RESPONSE TIME DENSITIES AND QUANTILES IN LARGE MARKOV AND SEMI-MARKOV MODELS

JEREMY T. BRADLEY NICHOLAS J. DINGLE ULI HARDER PETER G. HARRISON
WILLIAM J. KNOTTENBELT*

Abstract. Response time quantiles reflect user-perceived quality of service more accurately than mean or average response time measures. Consequently, on-line transaction processing benchmarks, telecommunications Service Level Agreements and emergency services legislation all feature stringent 90th percentile response time targets. This chapter describes a range of techniques for extracting response time densities and quantiles from large-scale Markov and semi-Markov models of real-life systems. We describe a method for the computation of response time densities or cumulative distribution functions which centres on the calculation and subsequent numerical inversion of their Laplace transforms. This can be applied to both Markov and semi-Markov models. We also review the use of uniformization to calculate such measures more efficiently in purely Markovian models. We demonstrate these techniques by using them to generate response time quantiles in a semi-Markov model of a high-availability web-server. We show how these techniques can be used to analyse models with state spaces of $O(10^7)$ states and above.

1. Introduction. A fast response time is an important performance criterion for almost all computer-communication and transaction processing systems. Examples of systems with stringent response time requirements include stock market trading systems, mobile communication systems, web servers, database servers, manufacturing systems, communication protocols and communications networks. Typically, response time targets are specified in terms of quantiles (percentiles). For example, in a mobile messaging system it might be required that "there should be a 95% probability that a text message will be delivered within 3 seconds".

In another context the Transaction Processing Performance Council (TPC) benchmarks [51] were conceived to compare different implementations of large-scale on-line transaction processing (OLTP) systems in a consistent way. A range of benchmarks are available, each of which is suitable for different applications including transaction processing, decision support, business reporting and e-Commerce [51].

Response times are also used for Service Level Agreements (SLAs). SLAs exist as contracts between service providers and their customers [20, 25]. For example, an e-commerce site may have an SLA with the company which hosts its website, or two Internet Service Providers (ISPs) may have mutual SLAs to regulate the carrying of each other's traffic. A typical SLA specifies the level of service to be provided (according to metrics such as availability, response time, latency, packet loss and so forth) and how much this will cost, as well describing what financial penalties will be incurred if this level is not met. It should also describe what level of technical support will be given to the customer in the event of problems.

Usually, the main metric of interest to customers is the availability of the provider's service (e.g. network or server uptime). However, customers (particularly those involved in web-commerce) often require response time guarantees as well [25].

Response time percentiles are also used by governmental organisations when measuring the effectiveness of emergency services. Indeed, in Ontario, Canada, it is a legal requirement to report 90th percentile response times for ambulance services [19, 41, 50]: Similar reporting takes place in Australia [6] and San Francisco [48]. In the UK, the London Ambulance Service aims to have an ambulance at the scene of 75% percent of life-threatening incidents within 8 minutes [38] while the National Health Service aims to see 90% of accident and emergency patients within 4 hours [18].

As can be seen from the above examples, it is important to ensure that systems will meet quality of service targets expressed in terms of response time quantiles. Ideally, it should be possible to determine whether or not this will be the case at design time. This can be achieved through the modelling and analysis of the system in question. Such analysis is usually conducted by capturing the behaviour of the system with a formal model; that is, identifying the possible states the system may be in and the way in which it can move between these states. The concept

*Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ, UK. Email: {jb,njd200,uh,pgh,wjk}@doc.ic.ac.uk

of time can be introduced by associating delays with the state transitions. The result is that a certain amount of time will be spent in a state before moving to another, and we term this the *state sojourn time*. When the choice of the next state depends only on the current state and state sojourn times are random numbers sampled from the negative exponential distribution, we call such a model a *continuous-time Markov chain*.

As specifying every state and transition in the state space of a complex model of a real-life system is infeasible, high-level formalisms such as stochastic Petri nets [7], stochastic process algebras [32] and queueing networks [44] can be employed. These permit a succinct description of the model from which a Markov chain can automatically be extracted and then solved for performance measures of interest. From the equilibrium (steady-state) probability distribution of the model's underlying Markov chain, standard resource-based performance measures, such as mean buffer occupancy, system availability and throughput, and *expected* values of various sojourn times can be obtained. There is a large body of previous work on the efficient calculation of steady-state probabilities in large Markov chains, including parallel [8, 15, 34] and disk-based [21, 35, 36] implementations, as well as those which employ implicit state space representation techniques [16, 22, 31, 37]. Steady-state measures allow the answering of questions such as: "What is the probability that the system will be in a failure state in the long run?" and "What is the average utilisation of this resource?".

