Passage Time Distributions in Large Markov Chains

Peter G. Harrison and William J. Knottenbelt
Department of Computing Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, United Kingdom
{pgh,wjk}@doc.ic.ac.uk

ABSTRACT

Probability distributions of response times are important in the design and analysis of transaction processing systems and computer-communication systems. We present a general technique for deriving such distributions from high-level modelling formalisms whose state spaces can be mapped onto finite Markov chains. We use a load-balanced, distributed implementation to find the Laplace transform of the first passage time density and its derivatives at arbitrary values of the transform parameter s. Setting $s = 0$ yields moments while the full passage time distribution is obtained using a novel distributed Laplace transform inverter based on the Laguerre method. We validate our method against a variety of simple densities, cycle time densities in certain overtake-free (tree-like) queueing networks and a simulated Petri net model. Our implementation is thereby rigorously validated and has already been applied to substantial Markov chains with over 1 million states. Corresponding theoretical results for semi-Markov chains are also presented.

1. INTRODUCTION

Quantiles for message transmission times and other response times are required in the design of computer-communication systems and are often specified in on-line transaction processing benchmarks. However, their calculation in analytic models has proved problematic over a period of many years and simulation estimates are expensive to provide, especially in the tail. The Laplace transform of (first) passage time density can be determined for arbitrary continuous time Markov chains (CTMCs) as the solution of a set of linear equations, but this route is expensive and has only been numerically tractable for small chains. This is because there is an equation for every state in the chain and the set of equations have to be solved at every point required in the Laplace domain. An alternative to using Laplace transforms is uniformization [5], but this technique does not readily generalize to semi-Markov chains, unlike the present study. One approach to these computational problems is to develop approximate techniques for calculating response time distributions, but these will be difficult to validate in the light of the above obstacles. However, even a small increase in the complexity of systems of general structure that can be analysed exactly would help to validate new approximate methods.

Our contribution is to apply recent results on the solution of very large Markov chains to this problem. We can presently solve chains with over 100M states for their steady state probability distribution using a combination of probabilistic techniques in the representation of states (through hashing) and a load-balanced, parallel linear solver [17, 16]. The problem of finding the value of the Laplace transform of a passage time density at a given point requires similar computational resources. We obtain passage time densities from a checkpointed, distributed Laplace transform inverter which typically evaluates the transform at several hundred points. There are also simpler ways to parallelise than to use parallel solution techniques for solving linear equations. In particular, if enough points are known at which the Laplace transform needs to be computed (specifically at least as many points as there are processors), a linear equation solver algorithm can be run on each processor independently. Obtaining moments is an easier problem, requiring the solution of just one set of linear equations – given by the derivatives of the Laplace transform at the origin – for each moment. Hence 100M state chains can be solved for passage time moments using existing tools.

Using our techniques significant models can be solved for response time distributions. We validate our inversion algorithm by comparing with exact results for simple densities and overtake-free queueing networks. These results can be obtained by analytically inverting the Laplace transform of response time density. This yields a solution in the time domain, which can be implemented efficiently to provide directly the density function of response time at high accuracy. In this way, our general method for numerically evaluating passage time distributions in Markov chains can be tested rigorously. A further application is a Petri net model describing the readers/writers problem, which is validated by simulation.

2. FIRST PASSAGE TIMES

2.1 First passage time equations

Consider a finite, irreducible, continuous time Markov Chain with n states $\{1, 2, \ldots, n\}$ and generator matrix Q. If $X(t)$ denotes the state of the CTMC at time t ($t \geq 0$), then the first passage time from a source state i into a non-empty set of target states $\vec{j}$ is:

$$T_{i\vec{j}}(t) = \inf\{u > 0 : \ X(t+u) \in \vec{j} \mid X(t) = i\} \qquad (\forall t \geq 0)$$

For a stationary time-homogeneous CTMC, $T_{i\vec{j}}(t)$ is independent of t, so:

$$T_{i\vec{j}} = \inf\{u > 0 : \ X(u) \in \vec{j} \mid X(0) = i\}$$

$T_{i\vec{j}}$ is a random variable with an associated probability density

function $f_{i\vec{j}}(t)$ such that

$$\Pr(a < T_{i\vec{j}} < b) = \int_a^b f_{i\vec{j}}(t)dt \quad (0 \le a < b)$$

Our aim is to determine $f_{i\vec{j}}(t)$. In effect, this involves convolving state holding times over all possible paths (including cycles) from state i into any of the states in the set $\vec{j}$. By shifting the problem into the Laplace domain we can exploit the basic transform property that the transform of a convolution of two functions is the product of the transforms of those functions [2]. Another important advantage of working with Laplace transforms is that we can derive arbitrary moments of $f_{i\vec{j}}(t)$ by evaluating derivatives of $L_{i\vec{j}}(s)$ at $s = 0$ (see Section 2.2). From the transform we can recover the value of $f_{i\vec{j}}(t)$ at any t by using one of several algorithms for numerical transform inversion. Examples of well-known numerical inversion algorithms include the Euler, Post-Widder, Gaver and Laguerre methods [3, 4, 1, 2]. These algorithms compute $f_{i\vec{j}}(t)$ at a given t by evaluating $L_{i\vec{j}}(s)$ at several values of s.

In general, the value of $L_{i\vec{j}}(s)$ can be computed by solving a set of n linear equations that are derived using a first-step analysis:

$$\begin{aligned} L_{i\vec{j}}(s) &= \int_0^\infty e^{-st} f_{i\vec{j}}(t)dt \\ &= E[e^{-sT_{i\vec{j}}}] \\ &= \sum_{i' \notin \vec{j}} -\frac{q_{ii'}}{q_{ii}} E[e^{-s(S_i + T_{i'\vec{j}})}] + \sum_{i' \in \vec{j}} -\frac{q_{ii'}}{q_{ii}} E[e^{-s(S_i)}] \\ &= \sum_{i' \notin \vec{j}} \frac{q_{ii'}}{(s - q_{ii})} L_{i'\vec{j}}(s) + \sum_{i' \in \vec{j}} \frac{q_{ii'}}{(s - q_{ii})} \end{aligned}$$

i.e.

$$(s - q_{ii}) L_{i\vec{j}}(s) = \sum_{i' \notin \vec{j}} q_{ii'} L_{i'\vec{j}}(s) + \sum_{i' \in \vec{j}} q_{ii'} \quad (1)$$

where $S_i \sim \text{Exp}(-q_{ii})$ is the sojourn time in state i ($1 \le i \le n$).

