Distributed Computation of Transient State Distributions and Passage Time Quantiles in Large Semi-Markov Models

Jeremy T. Bradley, Nicholas J. Dingle, Peter G. Harrison, William J. Knottenbelt

Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ, United Kingdom

Abstract

Semi-Markov processes (SMPs) are expressive tools for modelling parallel and distributed systems; they are a generalisation of Markov processes that allow for arbitrarily distributed sojourn times. This paper presents an iterative technique for transient and passage time analysis of large structurally unrestricted semi-Markov processes. Our method is based on the calculation and subsequent numerical inversion of Laplace transforms and is amenable to a highly scalable distributed implementation. Results for a distributed voting system model with up to 1.1 million states are presented and validated against simulation.

1 Introduction

Traditional techniques for the analytical performance modelling of parallel and distributed systems are predominantly based on the steady-state analysis of Markov chains. This is restrictive for three main reasons. Firstly, the Markov property imposes the (often unrealistic) limitation that all time delays must be exponentially distributed. Secondly, steady-state measures cannot give insight into the transient behaviour of the system before or after critical events, such as failures, reconfigurations and system startup. Thirdly, steady-state measures are adequate to determine mean resourse-based measures and even some mean passage or response time values, but not to determine passage time quantiles.

Email addresses: `jb@doc.ic.ac.uk` (Jeremy T. Bradley), `njd200@doc.ic.ac.uk` (Nicholas J. Dingle), `pgh@doc.ic.ac.uk` (Peter G. Harrison), `wjk@doc.ic.ac.uk` (William J. Knottenbelt).

Preprint submitted to Future Generation Computer Systems *1 June 2004*

This is a serious problem since passage time quantiles are assuming increasing importance as key quality of service and performance metrics.

The aim of the present study is to investigate the use of semi-Markov processes (SMPs) for the purposes of system description, calculation of transient state distributions and computation of passage time densities and quantiles. By using SMPs we can specify more realistic models with generally distributed delays while still maintaining some of the analytical tractability associated with Markovian models.

Our specific contribution is a novel iterative algorithm for large structurally unrestricted SMPs that generates transient state distributions. This builds upon our iterative technique for generating passage time densities and quantiles [1,2]. The algorithm is based on the calculation and subsequent numerical inversion of Laplace transforms. One of the biggest problems involved in working with semi-Markov processes is how to store the Laplace transform of state sojourn time distributions in an effective way, such that accuracy is maintained but representation explosion does not occur. We address this issue with a constant-space representation of a general distribution function based on the evaluation demands of the numerical inversion algorithm employed.

We implement our technique in a scalable, distributed and checkpointed analysis pipeline and apply it to instances of a distributed voting model. The high-level model description is given in the form of a semi-Markov Stochastic Petri net – our own proposal for a non-Markovian Stochastic Petri net formalism – and is textually described in an extended semi-Markovian version of the high-level DNAmaca Markov chain specification language [3]. Our results are validated against a simulation derived from the same high-level model.

The rest of this paper is organised as follows. In Section 2, we briefly detail the background theory behind semi-Markov processes, and show how to derive first passage times and transient state distributions. Our iterative passage time procedure is described in Section 3.2 and the new iterative transient scheme is presented in Section 3.3. Section 4 describes the practical issues in numerically inverting Laplace transforms as well as storing and manipulating general distributions. Section 5 describes the architecture of our distributed implementation. Section 6 briefly introduces the semi-Markov stochastic Petri net formalism and DNAmaca specification system from [1,4]. Passage time and transient results are produced for systems with up to $\sim 10^6$ states which are validated by simulations. Section 7 concludes and considers future work.

2 Definitions and Background Theory

2.1 Semi-Markov Processes

Consider a Markov renewal process $\{(X_n, T_n) : n \geq 0\}$ where T_n is the time of the nth transition ($T_0 = 0$) and $X_n \in \mathcal{S}$ is the state at the nth transition. Let the kernel of this process be:

$$R(n, i, j, t) = \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \tag{1}$$

for $i, j \in \mathcal{S}$. The continuous time semi-Markov process (SMP), $\{Z(t), t \geq 0\}$, defined by the kernel R, is related to the Markov renewal process by:

$$Z(t) = X_{N(t)} \tag{2}$$

where $N(t) = \max\{n : T_n \leq t\}$, i.e. the number of state transitions that have taken place by time t. Thus $Z(t)$ represents the state of the system at time t. We consider time-homogeneous SMPs, in which $R(n, i, j, t)$ is independent of any previous state except the last. Thus R becomes independent of n:

$$\begin{aligned} R(i, j, t) &= \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \quad \text{for any } n \geq 0 \\ &= p_{ij} H_{ij}(t) \end{aligned} \tag{3}$$

where $p_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i)$ is the state transition probability between states i and j and $H_{ij}(t) = \mathbb{P}(T_{n+1} - T_n \leq t \mid X_{n+1} = j, X_n = i)$, is the sojourn time distribution in state i when the next state is j.

2.2 First Passage Times

Consider a finite, irreducible, continuous-time semi-Markov process with N states $\{1, 2, \ldots, N\}$. Recalling that $Z(t)$ denotes the state of the SMP at time $t \geq 0$, the first passage time from a source state i at time t into a non-empty subset of the state space $\vec{j} \subseteq \mathcal{S}$ is:

$$P_{i\vec{j}}(t) = \inf\{u > 0 : Z(t + u) \in \vec{j}, N(t + u) > N(t), Z(t) = i\} \tag{4}$$

Throughout this paper we refer to $\vec{j}$, the set of states that terminate the passage, as the set of target states. For a stationary time-homogeneous SMP, $P_{i\vec{j}}(t)$ is independent of t and we have:

$$P_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j}, N(u) > 0, Z(0) = i\} \tag{5}$$

$P_{i\vec{j}}$ is a random variable with an associated probability density function $f_{i\vec{j}}(t)$

such that the passage time quantile is defined as:

$$\mathbb{P}(t_1 < P_{i\vec{j}} < t_2) = \int_{t_1}^{t_2} f_{i\vec{j}}(t)\,\mathrm{d}t \tag{6}$$