The focus of this chapter, however, is on the harder problem of calculating full response time densities in very large Markov models and semi-Markov models (a generalisation of Markov models in which state sojourn times can have an arbitrary distribution). As we have seen, the answers to response time questions provide greater insight into whether or not a system meets its user requirements than steady-state probabilities. In the context of high-level models, response times can be specified as *passage* times in the model's underlying Markov or semi-Markov chain – that is, the time taken to enter any one of a set of target states having started from a specified set of source states.

In the past, numerical computation of analytical passage time densities has proved prohibitively expensive except in some Markovian systems with restricted structure such as overtake-free tree-like queueing networks [29]. However, with the advent of high-performance parallel computing and the widespread availability of PC clusters, direct numerical analysis of Markov chains has now become a practical proposition. There are two main analytical methods for computing first passage time (and hence response time) densities in Markov chains: those based on Laplace transforms and their inversion [30] and those based on uniformization [39, 40, 42]. The former has wider application to semi-Markov processes (with generally-distributed state holding-times) but is less efficient than uniformization when restricted to Markov chains.

In general, the probability density function of the time taken to move from a set of source states to a set of target states is calculated by convolving the state-holding time functions along all possible paths between the two sets of states. To convolve two functions together directly requires the evaluation of an integral, and the convolution across a path n states long requires the evaluation of an $(n-1)$ dimensional integral. To perform such a calculation for large values of n (perhaps in the millions) would therefore be impractical. Instead, we make use of Laplace transforms, which uniquely map a real-valued function (e.g. a probability density function) to a function of a complex variable. We do this as we wish to exploit the convolution property of Laplace transforms, which states that the Laplace transform of the convolution of two functions is the product of the functions' individual Laplace transforms. Once the Laplace transform of the passage time measure has been calculated it is possible to retrieve the corresponding density function using a process known as *Laplace transform inversion*. A number of numerical techniques are available to accomplish this.

Although all state holding-times in Markov models are exponentially distributed, this does not make the direct calculation of their convolutions significantly easier as the rate parameters of the state holding-time distributions will usually be different for different states. An alternative technique known as uniformization can, however, be employed. This transforms the model's underlying continuous-time Markov chain with distinct exit rates into an equivalent one where all delay rate parameters are identical. The passage time density across any number of these states

can therefore be calculated easily because the convolution of exponential delays with the same rate parameter is simply an Erlang distribution.

As semi-Markov processes do not have identically distributed state holding-time functions, uniformization cannot be applied to calculate passage time measures in such processes. Until recently, very little work had been done on the problem of calculating passage time densities and distributions in semi-Markov models, and what had been done was limited to applying analytical techniques to models with small state spaces (of the order of 10^1 to 10^4 states) [27, 33]. In this chapter, we present a synopsis of our recent work in the field of response time and transient analysis techniques for large Markov [30, 23] and semi-Markov chains [13, 12].

2. Background.

2.1. Stochastic Processes. At the lowest level, the performance modelling of a system can be accomplished by identifying all possible configurations (or *states*) that the system can enter and describing the ways in which the system can move between those states. This is termed the *state-transition* level behaviour of the model, and the changes in state as time progresses describe a *stochastic process*. In this section, we focus on those stochastic processes which belong to the class known as *Markov processes*, specifically continuous-time Markov chains (CTMCs) and the more general semi-Markov processes (SMPs).

Consider a random variable χ which takes on different values at different times t. The sequence of random variables $\chi(t)$ is said to be a stochastic process. The different values which members of the sequence $\chi(t)$ can take (also referred to as *states*) all belong to the same set known as the *state space* of $\chi(t)$.

A stochastic process can therefore be classified by the nature of its state space and of its time parameter. If the values in the state space of $\chi(t)$ are finite or countably infinite, then the stochastic process is said to have a *discrete state space* (and may also be referred to as a *chain*). Otherwise, the state space is said to be *continuous*. Similarly, if the times at which $\chi(t)$ is observed are also countable, the process is said to be a *discrete time* process. Otherwise, the process is said to be a *continuous time* process. In this chapter, all stochastic processes considered have discrete and finite state spaces, and we focus mainly on those which evolve in continuous time (although some consideration is also given to the solution of discrete time chains).