Expressing this system of n linear equations in standard matrix-vector form ($Ax = b$) yields:

$$\begin{pmatrix} s - q_{11} & -q_{12} & \cdots & -q_{1n} \\ 0 & s - q_{22} & \cdots & -q_{2n} \\ 0 & -q_{32} & \cdots & -q_{3n} \\ 0 & \vdots & \ddots & \vdots \\ 0 & -q_{n2} & \cdots & s - q_{nn} \end{pmatrix} \begin{pmatrix} L_{1\vec{j}}(s) \\ L_{2\vec{j}}(s) \\ L_{3\vec{j}}(s) \\ \vdots \\ L_{n\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} 0 \\ q_{21} \\ q_{31} \\ \vdots \\ q_{n1} \end{pmatrix} \quad (2)$$

where $\vec{j} = \{1\}$ in this case.

The problem can also be readily extended to multiple initial states. In particular, if the probability distribution of the initial states is known – typically the equilibrium distribution – the problem reduces to that of weighting the first passage time densities for each initial state.

2.2 Moments

The nth moment of the first passage time between a given source state i and set of target states $\vec{j}$ is

$$M_{i\vec{j}}(n) = (-1)^n \left. \frac{\mathrm{d}^n L_{i\vec{j}}(s)}{\mathrm{d}s^n} \right|_{s=0}$$

This can be found by differentiating (1) n times at $s = 0$ and solving a similar set of equations, for $n \ge 0$:

$$-q_{ii} M_{i\vec{j}}(n) = \sum_{k \notin \vec{j}} q_{ik} M_{k\vec{j}}(n) + n M_{i\vec{j}}(n-1) \quad (3)$$

for $i \notin \vec{j}$ and $M_{i\vec{j}}(n) = 0$ for $i \in \vec{j}$. For $n = 0$, we have $M_{i\vec{j}}(0) = 1$ and so each set of moments can be computed iteratively.

3. EXTENSION TO SEMI-MARKOV CHAINS

3.1 Problem definition

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

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

for $i, j \in \mathcal{S}$. The continuous time semi-Markov process (CTSMC) defined by the kernel R is then $Y(t) = X_n$ where n is the non-negative integer for which $t \in [T_n, T_{n+1})$. We consider time homogeneous CTSMCs, in which $R(n, i, j, t)$ is independent of n; we write it $R(i, j, t)$. The first passage time between a given source state i and set of target states $\vec{j}$ then has a density function with Laplace transform $L_{i\vec{j}}(s)$ given by the following equations, analogous to (1) for Markov chains,

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} r^*_{ik}(s) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} r^*_{ik}(s) \quad (4)$$

where $r^*_{ik}(s)$ is the Laplace-Stieltjes transform of $R(i, k, t)$, i.e.

$$r^*_{ik}(s) = \int_0^\infty e^{-st} dR(i, k, t)$$

Here, Laplace transform inversion has higher complexity in general since all of the coefficients in the linear equations are complex functions of s. However, if X_{n+1} depends only on X_n and not $T_{n+1} - T_n$ (i.e. the next state depends only on the current state, not the holding time in the current state),

$$R(n, i, j, t) = p_{ij} H_i(t)$$

where $p_{ij} = P(X_{n+1} = j \mid X_n = i)$ is the state transition probability between states i and j and $H_i = P(T_{n+1} - T_n \le t \mid X_n = i)$ is the state i holding time probability distribution. Thus, in the special case of a Markov process, $p_{ij} = -q_{ij}/q_{ii}$ and $H_i(t) = 1 - e^{q_{ii}t}$. Denoting the Laplace-Stieltjes transform of $H_i(t)$ by $h^*_i(s)$, we then get

$$L_{i\vec{j}}(s)/h^*_i(s) = \sum_{k \notin \vec{j}} p_{ik} L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} p_{ik} \quad (5)$$

The complex numbers then only appear in the diagonal elements of the matrix equations and the analysis mirrors that of the Markovian case.

3.2 Moments

Assuming the derivatives of $r^*_{ik}(s)$ exist at the origin, we write

$$m_{ik}(n) = (-1)^n \left. \frac{\mathrm{d}^n r^*_{ik}(s)}{\mathrm{d}s^n} \right|_{s=0}$$

for the nth moment of the holding time in state i with next state k. Hence we get, using Leibnitz' rule,

$$\begin{aligned} M_{i\vec{j}}(n) &= \sum_{k\notin\vec{j}}\sum_{r=0}^{n}\left(\begin{array}{c} n \\ r \end{array}\right) m_{ik}(r)M_{k\vec{j}}(n-r) + \sum_{k\in\vec{j}} m_{ik}(n) \\ &= \sum_{k\notin\vec{j}}\sum_{r=1}^{n}\left(\begin{array}{c} n \\ r \end{array}\right) m_{ik}(r)M_{k\vec{j}}(n-r) + \\ &\quad \sum_{k\notin\vec{j}} p_{ik}M_{k\vec{j}}(n) + \sum_{k\in\vec{j}} m_{ik}(n) \end{aligned} \quad (6)$$

for $i \notin \vec{j}$ and $M_{i\vec{j}}(n) = 0$ for $i \in \vec{j}$, where $p_{ik} = r^*_{ik}(0) \equiv m_{ik}(0)$. The first and third terms on the right hand side will be known prior to the iteration, facilitating a straightforward iteration that solves a set of linear equations at each step.