In general, the Laplace transform of $f_{i\vec{j}}$, $L_{i\vec{j}}(s)$, can be computed by solving a set of N linear equations:

$$L_{i\vec{j}}(s) = \sum_{k\notin\vec{j}} r^*_{ik}(s) L_{k\vec{j}}(s) + \sum_{k\in\vec{j}} r^*_{ik}(s) \qquad : \text{for } 1 \leq i \leq N \tag{7}$$

where $r^*_{ik}(s)$ is the Laplace-Stieltjes transform (LST) of $R(i,k,t)$ from Section 2.1 and is defined by:

$$r^*_{ik}(s) = \int_0^\infty e^{-st}\,\mathrm{d}R(i,k,t) \tag{8}$$

Eq. (7) has a matrix–vector form, $A\tilde{x} = \tilde{b}$, where the elements of A are arbitrary complex functions. For example, when $\vec{j} = \{1\}$, Eq. (7) yields:

$$\begin{pmatrix} 1 & -r^*_{12}(s) & \cdots & -r^*_{1N}(s) \\ 0 & 1 - r^*_{22}(s) & \cdots & -r^*_{2N}(s) \\ 0 & -r^*_{32}(s) & \cdots & -r^*_{3N}(s) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -r^*_{N2}(s) & \cdots & 1 - r^*_{NN}(s) \end{pmatrix} \begin{pmatrix} L_{1\vec{j}}(s) \\ L_{2\vec{j}}(s) \\ L_{3\vec{j}}(s) \\ \vdots \\ L_{N\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} r^*_{11}(s) \\ r^*_{21}(s) \\ r^*_{31}(s) \\ \vdots \\ r^*_{N1}(s) \end{pmatrix} \tag{9}$$

When there are multiple source states, denoted by the vector $\vec{i}$, the Laplace transform of the passage time distribution is:

$$L_{\vec{i}\vec{j}}(s) = \sum_{k\in\vec{i}} \alpha_k L_{k\vec{j}}(s) \tag{10}$$

where the weight α_k is the probability of being in state $k \in \vec{i}$ at the starting instant of the passage.

If measuring the system from equilibrium then $\tilde{\alpha}$ is a normalised steady-state vector. That is, if $\tilde{\pi}$ denotes the steady-state vector of the embedded discrete-time Markov chain (DTMC) with one-step transition probability matrix $P = [p_{ij}, 1 \leq i,j \leq N]$, then α_k is given by:

$$\alpha_k = \begin{cases} \pi_k / \sum_{j\in\vec{i}} \pi_j & \text{if } k \in \vec{i} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

2.3 Transient state distributions

Another key modelling result is the transient state distribution, $\pi_{ij}(t)$, of a stochastic process:

$$\pi_{ij}(t) = \mathbb{P}(Z(t) = j \mid Z(0) = i) \tag{12}$$

From Pyke's seminal paper on SMPs [5], we have the following relationship between passage time densities and transient state distributions, in Laplace form:

$$\pi^*_{ij}(s) = \frac{1}{s}\frac{1 - h^*_i(s)}{1 - L_{ii}(s)} \quad \text{if } i = j, \quad \pi^*_{ij}(s) = L_{ij}(s)\pi^*_{jj}(s) \quad \text{if } i \neq j \tag{13}$$

where $\pi^*_{ij}(s)$ is the Laplace transform of $\pi_{ij}(t)$ and $h^*_i(s) = \sum_j r^*_{ij}(s)$ is the LST of the sojourn time distribution in state i. For multiple target states, this becomes:

$$\pi^*_{i\vec{j}}(s) = \sum_{k \in \vec{j}} \pi^*_{ik}(s) \tag{14}$$

However, to construct $\pi^*_{i\vec{j}}(s)$ directly using this translation is computationally expensive: for a vector of target states $\vec{j}$, we need $2|\vec{j}| - 1$ passage time quantities, $L_{ik}(s)$, which in turn require the solution of $|\vec{j}|$ linear systems of the form of Eq. (9).

This motivates our development of a new transient state distribution formula for multiple target states in semi-Markov processes which requires the solution of only one system of linear equations.

From Pyke's formula for a transient state distribution between two states [5, Eq. (3.2)], we can derive:

$$\pi_{ij}(t) = \delta_{ij}\overline{F}_i(t) + \sum_{k=1}^{N} \int_0^t R(i, k, t - \tau)\, \pi_{kj}(\tau)\, \mathrm{d}\tau \tag{15}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, and $\overline{F}_i(t)$ is the reliability function of the sojourn time distribution in state i, i.e. the probability that the system has not left state i after t time units. $R(i, k, t - \tau)$ represents the occurrence of a single transition out of state i to an adjacent state k in time $t - \tau$ and $\pi_{kj}(\tau)$ is the probability of being in state j having left state k after a further time τ. Transforming this convolution into the Laplace domain and generalising to multiple target states, $\vec{j}$, we obtain:

$$\pi^*_{i\vec{j}}(s) = \delta_{i \in \vec{j}}\overline{F}^*_i(s) + \sum_{k=1}^{N} r^*_{ik}(s)\pi^*_{k\vec{j}}(s) \tag{16}$$

Here, $\delta_{i \in \vec{j}} = 1$ if $i \in \vec{j}$ and 0 otherwise. The Laplace transform of the reliability function $\overline{F}^*_i(s)$ is generated from $h^*_i(s)$ as:

$$\overline{F}^*_i(s) = \frac{1 - h^*_i(s)}{s}$$

Eq. (16) has a matrix–vector form as well; for example, when $\vec{j} = \{1, 3\}$, we have:

$$\begin{pmatrix} 1 - r^*_{11}(s) & -r^*_{12}(s) & \cdots & -r^*_{1N}(s) \\ -r^*_{21}(s) & 1 - r^*_{22}(s) & \cdots & -r^*_{2N}(s) \\ -r^*_{31}(s) & -r^*_{32}(s) & \cdots & -r^*_{3N}(s) \\ \vdots & \vdots & \ddots & \vdots \\ -r^*_{N2}(s) & -r^*_{N2}(s) & \cdots & 1 - r^*_{NN}(s) \end{pmatrix} \begin{pmatrix} \pi^*_{1\vec{j}}(s) \\ \pi^*_{2\vec{j}}(s) \\ \pi^*_{3\vec{j}}(s) \\ \vdots \\ \pi^*_{N\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} \overline{F}^*_1(s) \\ 0 \\ \overline{F}^*_3(s) \\ \vdots \\ 0 \end{pmatrix} \tag{17}$$

Again for multiple source states, with initial distribution $\tilde{\alpha}$, the Laplace transform of the transient function is:

$$\pi^*_{\vec{i}\vec{j}}(s) = \sum_{k \in \vec{i}} \alpha_k \pi^*_{k\vec{j}}(s) \tag{18}$$

3 Iterative Passage Time and Transient Analysis

3.1 Introduction

In this section, we describe iterative algorithms for generating passage time densities/quantiles and transient state distributions. The algorithms create successively more accurate approximations to the analytic passage time function given by Eq. (7) and transient function given by Eq. (16), respectively.

