure for the efficient kronecker solution of gspns. In *Proc. of the 8th Int. Work. on Petri Nets and Performance Models*, pages 22–31, 1999.

[13] G. Ciardo, A. S. Miner, and S. Donatelli. Using the exact state space of a model to compute approximate stationary measures. In *Proc. ACM Sigmetrics*, 2000.

[14] A. Cumani. On the canonical representation of homogeneous Markov processes modelling failure-time distributions. *Microelectronics and Reliability*, 22:583–602, 1982.

[15] P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor-vector multiplication in stochastic automata networks. *Journal of the ACM*, 45(3):381–414, 1998.

[16] B. S. Garbow, G. Giunta, J. N. Lyness, and A. Murli. Software for an implementation of Weeks' method for the inverse Laplace transform. *ACM Transactions on Mathematical Software (TOMS)*, 14(2):163–170, 1988.

[17] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, Volume 12 of Distinguished Dissertations in Computer Science, Cambridge University Press, 1996.

[18] A. Horváth and M. Telek. Markovian modeling of real data traffic: Heuristic phase type and map fitting of heavy tailed and fractal like samples. In *Tutorial of the IFIP WG 7.3 International Symposium on Computer Performance Modeling, Measurement and Evaluation (PERFORMANCE 2002)*, volume 2459 of *Lecture Notes in Computer Science*, Rome, Italy, Sept 2002.

[19] A. Horváth and M. Telek. Matching more than three moments with acyclic phase type distributions. *Stochastic Models*, 23(2):167–194, 2007.

[20] A. Lang and J. L. Arthur. Parameter approximation for phase-type distributions. In S. R. Chakravarty and A. S. Alfa, editors, *Matrix-analytic methods in stochastic models*, Lecture notes in pure and applied mathematics, pages 151–206. Marcel Dekker, Inc., 1996.

[21] R. Gorrieri. M. Bernardo, L. Donatiello. A formal approach to the integration of performance aspects in the modelling and analysis of concurrent systems. *Information and Computation*, 144:83–154, 1998.

[22] A. S. Miner and G. Ciardo. Efficient reachability set generation and storage using decision diagrams. In *Proceedings of the 20th Int. Conf. on Applications and Theory of Petri Nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 6–25, Williamsburg, VA, USA, 1999.

[23] M. Rettelbach N. Goetz, U. Herzog. Multiprocessor and distributed system design: The integration of functional specication and performance analysis using stochastic process algebras. In *Proceedings of Performance Evaluation of Computer and Communication Systems*, volume 729 of *Lecture Notes in Computer Science*, pages 121–146, 1993.

[24] M. Neuts. Probability distributions of phase type. In *Liber Amicorum Prof. Emeritus H. Florin*, pages 173–206. University of Louvain, 1975.

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

[26] M. Scarpa and A. Bobbio. Kronecker representation of Stochastic Petri nets with discrete PH distributions. In *International Computer Performance and Dependability Symposium - IPDS98*, pages 52–61. IEEE CS Press, 1998.

[27] J. A. C. Weideman. Algorithms for parameter selection in the Weeks method for inverting the Laplace transform. *SIAM Journal on Scientific Computing*, 21(1):111–128, 1999.