When state holding times are independent of the next state, we have

$$m_{ik}(n) = p_{ik}m_i(n)$$

where $m_i(n) = (-1)^n \left.\frac{\mathrm{d}^n h^*_i(s)}{\mathrm{d}s^n}\right|_{s=0}$ is the nth moment of the holding time in state i. Instead of Eq. 6, we now have, by differentiating Eq. 5, the simpler

$$M_{i\vec{j}}(n) + \sum_{r=1}^{n}\left(\begin{array}{c} n \\ r \end{array}\right) M_{i\vec{j}}(n-r)u_i(r) = \sum_{k\notin\vec{j}} p_{ik}M_{k\vec{j}}(n) \quad (7)$$

where $u_i(r) = (-1)^r \left.\frac{\mathrm{d}^r}{\mathrm{d}s^r}[1/h^*_i(s)]\right|_{s=0}$ We can calculate the $u_i(r)$ as follows. Dropping the subscript i for now, let $y(s) = 1/h^*(s)$. Denote the jth derivative of $y(s)$ by $y^{(j)} \equiv \frac{\mathrm{d}^j y}{\mathrm{d}s^j}$ ($j \geq 0$) and $h^{*(j)}$ similarly. Then, differentiating r times the equation $y(s)h^*(s) = 1$ yields

$$\sum_{j=0}^{r}\left(\begin{array}{c} r \\ j \end{array}\right) h^{*(j)}y^{(r-j)} = 0$$

for $r \geq 1$, so that

$$h^*y^{(r)} = -\sum_{j=1}^{r}\left(\begin{array}{c} r \\ j \end{array}\right) h^{*(j)}y^{(r-j)}$$

Setting $s = 0$ and recalling that $h^*(0) = 1$, we obtain

$$\begin{aligned} u_i(r) &= -\sum_{j=1}^{r}\left(\begin{array}{c} r \\ j \end{array}\right) m_i(j)u_i(r-j) \quad (8) \\ u_i(0) &= 1 \end{aligned}$$

This simple recurrence allows all of the $u_i(r)$ terms to be precalculated for every $i \in \mathcal{S}$ and required moments $r \geq 0$.

4. PASSAGE TIME DENSITIES

4.1 The Laguerre Method for Laplace Transform Inversion

The Laguerre method [1] (sometimes also referred to as Weeks' method) represents a function $f(t)$ in terms of its Laplace transform $L(s)$ as the sum

$$f(t) = \sum_{n=0}^{\infty} q_n l_n(t)$$

where:

- the q_n are the *Laguerre coefficients*, given by the Cauchy contour integral

$$q_n = \frac{1}{2\pi i}\int_{C_r} Q(z)/z^{n+1}dz \quad (9)$$

  In Eq. 9 $Q(z)$ is the *Laguerre generating function* given by

$$Q(z) = \sum_{n=0}^{\infty} q_n z^n = (1-z)L\left(\frac{1+z}{2(1-z)}\right) \quad (10)$$

  and C_r is a circle about the origin of radius r ($0 < r < 1$) such that $Q(z)$ is analytic in $\{z : |z| < r\}$.

- the $l_n(t)$ are the *Laguerre functions*, which can be calculated in a numerically stable way from the recursion:

$$l_n(t) = \left(\frac{2n-1-t}{n}\right) l_{n-1}(t) - \left(\frac{n-1}{n}\right) l_{n-2}(t)$$

  starting with $l_0(t) = e^{-t/2}$ and $l_1(t) = (1-t)l_0(t)$ [1].

4.2 Convergence of the Laguerre Series

As noted in [1], $|l_n(t)| <= 1$ for all n and t and $l_n(t)$ approaches 0 as $n \to \infty$. However, the latter rate of convergence is very slow so the convergence of the Laguerre series effectively depends on the decay rate of q_n as $n \to \infty$. If f is continuous and has continuous derivatives, convergence of the Laguerre coefficients is rapid. In the context of Markov chains, this is typically the case since passage time densities are smooth functions, namely weighted convolutions of exponential densities. However, lack of smoothness in f and in its derivatives (for example in the context of semi-Markov chains) can lead to particularly slow convergence of the Laguerre coefficients [12].

Slow convergence of the q_n coefficients can often be addressed by exponential damping and scaling using two real parameters σ and b [18]. The idea is to apply the Laguerre inversion algorithm to the function:

$$f_{\sigma,b}(t) = e^{-\sigma t}f(t/b)$$

Then $f(t)$ can be recovered as:

$$f(t) = e^{\sigma bt}f_{\sigma,b}(bt).$$

The corresponding Laguerre generating function for $f_{\sigma,b}(t)$ is

$$Q_{\sigma,b}(z) = \frac{b}{1-z}L\left(\frac{b(1+z)}{2(1-z)} + b\sigma\right).$$

Suitable scaling parameters can be automatically determined using the simple algorithm presented in Fig. 1. This algorithm is based on the heuristic observations in [1] that increasing b (up to a given limit) can significantly lower the ratio $|q_n|/|q_0|$, and our own observation that excessive values of the damping parameter σ can lead to numerical instability in finite precision arithmetic.

It may be that no suitable scaling parameters can be found (because of discontinuities in the underlying response time distribution or its derivatives for example). In this case one alternative is to use the Euler Laplace transform inversion method [3] instead. The Euler method is able to handle discontinuities well but requires substantially more computation – of the order of 50 distinct evaluations of $L(s)$ for each t point are required.

$\sigma = 0$
$b = 1$
while $|q_{200}| > 10^{-10}$ **or** $|q_{201}| > 10^{-10}$ **do begin**
  if $\sigma = 0$ **then**
    $\sigma = 0.001$
  else
    $\sigma = 2\sigma$
  if $\sigma > 0.2$ **then begin**
    $b = b + 4$
    if $b > 10$ **then**
      terminate("no suitable parameters were found")
    $\sigma = 0$
  end
end

Figure 1: Algorithm for automatically determining scaling parameters

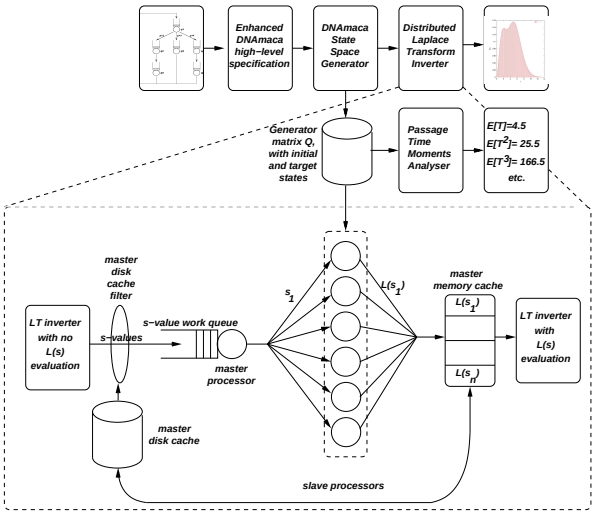


Figure 2: Passage time analysis pipeline showing the detailed operation of the distributed Laplace transform inverter