3.2 Iterative Passage Time Method

The iterative passage time technique considers the rth transition passage time of the system, $P^{(r)}_{i\vec{j}}$. This is the time for r consecutive transitions to occur, starting from state i and ending in any of the states in $\vec{j}$. The unconditioned passage time density, $P_{i\vec{j}}$, is then obtained in the limit as $r \to \infty$. We calculate $P^{(r)}_{i\vec{j}}$ for a sufficiently high value of r to give an approximation to within a specified degree of accuracy.

Recall the semi-Markov process, $Z(t)$, of Section 2.2, where $N(t)$ is the number of state transitions that have taken place by time t. Formally, we define the

rth transition first passage time to be:

$$P^{(r)}_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j} \mid 0 < N(u) \leq r, Z(0) = i\} \quad (19)$$

which is the time taken to enter a state in $\vec{j}$ for the first time having started in state i at time 0 and having undergone up to r state transitions. $P^{(r)}_{i\vec{j}}$ is a random variable with associated probability density function, $f^{(r)}_{i\vec{j}}(t)$, which has Laplace transform $L^{(r)}_{i\vec{j}}(s)$.

$L^{(r)}_{i\vec{j}}(s)$ is, in turn, the ith component of the vector

$$\tilde{L}^{(r)}_{\vec{j}}(s) = (L^{(r)}_{1\vec{j}}(s), L^{(r)}_{2\vec{j}}(s), \ldots, L^{(r)}_{N\vec{j}}(s))$$

which may be computed as:

$$\tilde{L}^{(r)}_{\vec{j}}(s) = U(I + U' + U'^2 + \cdots + U'^{(r-1)})\,\tilde{e} \quad (20)$$

Here U is a matrix with elements $u_{pq} = r^*_{pq}(s)$ and U' is a modified version of U with elements $u'_{pq} = \delta_{p \notin \vec{j}}\, u_{pq}$, where states in $\vec{j}$ have been made absorbing. The column vector $\tilde{e}$ has entries $e_k = \delta_{k \in \vec{j}}$.

We include the initial U term in Eq. (20) so as to generate cycle times for cases such as $L^{(r)}_{ii}(s)$ which would otherwise register as 0 if U' were used instead.

From Eqs. (5) and (19):

$$P_{i\vec{j}} = P^{(\infty)}_{i\vec{j}} \quad \text{and thus} \quad L_{i\vec{j}}(s) = L^{(\infty)}_{i\vec{j}}(s). \quad (21)$$

Now, $L^{(r)}_{i\vec{j}}(s)$ can be generalised to multiple source states $\vec{i}$ using, for example, the normalised steady-state vector, $\tilde{\alpha}$, of Eq. (11):

$$\begin{aligned} L^{(r)}_{\vec{i}\vec{j}}(s) &= \tilde{\alpha}\tilde{L}^{(r)}_{\vec{j}}(s) \\ &= \textstyle\sum_{k=0}^{r-1} \tilde{\alpha} U U'^k \tilde{e} \end{aligned} \quad (22)$$

The sum of Eq. (22) can be computed efficiently using sparse matrix–vector multiplications with a vector accumulator. At each step, the accumulator (initialised to $\tilde{\alpha}U$) is post-multiplied by U' and $\tilde{\alpha}U$ is added. The worst-case time complexity for this sum is $O(N^2 r)$ versus the $O(N^3)$ of typical matrix inversion techniques. In practice, we typically observe $r << N$ for large N.

Convergence of the sum in Eq. (22) is said to have occurred at a particular r, if for a given s-point:

$$|\mathrm{Re}(L^{(r+1)}_{\vec{i}\vec{j}}(s) - L^{(r)}_{\vec{i}\vec{j}}(s))| < \epsilon \;\text{ and }\; |\mathrm{Im}(L^{(r+1)}_{\vec{i}\vec{j}}(s) - L^{(r)}_{\vec{i}\vec{j}}(s))| < \epsilon \quad (23)$$

where ϵ is chosen to be a suitably small value (e.g. 10^{-8}).

3.3 Iterative Transient Method

Our iterative transient state distribution generation technique builds on the passage time computation technique of the previous section. We aim to calculate $\pi_{i\vec{j}}(t)$, that is the probability of being in any of the states of $\vec{j}$ at time t, having started in state i at time, $t = 0$. We approximate this transient state distribution by constructing $\pi^{(r)}_{i\vec{j}}(s)$, which is the rth iterative approximation to the Laplace Transform of the transient state distribution function.

$\pi^{(r)}_{i\vec{j}}(s)$ is, in turn, the ith component of the vector:

$$\tilde{\pi}^{(r)}_{\vec{j}}(s) = (\pi^{(r)}_{1\vec{j}}(s), \pi^{(r)}_{2\vec{j}}(s), \ldots, \pi^{(r)}_{N\vec{j}}(s))$$

which may be computed as:

$$\tilde{\pi}^{(r)}_{\vec{j}}(s) = (I + U + U^2 + \cdots + U^r)\, \tilde{v} \tag{24}$$

where $\tilde{v}$ is made up of the reliability functions for each of the target states in $\vec{j}$, i.e. $v_i = \delta_{i\in\vec{j}}\overline{F}^*_i(s)$.

Note that, instead of using an absorbing transition matrix as in the passage time scheme, the transient method makes use of the unmodified transition matrix U, which has elements $u_{ij} = r^*_{ij}(s)$. This reflects the fact that the transient state distribution accumulates probability from all the passages through the system and not just the first one.

The astute reader may notice that this method bears a loose resemblance to the well-known *uniformization* technique [6–8] which can be used to generate transient-state distributions and passage time densities for Markov chains. However, as we are working with semi-Markov systems, there can be no *uniformizing* of the general distributions in the SMP. The general distribution information has to be maintained as precisely as possible throughout the process. We achieve this by using the representation technique described in Section 4.