DEFINITION 2.1. *A* Markov *process is a stochastic process in which the* Markov property *holds. Given that $\chi(t) = x_t$ indicates that the state of the process $\chi(t)$ at time t is x_t, this property stipulates that:*

$$\mathbb{P}\big(\chi(t) = x \mid \chi(t_n) = x_n, \chi(t_{n-1}) = x_{n-1}, \ldots, \chi(t_0) = x_0\big) = \mathbb{P}\big(\chi(t) = x \mid \chi(t_n) = x_n\big) \qquad t > t_n > t_{n-1} > \ldots > t_0$$

That is, the future evolution of the system depends only on the current state and not on any prior states.

DEFINITION 2.2. *A Markov process is said to be* homogeneous *if it is invariant to shifts in time, that is:*

$$\mathbb{P}\big(\chi(t+s) = x \mid \chi(t_n+s) = x_n\big) = \mathbb{P}\big(\chi(t) = x \mid \chi(t_n) = x_n\big)$$

2.1.1. Continuous-Time Markov Chains. There also exists a family of Markov processes with discrete state spaces but whose transitions can occur at arbitrary points in time; we call these continuous-time Markov chains (CTMCs). The definitions above for homogeneity, aperiodicity and irreducibility in DTMCs also hold for CTMCs. An homogeneous N-state $\{1, 2, \ldots, N\}$ CTMC has state at time t denoted $\chi(t)$. Its evolution is described by an $N \times N$ generator matrix $\mathbf{Q}$, where q_{ij} is the infinitesimal rate of moving from state i to state j ($i \neq j$), and $q_{ii} = -\sum_{i \neq j} q_{ij}$.

The Markov property imposes the restriction on the distribution of the sojourn times in states in a CTMC that they must be *memoryless* – the future evolution of the system therefore does not depend on the evolution of the system up until the current state, nor does it depend on how long

has already been spent in the current state. This means that the sojourn time ν in any state must satisfy:

$$\mathbb{P}(\nu \geq s + t \mid \nu \geq t) = \mathbb{P}(\nu \geq s) \tag{2.1}$$

A consequence of Equation (2.1) is that all sojourn times in a CTMC must be exponentially distributed. The rate out of state i, and therefore the parameter of the sojourn time distribution, is μ_i and is equal to the sum of all rates out of state i, that is $\mu_i = -q_{ii}$. This means that the density function of the sojourn time in state i is $f_i(t) = \mu_i\, e^{-\mu_i t}$ and the average sojourn time in state i is μ_i^{-1}.

We define the steady-state distribution for a CTMC in a similar manner as for a DTMC. Once again, we denote the set of steady-state probabilities as $\{\pi_j\}$.

Definition 2.3. *In a CTMC which has all states recurrent non-null and which is irreducible and homogeneous, the* limiting *or* steady-state probability distribution $\{\pi_j\}$ *is given by [7]:*

$$\pi_j = \lim_{t\to\infty} \mathbb{P}(\chi(t) = j \mid \chi(0) = i)$$

Theorem 2.4. *For an finite, irreducible and homogeneous CTMC, the steady-state probabilities $\{\pi_j\}$ always exist and are independent of the initial state distribution. They are uniquely given by the solution of the equations:*

$$-q_{jj}\pi_j + \sum_{k\neq j} q_{kj}\pi_k = 0 \quad \textit{subject to} \quad \sum_i \pi_i = 1$$

Again, this can be expressed in matrix vector form (in terms of the vector $\boldsymbol{\pi}$ with elements $\{\pi_1, \pi_2, \ldots, \pi_N\}$ and the matrix $\mathbf{Q}$ defined above) as:

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$$

A CTMC also has an embedded discrete-time Markov chain (EMC) which describes the behaviour of the chain at state-transition instants, that is to say the probability that the next state is j given that the current state is i. The EMC of a CTMC has a one-step $N \times N$ transition matrix $\mathbf{P}$ where $p_{ij} = -q_{ij}/q_{ii}$ for $i \neq j$ and $p_{ij} = 0$ for $i = j$.