4.3 Computing the Laguerre coefficients

The most computationally intensive part of the Laguerre algorithm is the computation of the q_n coefficients. Making the change of variable $z = re^{iu}$ in Eq. 9 gives

$$q_n = \frac{1}{2\pi r^n} \int_0^{2\pi} Q(re^{iu}) e^{-inu} du$$

This integral can be approximated using the trapezoidal rule using p trapezoids so that

$$q_n \approx \frac{1}{2pr^n}(Q(r) + (-1)^p Q(-r) + 2\sum_{j=1}^{p-1} \mathrm{Re}(Q(re^{2\pi ji/p})(e^{-2\pi nji/p})))$$

In [1] $p = 2n$ trapezoids are used when calculating q_n, with a suggested value of $r = (0.1)^{4/n}$. However, if we apply the scaling procedure of Section 4.2 to ensure that q_n has decayed to (almost) zero by term p_0 (say $p_0 = 200$), we can instead make use of a constant number of $2p_0$ trapezoids and a constant $r = (0.1)^{4/p_0}$ when calculating each q_n. This allows us to calculate each q_n with the same or higher accuracy as in [1] while simultaneously providing the opportunity to cache and re-use values of $Q(z)$. Since q_n does not depend on t, and each evaluation of $Q(z)$ involves a single evaluation of $L(s)$, we can therefore obtain the response time distribution at an arbitrary number of t-values at the fixed cost of solving just $2p_0$ linear systems (of the form given in Eq. 2).

5. IMPLEMENTATION OF A COMPLETE PASSAGE TIME ANALYSIS PIPELINE

5.1 Architecture

We have implemented a complete passage time analysis pipeline, as shown in Fig. 2. Models are specified in an enhanced form of the DNAmaca Markov Chain Analyser interface language [14, 15]. This interface language supports the specification of Queueing networks, stochastic Petri nets, stochastic Process Algebras and other high-level formalisms that can be mapped onto Markov chains.

From the high-level input model, DNAmaca's state generator uses a probabilistic algorithm based on 3-level hashing [17] to produce the generator matrix Q of the model's underlying Markov chain, as well as a list of the initial states (with their corresponding weightings) and the target states.

Control is now passed to the distributed Laplace transform inverter, which implements the master-slave model shown in Fig. 2. The inverter is written in C++ and uses the Message Passing Interface (MPI) [9] standard so it is portable to a wide variety of parallel computers and workstation clusters. Both the Laguerre (with the modified kernel for calculating the Laguerre coefficients described in Section 4.3) and Euler inversion algorithms are supported.

Initially, the master simply runs through the Laplace transform inversion algorithm (Laguerre or Euler) and notes the distinct values of s at which $L(s)$ will need to be evaluated. Those values of s for which there is no value of $L(s)$ already stored in the disk cache are added to a global work queue.

At start-up, slave processors read into memory the generator matrix Q, as well a list of the initial and target states. Each slave processor then applies for an s-value from the global work queue. The slave calculates the corresponding value of $L(s)$ by solving a set of sparse linear equations (of the form given in Eq. 2) using an appropriate iterative numerical method; currently Gauss-Seidel, SOR with dynamic parameter adjustment and Conjugate Gradient Squared (CGS) are supported.

The solution of the above linear systems are currently performed in-core, which limits the solution capacity of an individual slave processor to around 3 million states on a 256MB machine. However, this solution capacity could easily be increased to around 20 million states by implementing an out-of-core solution method such as the disk-based method described in [7]. Further, groups of slave processors could be used to jointly solve very large systems of around 100 million states or more by implementing a parallel disk-based solution method such as that described in [16]. Note that, for each different value of s, it is only necessary for a slave processor to modify the diagonal elements of its linear system so setup is rapid.

Slave processors return computed values of $L(s)$ to the master. The master stores the returned value in memory and disk caches and immediately issues more work to the slaves if any is available. The disk cache stores values of $L(s)$ using both the value of s and an MD5 checksum (i.e. a 128-bit collision-resistant fingerprint) of the original high-level model file (as provided by the UNIX utility `md5sum`) as the key. This mechanism avoids redundant work by ensuring that no slave will have to recompute a value of $L(s)$ that has been previously computed for a given model at any point in the past. It also provides a convenient distributed checkpointing mechanism so that parallel jobs that are interrupted can be rapidly restarted without losing work already done.

5.2 Density functions

When all values of $L(s)$ have been computed, the master runs through the Laplace transform inversion algorithm again, this time performing all calculations and obtaining any values of $L(s)$ needed from the memory cache. The resulting points on the passage time distribution curve are written to a disk file, and displayed using the GNUplot graph plotting utility.

5.3 Moments

The moments analyser uses Eq. 3 to generate and solve a set of linear equations for each moment required. These are solved sequentially on a single PC, since the nth moment depends on the $(n-1)$th moment. As before, for very large chains of the order of 100 million states, we could apply a parallel linear equation solver. Note that, for semi-Markov chains, the calculation of the nth moment requires all moments from 1 to $n-1$.

5.4 Scalability

The distributed Laplace transform inverter described above is highly scalable for two reasons. Firstly, tqhe amount of communication performed is minimal and depends only on the number of slave processors used (and not on the problem size). Secondly, The single global work queue with multiple servers ensures a good load balance and very high utilization of slave processors.

For moment calculations, scalability is not applicable unless a parallel linear equation solver is used. If this is the case, scalability is not as impressive because of the higher communication load [16].

6. SIMPLE NUMERICAL RESULTS

6.1 Erlang-3

We begin by validating our analyser on a very simple example. On the left in Fig. 3 is a simple 3-stage Erlang distribution with parameter $\lambda = 2$. The closed-form analytical expression for the cycle time density function in this system is well-known to be

$$f(t) = \lambda^3 t^2 e^{-\lambda t}/2!.$$

As shown on the right in Fig. 3, the numerical cycle time density determined by our tool matches this analytical result. The moments analyser calculates the first three moments of the passage time as 1.5, 3 and 7.5 respectively, agreeing with their theoretical values exactly.