Finally, as before, the technique can be generalised to multiple start states by employing an initial vector, $\tilde{\alpha}$, where α_i is the probability of being in state i at time 0:

$$\pi^{(r)}_{\vec{i}\vec{j}}(s) = \tilde{\alpha}(I + U + U^2 + \cdots + U^r)\, \tilde{v} \tag{25}$$

Fig. 1 shows a transient state distribution, $\pi_{00}(t)$, that is the probability of being in state 0, having started in state 0, at time t. The system being analysed is a simple two state system with an exponential (rate 2) transition from state 0 to state 1, and a deterministic transition (parameter 2) from 1 to 0. The discontinuities in the derivative from the deterministic transition can clearly be made out at points $t = 2, 4$ and in fact also exist at $t = 6, 8, 10, \ldots$. Also shown on the graph are 5 iterations of the algorithm which exhibit increasing accuracy in approximating the transient curve.

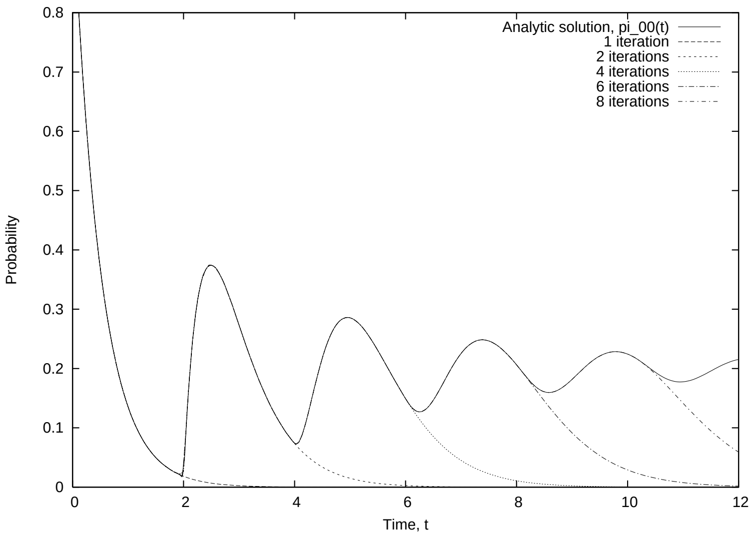

Fig. 1. Example iterations towards a transient state distribution in a system with successive exponential and deterministic transitions

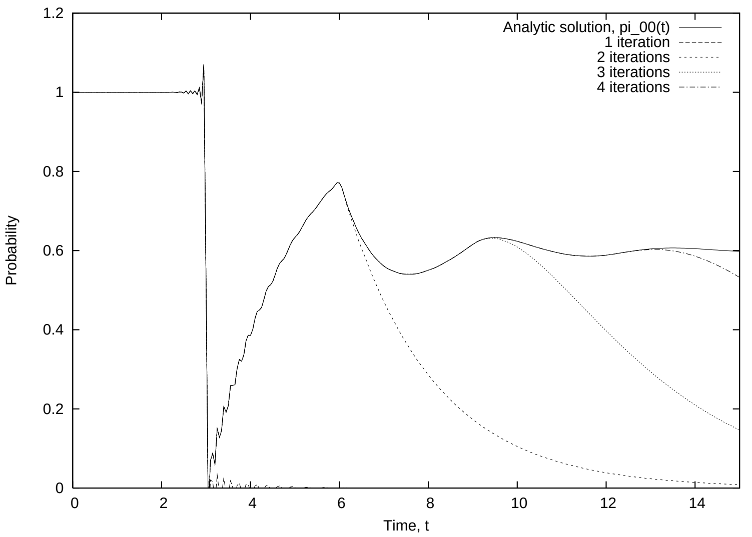

Fig. 2. Example iterations towards a transient state distribution in a system with successive deterministic and exponential transitions

Fig. 2 shows the transient state distribution $\pi_{00}(t)$ for a two state system with a deterministic transition (parameter 3) from state 0 to state 1, and an exponential (rate 0.5) transition from 1 to 0. The graph clearly shows the system remaining in state 0 for the initial 3 time units, as dictated by the out-going deterministic transition. The perturbations in the graph observed around $t = 3$ are generated by small numerical instabilities (Gibb's Phenomena) in the Laplace inversion algorithm [9]. These are most pronounced when an initial deterministic distribution is observed and are, for systems with more smoothing, almost always unobservable. Also shown on the graph are 4 iterations of the algorithm which exhibit increasing accuracy in approximating the transient curve, as before.

4 Laplace Transform Representation and Inversion

The key to the practical analysis of semi-Markov processes lies in the efficient representation of their generally distributed sojourn time distribution functions. Without care the structural complexity of the SMP can be recreated within the representation of the distribution functions. This is especially true with the manipulations performed in the iterative passage time calculation of Section 3.

Many techniques have been used for representing arbitrary distributions – two of the most popular being *phase-type distributions* and *vector-of-moments* methods. These methods suffer from, respectively, exploding representation size under composition and containing insufficient information to produce accurate answers after large amounts of composition.

As all our distribution manipulations take place in Laplace-space, we link our distribution representation to the Laplace inversion technique that we ultimately use. Our implementation supports two Laplace transform inversion algorithms: the Euler technique [10] and the Laguerre method [11] with modifications summarised in [12].

Both algorithms work on the same general principle of sampling the transform function $L(s)$ at n points, $s_1, s_2, \ldots, s_n$ and generating values of $f(t)$ at m user-specified t-points $t_1, t_2, \ldots, t_m$. In the Euler inversion case $n = km$, where k typically varies between 15 and 50, depending on the accuracy of the inversion required. In the modified Laguerre case, $n = 400$ and, crucially, is independent of m.

The choice of inversion algorithm depends on the characteristics of the density function $f(t)$. If the function is continuous, and has continuous derivatives (i.e. it is "smooth") then the Laguerre method can be used. If, however, the density function or its derivatives contain discontinuities – for example if the system exclusively contains transitions with deterministic or uniform holding-time distributions – then the Euler method must be employed.

Whichever inversion algorithm is used, it is important to note that calculating $s_i, 1 \leq i \leq n$ and storing all the distribution transform functions, sampled at these points, will be sufficient to provide a complete inversion. Storing our distribution functions in this way has three main advantages. Firstly, the function has constant storage space, independent of the distribution-type. Secondly, each distribution has, therefore, the same constant storage even after composition with other distributions. Finally, the function has sufficient information about a distribution to determine the required passage time or transient density (and no more).

5 Implementation Architecture

Our implementation employs a distributed master–slave architecture similar to that of the Markovian passage time calculation tool of [12]. The master processor computes in advance the values of s at which it will need to know the value of $L_{\vec{i}\vec{j}}(s)$ in order to perform the inversion. The s-values are then placed in a global work-queue to which the slave processors make requests. On making a request slave processors are assigned the next available s-value and use this to construct the matrices U and U'. The iterative algorithm is then applied to calculate the truncated sum of Eq. (22) or Eq. (25) (as appropriate) for that s-value. The result is returned to the master and cached (both in memory and on disk so that all computation is checkpointed), and once all values have been computed and returned, the final Laplace inversion calculations are made by the master. The resulting t-points can then be plotted on a graph. As inter-slave communication is not required, the algorithm exhibits excellent scalability (see Section 6.4.3).