2.1.2. Semi-Markov Processes. Semi-Markov Processes (SMPs) are an extension of Markov processes which allow for generally distributed sojourn times. Although the memoryless property no longer holds for state sojourn times, at transition instants SMPs still behave in the same way as Markov processes (that is to say, the choice of the next state is based only on the current state) and so share some of their analytical tractability.

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

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

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

$$Z(t) = \chi_{N(t)}$$

where $N(t) = \max\{n : T_n \leq t\}$ is 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 only time-homogeneous SMPs in which $R(n, i, j, t)$ is independent of n:

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

where $p_{ij} = \mathbb{P}(\chi_{n+1} = j \mid \chi_n = i)$ is the state transition probability between states i and j and $H_{ij}(t) = \mathbb{P}(T_{n+1} - T_n \leq t \mid \chi_{n+1} = j, \chi_n = i)$, is the sojourn time distribution in state i when the next state is j. An SMP can therefore be characterised by two matrices $\mathbf{P}$ and $\mathbf{H}$ with elements p_{ij} and H_{ij} respectively.

Semi-Markov processes can be analysed for steady-state performance metrics in the same manner as DTMCs and CTMCs. To do this, we need to know the steady-state probabilities of the SMP's EMC and the average time spent in each state. The first of these can be calculated by solving $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$, as in the case of the DTMC. The average time in state i, $\mathbb{E}(\tau_i)$, is the weighted sum of the averages of the sojourn time in the state i when going to state j, $\mathbb{E}(\tau_{ij})$, for all successor states j of i, that is:

$$\mathbb{E}(\tau_i) = \sum_j p_{ij}\mathbb{E}(\tau_{ij})$$

The steady-state probability of being in state i of the SMP is then [7]:

$$\phi_i = \frac{\pi_i \mathbb{E}(\tau_i)}{\sum_{r=1}^{N} \pi_r \mathbb{E}(\tau_r)} \tag{2.2}$$

That is, the probability of finding the SMP in state i is the probability of its EMC being in state i multiplied by the average amount of time the SMP spends in state i, normalised over the mean total time spent in all of the states of the SMP.

2.2. Semi-Markov Stochastic Petri Nets. The motivation behind semi-Markov stochastic Petri Nets (SM-SPNs) [10, 11] is as a specification formalism and higher-level abstraction for semi-Markov models. It is important to note that SM-SPNs do not try to tackle the issue of concurrently enabled generally distributed (GEN) transitions in the most general case. If more that one GEN transition is enabled then a probabilistic choice is used to determine which will be fired. Pre-empted GEN transition use a *prd* schedule if they later become re-enabled. This approach is correctly described in [45] as not being a solution to the more complex issue of properly concurrently enabled GEN transitions, but is merely a way of specifying a different type of model – a semi-Markov model where GEN transitions are essentially forced to be exclusive. Where concurrently enabled GEN transitions do not occur then proper concurrent and competitive transition behaviour is catered for with full *prs* scheduling for pre-empted transitions.

SM-SPNs are extensions of GSPNs [5] which support arbitrary marking-dependent holding-time distributions and generate an underlying semi-Markov process rather than a Markov process. An SM-SPN is defined formally as follows:

Definition 2.5.

An SM-SPN is 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 is 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.*
- *$\mathcal{P} : T \times \mathcal{M} \to \mathbb{N}_0$, denoted $p_t(m)$, is a marking-dependent priority function for a transition.*
- *$\mathcal{W} : T \times \mathcal{M} \to \mathbb{R}^+$, denoted $w_t(m)$, is a marking-dependent weight function for a transition, to allow implementation of probabilistic choice.*
- *$\mathcal{D} : T \times \mathcal{M} \to (\mathbb{R}^+ \to [0, 1])$, 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 markings for a given net. Further, we define the following net-enabling functions:

Definition 2.6.

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

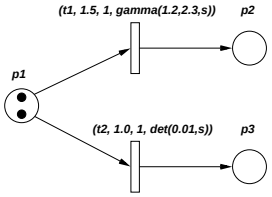

Fig. 2.1. An example Semi-Markov Stochastic Petri Net (SM-SPN) [11].