6.2 Branching Erlang

We now move on to a slightly more complicated example which shows the ability of the analyser to cope with bimodal passage time curves. Fig. 4 shows a system with two alternative branches, each of which is selected with equal probability. One branch is a 3-stage Erlang distribution with parameter $\lambda_1 = 1.0$; the other is a 12-stage Erlang distribution with parameter $\lambda_2 = 2.0$. The analytical expression for the cycle time density in this case is:

$$f(t) = \left(\frac{\lambda_1^3 t^2 e^{-\lambda_1 t}}{2!} + \frac{\lambda_2^{12} t^{11} e^{-\lambda_2 t}}{11!} \right) /2$$

Agreement between numerical and analytical results is once again excellent, as shown on the left in Fig. 5. The moments analyser again correctly calculates the first three moments of the passage time as 4.5, 25.5 and 166.5.

To validate the analyser for a case where the initial and target states are different, we consider the passage time from the first to the last stage in the upper branch (i.e. from the stage marked light green to that marked dark red). This has an Erlang-11 distribution, as confirmed by the numerical and analytical passage time distribution shown on the right in Fig. 5. The corresponding first three moments are 5.5, 33 and 214.5.

7. MORE COMPLEX APPLICATIONS

7.1 Cycle times in queueing networks

We now consider, as a more complicated example, cycle times in closed *tree-like* Markovian queueing networks with first come first served queueing discipline at each node [10]. These are characterised by a unique root node and unique paths to a set of leaf nodes from each of which departures proceed to the root. Hence the term "tree-like", an instance of *overtake-free* [13, 8] networks. The response time density function in this type of network can be found analytically, while the networks can be complex and the number of states in the underlying Markov chain can be made arbitrarily large by increasing the number of nodes or the customer population. It is therefore ideal for validation of our passage time analyser which works directly on the underlying Markov chain.

In this section we derive the response time density function in the time domain and use this in the following section for validation. Consider a closed tree-like network with M nodes and population N, in which node i has service rate μ_i and visitation rate (proportional to) v_i, $1 \leq i \leq M$. The routing probability from node i to node $j \neq i$, i.e. the probability that a customer leaving node i next visits node j, is $p_{ij} (1 \leq i, j \leq M)$. Let G be the network's normalising constant function for the joint equilibrium state probabilities, i.e. at population n,

$$G(n) = \sum_{\substack{\sum_{i=1}^{M} n_i = n \\ n_i \geq 0}} \prod_{i=1}^{M} x_i^{n_i}$$

where $x_i = v_i/\mu_i$. Without loss of generality, the root node is numbered 1 and we define the cycle time random variable to be the elapsed time between a customer's successive arrival instants at node 1. Then we have the following result [10, 13, 8].

THEOREM 1. *For the above closed treelike network, the Laplace transform of the cycle time density function, conditional on choice*

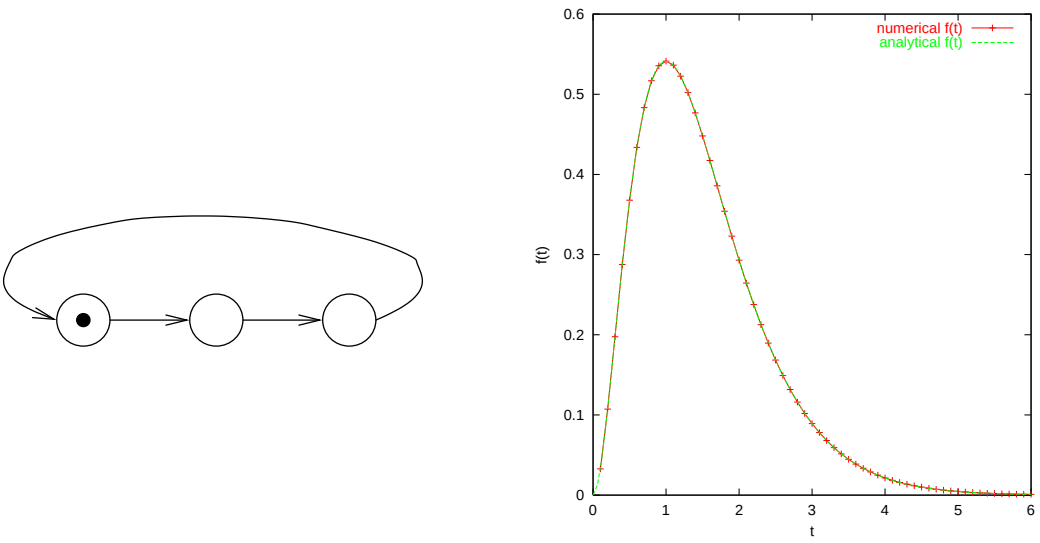


Figure 3: An Erlang-3 distribution (left) and corresponding numerical and analytical cycle time distributions (right)

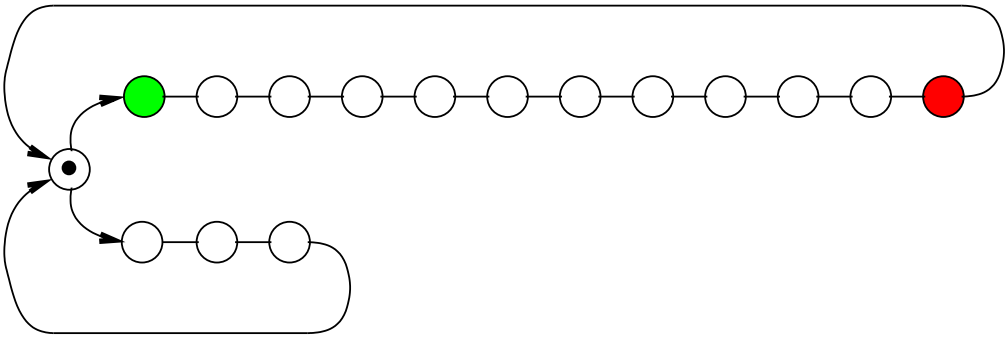


Figure 4: A branching Erlang distribution. The lower branch is a 3-stage Erlang distribution with $\lambda = 1$, while the upper branch is a 12-stage Erlang distribution with $\lambda = 2$.