6 Distributed System Modelling

6.1 Introduction

We demonstrate the SMP analysis techniques of the previous sections with a semi-Markov model of a distributed voting system. As there is a rich tradition of modelling distributed systems with stochastic Petri nets [13,14], we propose and then make use of a semi-Markov extension of GSPNs to generate the model.

6.2 Semi-Markov Stochastic Petri Nets

Semi-Markov stochastic Petri nets (SM-SPNs) are extensions of GSPNs [15], which can handle arbitrary state-dependent sojourn time distributions and which generate an underlying semi-Markov process rather than a Markov process. Formally a SM-SPN consists of a 4-tuple, $(PN, \mathcal{P}, \mathcal{W}, \mathcal{D})$, where:

- $PN = (P, T, I^-, I^+, M_0)$ is the underlying Place-Transition net. P is the set of places, T, the set of transitions, $I^{+/-}$ are the forward and backward incidence functions describing the connections between places and transitions and M_0 is the initial marking.
- $\mathcal{P} : T \times \mathcal{M} \rightarrow \mathbb{Z}^+$, denoted $p_t(m)$, is a state-dependent priority function for a transition.
- $\mathcal{W} : T \times \mathcal{M} \rightarrow \mathbb{R}^+$, denoted $w_t(m)$, is a marking-dependent weight function for a transition, to allow implementation of probabilistic choice.

- $\mathcal{D} : T \times \mathcal{M} \rightarrow (\mathbb{R}^+ \rightarrow [0,1])$, denoted $d_t(m)$, is a marking-dependent cumulative distribution function for the firing-time of a transition.

In the above $\mathcal{M}$ is the set of all reachable markings for a given net. Further, we define the following general net-enabling functions:

- $\mathcal{E}_N : \mathcal{M} \rightarrow P(T)$, a function that specifies net-enabled transitions from a given marking.
- $\mathcal{E}_P : \mathcal{M} \rightarrow P(T)$, a function that specifies priority-enabled transitions from a given marking.

The net-enabling function, $\mathcal{E}_N$, is defined in the usual way for standard Petri nets: if all preceding places have occupying tokens then a transition is net-enabled. Similarly, we define the more stringent priority-enabling function, $\mathcal{E}_P$. For a given marking, m, $\mathcal{E}_P(m)$ selects only those net-enabled transitions that have the highest priority, that is:

$$\mathcal{E}_P(m) = \{t \in \mathcal{E}_N(m) : p_t(m) = \max\{p_{t'}(m) : t' \in \mathcal{E}_N(m)\}\} \tag{26}$$

Now for a given priority-enabled transition, $t \in \mathcal{E}_P(m)$, there is a probability that it will actually fire after a delay sampled from its firing distribution, $d_t(m)$:

$$\mathbb{P}(t \in \mathcal{E}_P(m) \text{ fires}) = \frac{w_t(m)}{\sum_{t' \in \mathcal{E}_P(m)} w_{t'}(m)} \tag{27}$$

Note that the choice of which priority-enabled transition is fired in any given marking is made by a probabilistic selection based on transition weights, and is not a race condition based on finding the minimum of samples extracted from firing time distributions. This mechanism enables the underlying reachability graph of the SM-SPN to be mapped directly onto a semi-Markov chain.

The marking-dependence of the weights and distributions does, in fact, allow us to translate SPNs and GSPNs into the SM-SPN paradigm in a straightforward manner, but that translation is not within the scope of this paper.

6.3 A Distributed Voting System

Fig. 3 shows the distributed components of a voting system with breakdowns and repairs, which we will use to generate a semi-Markov model. A voting agent queues to vote in the buffer; then, as a polling unit becomes free, it can receive the agent's vote and the agent can be marked as having voted. The polling unit contacts all the currently operational central voting units to register votes with all of them; this is done in order to prevent multiple vote fraud and to provide fault tolerance through redundancy. The polling unit then becomes available to receive another voting agent.

The semi-Markov stochastic Petri net for this system is shown in Fig. 4. Voting

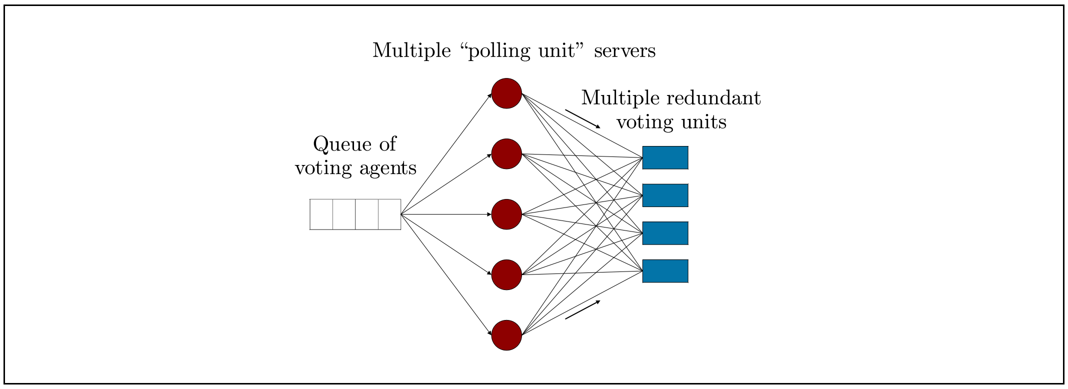

Fig. 3. A queueing model of a voting system

agents vote asynchronously, moving from place p_1 to p_2 as they do so. A restricted number of polling units which receive their votes transit t_1 from place p_3 to place p_4. At t_2, the vote is registered with as many central voting units as are currently operational in p_5.

The system is considered to be in a failure mode if either all the polling units have failed and are in p_7 or all the central voting units have failed and are in p_6. If either of these complete failures occur, then a high priority repair is performed, which resets the failed units to a fully operational state. If some but not all the polling or voting units fail, they attempt self-recovery. The system will continue to function as long as at least one polling unit and one voting unit remain operational.

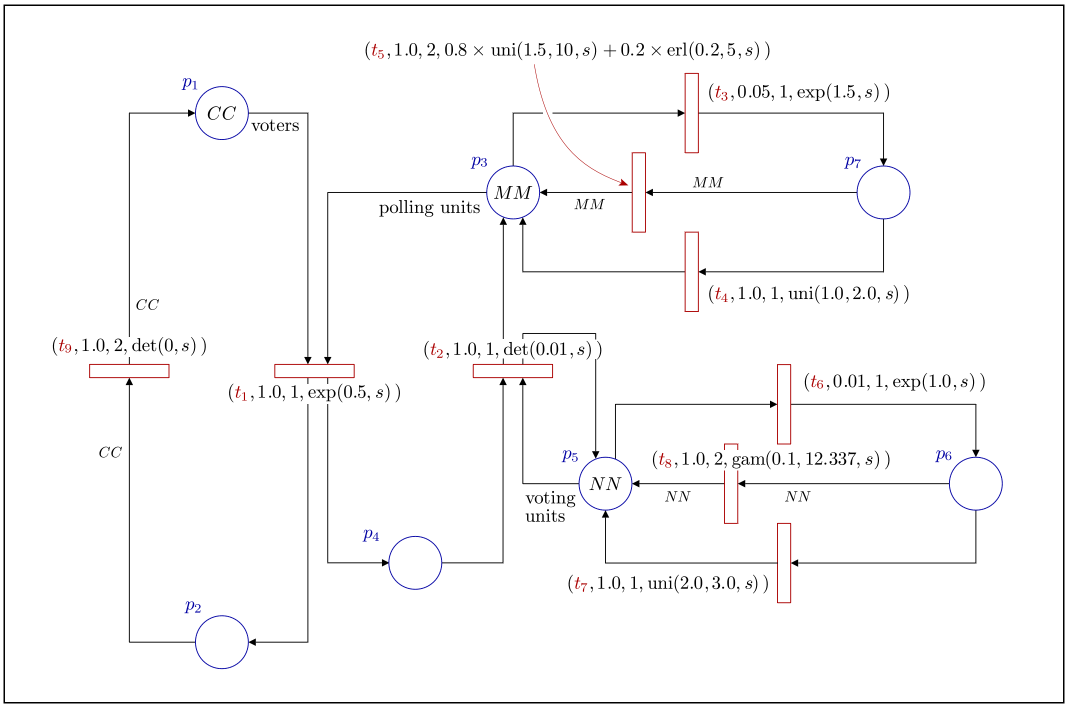

Fig. 4. A semi-Markov stochastic Petri net of a voting system with breakdowns and repairs

```
\transition{t5}{
  \condition{p7 > MM-1}
  \action{
    next->p3 = p3 + MM;
    next->p7 = p7 - MM;
  }
  \weight{1.0}
  \priority{2}
  \sojourntimeLT{
    return (0.8*uniformLT(1.5,10,s) + 0.2*erlangLT(0.001,5,s));
  }
}
```

Fig. 5. Excerpt from specification of voting example, showing definition of transition t_5.

This example is defined in full as a DNAmaca specification [3], an excerpt of which is shown in Fig. 5. This defines transition t_5, saying that it:

- is enabled when place p_7 has greater than $MM - 1$ tokens in it.
- removes MM tokens from place p_7 and adds MM tokens to place p_3, when fired.
- has a weight 1.0 (used to define probabilistic choice between transitions when two or more are concurrently enabled).
- has a priority of 2, which will enable it above other transitions which would otherwise be structurally enabled but have a lower priority.
- is given a firing distribution which, with probability 0.8, is a uniform distribution or, with probability 0.2, is an Erlang distribution. The Laplace transform $g^*(s)$ for this firing time distribution is:

  $$0.8 \times \mathit{uniformLT}(1.5, 10, s) + 0.2 \times \mathit{erlangLT}(0.001, 5, s)$$

  where

  $$\mathit{uniformLT}(a, b, s) = \frac{e^{-as} - e^{-bs}}{s(b-a)}$$

  and

  $$\mathit{erlangLT}(\lambda, n, s) = \left(\frac{\lambda}{\lambda + s}\right)^n$$

  In general, any arbitrary Laplace transform function can be specified as a firing distribution using the `\sojourntimeLT{...}` pragma.

6.4 Results

In this section, we compute passage time quantities for the time taken for a number of voters to pass from place p_1 to p_2 (a voter throughput quantity), as well as for the time taken for a fully operational system to enter a failure

System	CC	MM	NN	States
0	18	6	3	2061
1	60	25	4	106,540
2	100	30	4	249,760
3	125	40	4	541,280
4	150	40	5	778,850
5	175	45	5	1,140,050

Table 1
Different configurations of the voting system as used to present results

mode (i.e. when MM polling units fail in place p_7 or when NN central voting units fail in place p_6). We also extract simple reliability quantiles from cumulative distributions of the passage times, and transient measures for the voter throughput passage.

For the voting system described in Fig. 4, Table 1 shows how the size of the underlying SMP varies according to the configuration of the variables CC, MM, and NN, which are the number of voters, polling units and central voting units, respectively.

6.4.1 Example Passage Time Distributions

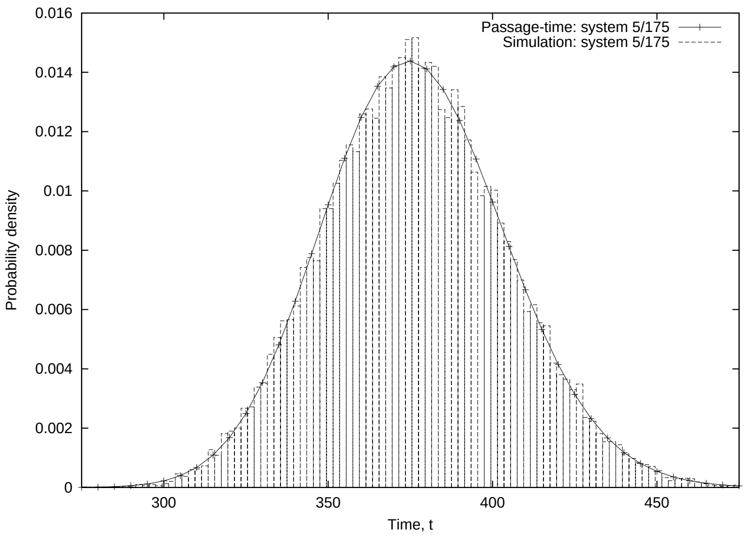

Fig. 6. Analytic and simulated density for the time taken to process 175 voters in system 5 (1.1 million states).