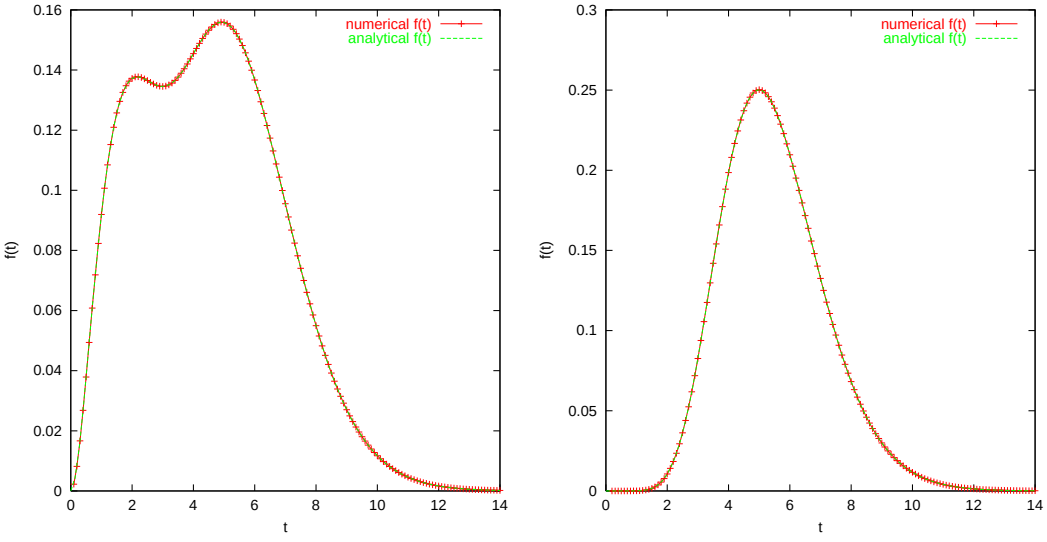


Figure 5: Branching Erlang cycle time distribution (left) and passage time distribution from the first to the last stage in the upper branch (right)

of path $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ *(*$m \leq M$*) is*

$$L(s|\mathbf{z}) = \frac{1}{G(n-1)} \times \sum_{\substack{\sum_{i=1}^{M} u_i = n-1 \\ u_i \geq 0}} \prod_{i=1}^{M} x_i^{u_i} \prod_{j=1}^{m} \left(\frac{\mu_{z_j}}{s + \mu_{z_j}} \right)^{u_{z_j}+1}$$

where $z_1 = 1$*,* z_m *is a leaf node and* $p_{z_i z_{i+1}} > 0$ *for* $1 \leq i \leq m-1$*.*

Without loss of generality, we take $z_i = i$ for $1 \leq i \leq m$, i.e. we consider the path $1, \ldots, m$. First, we can simplify the summation giving $L(s|\mathbf{z})$ by partitioning it over the state space according to the total number of customers, c, at servers in the overtake-free path $1, 2, ..., m$. This gives:

$$L(s|\mathbf{z}) = \frac{1}{G(n-1)} \sum_{c=0}^{n-1} G_m(n-c-1) \times \sum_{\substack{\sum_{i=1}^{m} n_i = c \\ n_i \geq 0}} \prod_{i=1}^{m} x_i^{n_i} \prod_{j=1}^{m} \left(\frac{\mu_j}{s + \mu_j} \right)^{n_j+1}$$

where $G_m(k)$ is the normalising constant of the whole network with servers $1, \ldots, m$ removed and population $k \geq 0$, i.e.

$$G_m(k) = \sum_{\substack{\sum_{i=m+1}^{M} n_i = k \\ n_i \geq 0}} \prod_{i=m+1}^{M} x_i^{n_i}$$

Now, the Laplace transforms in the inner sum are products of the Laplace transforms of Erlang densities. Moreover, their coefficients are geometric. Such transforms can be inverted analytically. In the simplest case, all the servers on the overtake-free path are identical, i.e. have the same rate, and the inversion can be done by inspection [11]. In the case that the μ_i are all distinct ($1 \leq i \leq m$), the density function is given by the following theorem, modified from that of [10].

THEOREM 2. *If the servers in an overtake-free path* $(1, 2, ..., m)$ *have distinct service rates* $\mu_1, \mu_2, ..., \mu_m$*, the passage time density function, conditional on the choice of path, is*

$$\frac{\prod_{i=1}^{m} \mu_i}{G(n-1)} \sum_{c=0}^{n-1} G_m(n-c-1) \times \sum_{j=1}^{m} \frac{e^{-\mu_j t}}{\prod_{1 \leq i \neq j \leq m} (\mu_i - \mu_j)} \sum_{i=0}^{c} \frac{(v_j t)^{c-i}}{(c-i)!} K^m(j, i)$$

where $K^m(j, \cdot)$ *is the normalising constant function for the subnetwork comprising only nodes in the set* $\{1, \ldots, m\} \backslash \{j\}$ *with the ratio* $x_k = v_k/\mu_k$ *replaced by* $\frac{v_k - v_j}{\mu_k - \mu_j}$ *for* $1 \leq k \neq j \leq m$*, i.e.*

$$K^m(j, l) = \sum_{\substack{\sum_{i=1}^{m} n_i = l \\ n_i \geq 0; n_j = 0}} \prod_{\substack{k=1 \\ k \neq j}}^{m} \left(\frac{v_k - v_j}{\mu_k - \mu_j} \right)^{n_k}$$

$K^m(j, l)$ is just a normalising constant that may be computed efficiently, along with $G_m(n-c-1)$ and $G(n-1)$, by Buzen's algorithm [6]. Thus we define the recursive function k, for real vector $\mathbf{y} = (y_1, \ldots, y_a)$ and integers a, b $(0 \leq a \leq M,\ \ 0 \leq b \leq N-1)$ by:

$$\begin{aligned} k(\mathbf{y}, a, b) &= k(\mathbf{y}, a-1, b) + y_a k(\mathbf{y}, a, b-1) \quad (a, b > 0) \\ k(\mathbf{y}, a, 0) &= 1 \quad (a > 0) \\ k(\mathbf{y}, 0, b) &= 0 \quad (b \geq 0) \end{aligned}$$

Then we have

$$\begin{aligned} G_m(l) &= k(\mathbf{x}_m, M-m, l) \quad (0 \leq l \leq n-1) \\ G(n-1) &\equiv G_0(n-1) = k(\mathbf{x}, M, n-1) \\ K^m(j, l) &= k(\mathbf{w}_j, m-1, l) \end{aligned}$$

where $\mathbf{x} = (x_1, \ldots, x_M), \mathbf{x}_m = (x_{m+1}, \ldots, x_M)$ and, for $1 \leq j \leq m$,

$$(\mathbf{w}_j)_k = \begin{cases} (v_k - v_j)/(\mu_k - \mu_j) & \text{if } 1 \leq k < j \\ (v_{k+1} - v_j)/(\mu_{k+1} - \mu_j) & \text{if } j \leq k < m \end{cases}$$

7.2 Numerical validation

We now use the distributed tool described in Section 5 to numerically compute the cycle time density for a path in the closed tree-like network shown in Figure 6. The results are then compared with the analytical results of the previous section.