Fig. 6 shows the density of the time taken for the passage of 175 voters from place p_1 to p_2 in system 5 as computed by both our (truncated) iterative technique and by simulation. The close agreement provides mutual validation of the analytical method, with its numerical approximation, and the simulation.

It is interesting that, qualitatively, the density appears close to Normal. Certainly, the passage time random variable is a (weighted) sum of a large number of independent random variables, but these are, in general, not identically distributed.

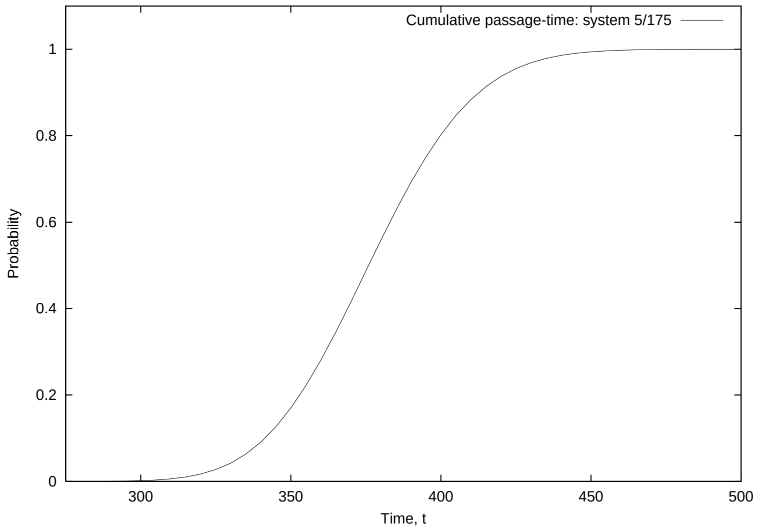

Fig. 7. Cumulative distribution function for the time taken to process 175 voters in system 5 (1.1 million states).

Fig. 7 shows a cumulative distribution for the same passage as Fig. 6. This is easily obtained by inverting the Laplace transform $L_{\vec{i}\vec{j}}(s)/s$; it allows us to extract response time quantiles, for instance:

$$\mathbb{P}(\text{system 5 processes 175 voters in under 440s}) = 0.9858$$

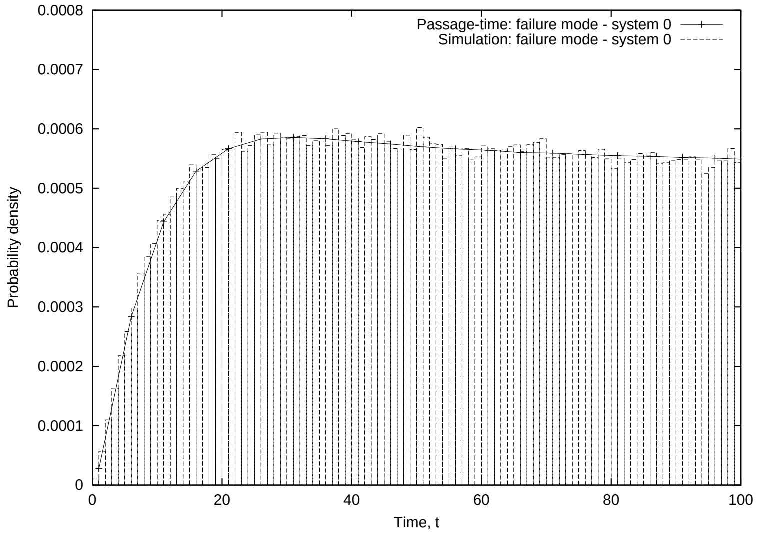

Fig. 8. Analytic and simulated density for failure mode passage in system 0 (2061 states).

Fig. 8 shows analytic and simulated results for the time to complete failure in an initially fully operational voting system. It is produced for a much smaller system (2061 states) as the probabilities for the larger systems were so small that the simulator was not able to register any meaningful distribution for the quantity without using rare-event techniques. As we wanted to validate the passage time algorithm, we reduced the number of states so that the simulator would register a density. Examining very-low-probability events is an excellent example of where analytical techniques out-perform simulations that would take many hours or even days to complete.

6.4.2 Example Transient State Distributions

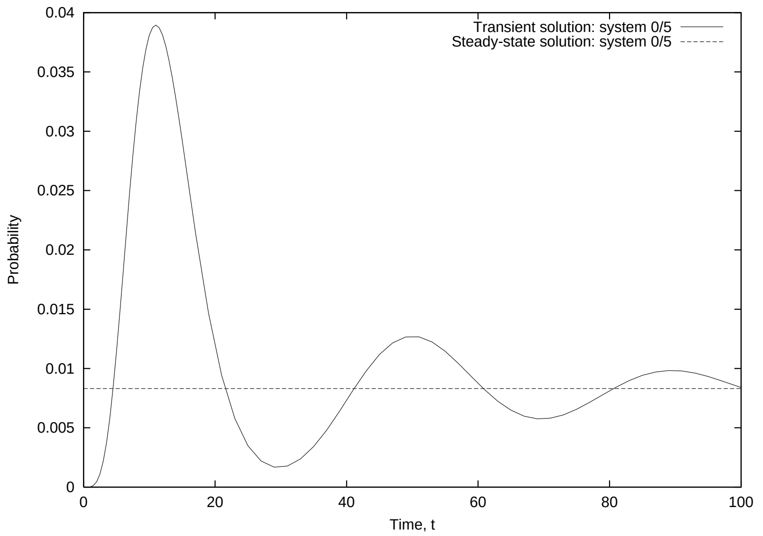

Fig. 9. Transient and steady-state values in system 0, for the transit of 5 voters from the initial marking to place p_2

We use the iterative transient calculation scheme of Section 3.3 to generate transient state distributions. Fig. 9 shows the transient state distribution for the transit of five voters from place p_1 to p_2 in system 0. As expected, the distribution tends towards its steady-state value as $t \to \infty$.

Fig. 10 shows the same measure but for a much larger system (106,000 states). There is a more noticeable separation between the first two peaks in Fig. 10 as there are many more voters to be processed (60 rather than 18 in the previous example). Again, we note that the transient state distribution tends towards the corresponding steady-state probability. It is worth noting that the iterative transient algorithm required at most 50 iterations to converge for each s-point (often less); this despite having a large time range of $0 < t < 500$.