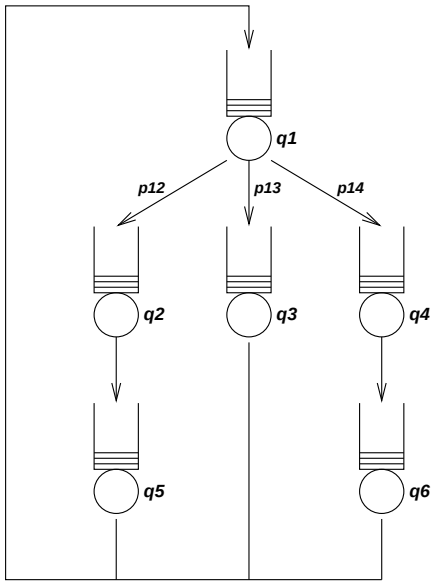


Figure 6: A tree-like network and its routing probabilities

This network has six servers with rates $\mu_1, \ldots, \mu_6$, non-zero routing probabilities as shown and variable customer population. Thus the visitation rates $v_1, \ldots, v_6$ for servers 1 to 6 are respectively proportional to: $1,\ p_{12},\ p_{13},\ p_{14},\ p_{12},\ p_{14}$.

The graph on the left in Fig. 7 presents the numerical and analytical cycle time distributions for an 18 customer system. Here $\{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6\} = \{3, 5, 4, 6, 2, 1\}$ and $\{p_{12}, p_{13}, p_{14}\} = \{0.2, 0.5, 0.3\}$. In order to track a tagged customer through the system, the state vector is augmented by 3 extra components: the queue containing the tagged customer, the position of the tagged customer in that queue, and the cycle sequence number (an alternating bit, flipped when the tagged customer joins $q1$). Because of the job observer property (see, for example, [11] pp. 241), the initial

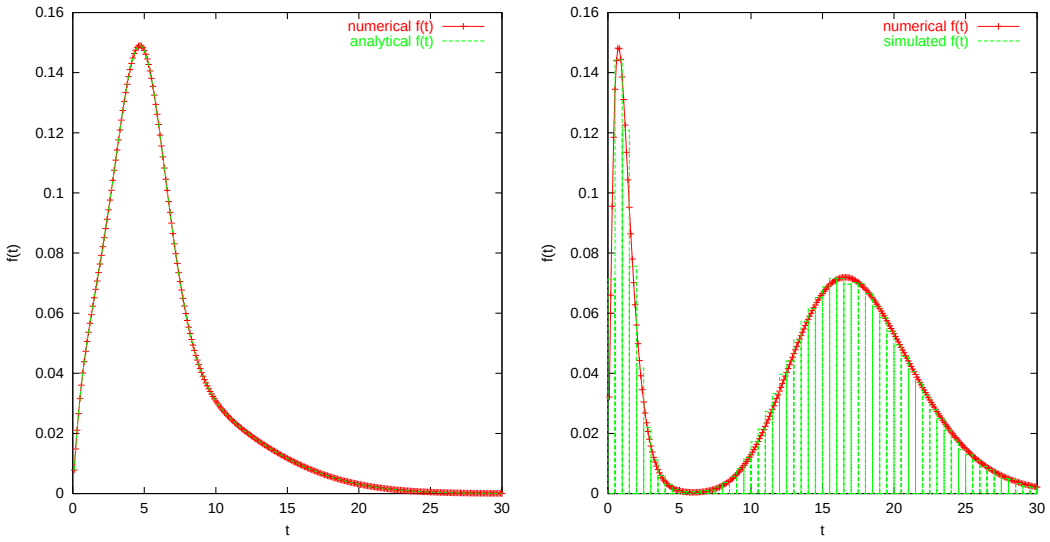


Figure 7: Numerical and analytical cycle time distribution for the tree-like queueing network of Fig. 6 with 18 customers (left) and numerical and simulated cycle time distribution for 18 customers in the same queueing network when overtaking between $q3$ and $q6$ is introduced (right).

states are weighted according to their steady state probabilities in the same network with population 17. The Markov Chain underlying the augmented model has 1 211 364 states and 8.5 million transitions, and the numerical cycle time distribution was calculated in 26 minutes using 32 slave PCs (each of which has a 1.4GHz AMD Athlon processor and 256MB RAM). No scaling was required and 59 Laguerre coefficients were used. Agreement between the numerical and analytical cycle time distributions is excellent, with relative errors of under 0.0001% well into the tail. The corresponding first three moments of the cycle time can be calculated by the moments analyser in 5 minutes 3 seconds on a single PC, and are 6.12717, 53.3067 and 612.887.

The graph on the right of Fig. 7 shows the effect of introducing an overtaking path between nodes 3 and 6 such that $p_{36} = 0.9$ and $p_{31} = 0.1$. The resulting distribution cannot be calculated with known theory, but can be numerically determined using our technique. The numerical results compare favourably with a simulated cycle time curve. Now the corresponding cycle time moments are calculated in 3 minutes 53 seconds as 13.5, 245.717 and 4812.96.