6.4.3 Tool Scalability

Table 2 shows the time, speedups and efficiency for the analysis pipeline of Section 4 with varying numbers of slave processors when calculating 5 t-points for a passage time of system 1. The slave processors, each of which has a 2 GHz

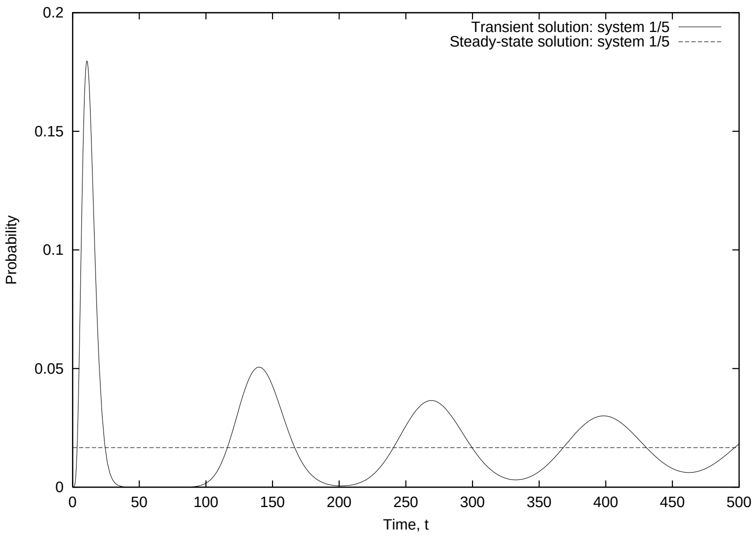

Fig. 10. Transient and steady-state values in system 1, for the transit of 5 voters from the initial marking to place p_2

Slave Processors	Time (s)	Speedup	Efficiency
1	549.08	1.00	1.000
8	71.11	7.72	0.965
16	39.16	14.02	0.876
32	24.10	22.79	0.712

Table 2
Time, speedup and efficiency for varying numbers of slave processors when calculating a passage time at 5 t-points for system 1, using Euler inversion (total of 165 s-point evaluations).

Intel Pentium 4 processor and 512 MB RAM, are part of a shared departmental network connected by 100MBps Ethernet. The master processor used was a dual 1 GHz Pentium III server with 2GB RAM (note, however, that a much lower spec machine would have been adequate as the master processor since it does not perform significant computation, nor does it require large amounts of memory). Even though exclusive access to the slave processors could not be guaranteed and the problem size in system 1 is relatively small, our distributed analysis pipeline still exhibits excellent scalability. The loss of efficiency, as the number of processors is increased, can be attributed to minor load imbalances between slaves (which can occur when the work queue is nearly empty) and the increase in the amount of communication between the master and slaves.

7 Conclusion

In this paper, we have derived passage time densities, quantiles and transient state distributions for distributed systems with underlying semi-Markov state spaces of up to 10^6 states.

Building on our recent passage time generation algorithm, we derived and implemented a new iterative algorithm that computes transient state distributions. Our implementation optimises storage by relating the function to a set of s-points necessary for Laplace transform inversion. In this way, storage of an arbitrary distribution is kept constant and successive vector–matrix iterations do not suffer from the problem of representation explosion.

Finally, we used a semi-Markov stochastic Petri net in conjunction with a semi-Markov extension to the DNAmaca language to specify a model of a distributed voting system, generate the corresponding semi-Markov state space and solve for a variety of transient and passage time measures.

Our research efforts in the near future will include studying the convergence behaviour of our transient algorithm, with the goal of obtaining analytical bounds on the truncation error. In addition, we will apply specialist techniques, e.g. using hypergraph partitioning of data structures, to achieve a scalable algorithm for systems with up to 10^8 states and beyond.

References

[1] J. T. Bradley, N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt, "Distributed computation of passage time quantiles and transient state distributions in large semi-Markov models," in *PMEO-PDS'03, Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems*, (Nice), IEEE Computer Society Press, April 2003.

[2] J. T. Bradley, N. J. Dingle, W. J. Knottenbelt, and H. J. Wilson, "Hypergraph-based parallel computation of passage time densities in large semi-Markov models," *Journal of Linear Algebra and Applications*, 2004. In press.

[3] W. J. Knottenbelt, "Generalised Markovian analysis of timed transitions systems," MSc thesis, University of Cape Town, South Africa, July 1996.

[4] J. T. Bradley, N. J. Dingle, W. J. Knottenbelt, and P. G. Harrison, "Performance queries on semi-Markov stochastic Petri nets with an extended Continuous Stochastic Logic," in *PNPM'03, Proceedings of Petri Nets and Performance Models* (G. Ciardo and W. Sanders, eds.), (University of Illinois at Urbana-Champaign), pp. 62–71, IEEE Computer Society Press, September 2003.

[5] R. Pyke, "Markov renewal processes with finitely many states," *Annals of Mathematical Statistics*, vol. 32, pp. 1243–1259, December 1961.

[6] J. K. Muppala and K. S. Trivedi, "Numerical transient analysis of finite Markovian queueing systems," in *Queueing and Related Models* (U. N. Bhat and I. V. Basawa, eds.), pp. 262–284, Oxford University Press, 1992.

[7] B. Melamed and M. Yadin, "Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes," *Operations Research*, vol. 32, pp. 926–944, July–August 1984.

[8] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*. Wiley, August 1998.

[9] J. Abate and W. Whitt, "The Fourier-series method for inverting transforms of probability distributions," *Queueing Systems*, vol. 10, no. 1, pp. 5–88, 1992.

[10] J. Abate and W. Whitt, "Numerical inversion of Laplace transforms of probability distributions," *ORSA Journal on Computing*, vol. 7, no. 1, pp. 36–43, 1995.

[11] J. Abate, G. L. Choudhury, and W. Whitt, "On the Laguerre method for numerically inverting Laplace transforms," *INFORMS Journal on Computing*, vol. 8, no. 4, pp. 413–427, 1996.

[12] P. G. Harrison and W. J. Knottenbelt, "Passage-time distributions in large Markov chains," in *Proceedings of ACM SIGMETRICS 2002* (M. Martonosi and E. d. S. e Silva, eds.), pp. 77–85, Marina Del Rey, USA, June 2002.

[13] H. H. Ammar, "Performance models of parallel and distributed processing systems," in *Proceedings of ACM 14th Annual Computer Science Conference: CSC'86*, ACM, 1986.

[14] Y. Sugasawa, Q. Jin, and K. Seya, "Extended stochastic Petri net models for systems with parallel and cooperative motions," *Computer Mathematical Application/2*, vol. 24, no. 1, 1992.

[15] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, pp. 93–122, May 1984.