7.3 Applications to Stochastic Petri nets

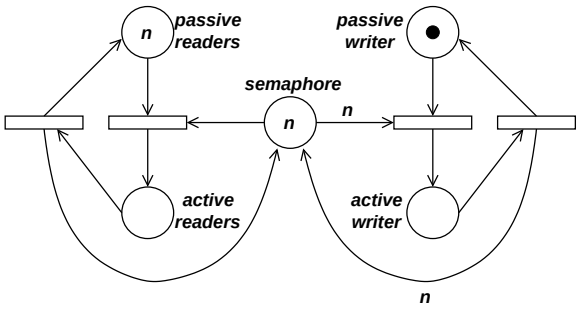


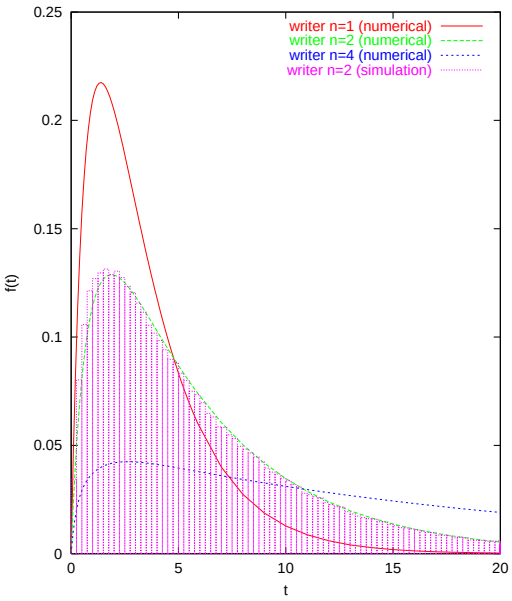
Figure 8: A Stochastic Petri net of the readers-writers problem

Our approach can equally be applied to other high-level formalisms whose state spaces map onto Markov chains. As an example, Fig. 8 shows a Petri net model of the readers-writers problem. Fig. 9 shows the resulting distribution and moments of the cycle time for various numbers of readers. The well-known problem of writer starvation as the number of readers increases is very evident. A simulation of the cycle time distribution is used to validate our results for the case of 2 readers.

8. CONCLUSION

We have developed numerical algorithms based on Laplace transforms which compute moments and density functions of first passage time distributions in large Markov chains. Hitherto progress in this area has been limited and the problem largely unattempted due to a lack of widely available computing power. Using contemporary high-performance parallel computer systems together with tailored linear solvers, the moments of passage times can now be obtained for Markov chains with over 100 million states. Furthermore, our highly scalable and checkpointed distributed algorithm of Section 5.1 makes it possible to compute passage time density functions for systems with 20 million states, using only a vanilla networked PC cluster. This degree of scalability is made possible by our choice of the Laguerre algorithm for Laplace transform inversion, which uniquely uses a relatively small number of pre-computed Laplace transform evaluations at every t point. We have further reduced the number of pre-computed evaluations required through our customized computation of the Laguerre coefficients described in Section 4.3. The alternative approach of using uniformization to compute passage time densities is much less scalable (being based on matrix multiplication operations) and does not extend to semi-Markov processes.

We have successfully implemented a complete passage time analysis pipeline and validated our results on Markov chains arising from both queueing networks and Petri nets. In our examples, we have used an in-core linear solver to deal with Markov chains of up to 1.2 million states on a 256MB machine. With the implementation of a standard out-of-core linear solver, 20 million states would be attainable; above this a parallel linear solver is needed.



	$E[T]$	$E[T^2]$	$E[T^3]$
$n = 1$	3.57	20	162
$n = 2$	6.25	68.75	1111.5
$n = 4$	21.0417	858.319	52447.5

Figure 9: Distribution of writer cycle time for various numbers of readers (n) (left) and corresponding cycle time moments (right)

Using state vector augmentation, response times specified in a high-level formalism can be analysed by mapping onto first passage times in the underlying Markov chain. We have tool support for specifying augmented state vectors, initial and target states, and the range of t-values over which the density should be computed.

Finally, we have shown in principle how to extend the theory to semi-Markov chains. This should find application in, amongst other areas, the analysis of passage times in Generalized Stochastic Petri net models.

9. ACKNOWLEDGEMENTS

The authors would like to thank their M.Sc. student Teresa Au-Young for bringing the Laguerre method to their attention in her M.Sc. thesis "Numerical Techniques for Laplace Transform Inversion".

10. REFERENCES

[1] J. Abate, G. Choudhury, and W. Whitt. On the Laguerre method for numerically inverting Laplace transforms. *INFORMS Journal on Computing*, 8(4):413–427, 1996.

[2] J. Abate, G. Choudhury, and W. Whitt. An introduction to numerical transform inversion and its application to probability models. In W. Grassman, editor, *Computational Probability*, pages 257–323, Kluwer, Boston, 2000.

[3] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10(1):5–88, 1992.

[4] J. Abate and W. Whitt. Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, 7(1):36–43, 1995.

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

[6] J. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16:527–531, 1973.

[7] D. Deavours and W. Sanders. An efficient disk-based tool for solving large Markov models. *Performance Evaluation*, 33(1):67–84, June 1998.

[8] H. Duduna. Passage times for overtake-free paths in Gordon-Newell networks. *Advances in Applied Probability*, 14:672–686, 1982.

[9] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, Massachussetts, 1994.

[10] P. Harrison. Laplace transform inversion and passage-time distributions in Markov processes. *Journal of Applied Probability*, 27:74–87, 1990.

[11] P. Harrison and N. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. International Computer Science Series. Addison Wesley, 1993.

[12] J. Keilson and W. Nunn. Laguerre transformation as a tool for the numerical solution of integral equations of convolution type. *Applied Mathematics and Computation*, 5:313–359, 1979.

[13] F. Kelly and P. Pollett. Sojourn times in closed queueing networks. *Advances in Applied Probability*, 15:638–656, 1983.

[14] W. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, Cape Town, South Africa, July 1996.

[15] W. Knottenbelt. *Parallel Performance Analysis of Large Markov Models*. PhD thesis, Imperial College, London, United Kingdom, February 2000.

[16] W. Knottenbelt and P. Harrison. Distributed disk-based solution techniques for large Markov models. In *Proceedings of the 3rd International Meeting on the Numerical Solution of Markov Chains (NSMC '99)*, pages 58–75, Zaragoza, Spain, September 1999.

[17] W. Knottenbelt, P. Harrison, M. Mestern, and P. Kritzinger. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation*, 39(1–4):127–148, February 2000.

[18] W. Weeks. Numerical inversion of Laplace transforms using Laguerre functions. *Journal of the ACM*, 13:419–426, 1966.