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. The more stringent priority-enabling function $\mathcal{E}_P(m)$ is defined for a given marking m which selects only those net-enabled transitions that have the highest priority, that is:

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

For a given priority-enabled transition, $t \in \mathcal{E}_P(m)$, the probability that it will be the one that actually fires (after a delay sampled from its firing distribution $d_t(m)$) is a probabilistic choice based on the relative weights of all enabled transitions:

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

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.

To illustrate this enabling and firing strategy, Figure 2.1 shows an enabled pair of GEN transitions in an SM-SPN. Transition t_1 has a weight of 1.5, a priority of 1 and a $gamma(1.2, 2.3)$ firing distribution, while t_2 has a weight of 1.0, a priority of 1 and a det(0.01) firing distribution. Graphically, each transition is annotated with a 4-tuple specifying the transition name, weight, priority and Laplace transform of its firing time distribution. The weights are used to select which GEN transition will fire: in this case t_1 will be selected to fire with probability $1.5/(1.0+1.5) = 0.6$ and p_2 with probability 0.4. After a delay sampled from the selected transition's firing-time distribution, the probabilistic selection takes place again (for the remaining token on p_1); this is followed by another sampled delay.

3. Laplace Transforms. The Laplace transform is an integral transform which is widely used in the solution of differential equations arising from physical problems. When the Laplace transform $f^*(s)$ of a real-valued function $f(t)$ exists, it is uniquely given by:

$$f^*(s) = \int_0^\infty e^{-st} f(t)\,\mathrm{d}t \tag{3.1}$$

where s is a complex number. For the Laplace transform of a function $f(t)$ to exist, $f(t)$ must be of *exponential order*. Examples of functions which meet this restriction are polynomial or exponential (those of the form e^{kt}) functions and bounded functions. Also included are those functions with a finite number of finite discontinuities. Examples of functions which do not fall within this category are those which have singularities (e.g. $ln(x)$), those whose growth rates are faster than exponential (e.g. e^{x^2}) or those with an infinite number of finite discontinuities (e.g. $f(x) = 1$ if x is rational and 0 otherwise). In the context used here all functions under consideration are *well-behaved*, though the transform may not exist in closed form (e.g. for the Weibull distribution). The Laplace transform is a linear transformation from the t-space to the s-space of functions.

THEOREM 3.1. Linearity: *If $f(t)$ and $g(t)$ are functions whose Laplace transforms exist, then:*

$$L\{af(t) + bg(t)\} = aL\{f(t)\} + bL\{g(t)\}$$

for any complex numbers a and b. The other property that immediately indicates that the Laplace transform is useful to solve linear differential equations, is that integration w.r.t. t in t-space is equivalent to division by s in s-space. We make use of this fact to compute cumulative distribution functions.

THEOREM 3.2. Integration: *If $f(t)$ is a probability density function and $F(t)$ is the corresponding cumulative distribution function, $\int_0^\infty f(t)\,\mathrm{d}t = F(t)$. The Laplace transform of $F(t)$ can be calculated from the Laplace transform of $f(t)$ by dividing $L\{f(t)\}$ by s:*

$$L\{F(t)\} = L\{f(t)\}/s$$

Finally, a convolution in t-space corresponds to multiplication in s-space. The convolution of two functions $f(t)$ and $g(t)$ denoted $f(t) * g(t)$ is given by:

$$f(t) * g(t) = \int_0^t f(\tau)\, g(t-\tau)\,\mathrm{d}\tau \tag{3.2}$$

THEOREM 3.3. Convolution: *The Laplace transform of the convolution of two functions $f(t)$ and $g(t)$, denoted $L\{f(t) * g(t)\}$, is the product of the Laplace transforms of the two functions, that is:*

$$L\{f(t) * g(t)\} = f^*(s)\, g^*(s)$$

This is especially useful as the theorem holds for convolutions of n functions. In this case the integral becomes n-dimensional. The calculation of the probability density function of a passage time between two states is achieved by convolving the probability density functions of the sojourn times of the states along all the paths between the source and target states.

3.1. Laplace Transform Inversion. As the Laplace transform of a function is unique it is possible to recover the function $f(t)$ from its Laplace transform $f^*(s)$. This process is called Laplace transform inversion. The inverse of the Laplace transform $f^*(s)$ of a function $f(t)$ (which we denote $L^{-1}\{f^*(s)\}$) is the function $f(t)$ itself:

$$L^{-1}\{f^*(s)\} = f(t) = \frac{1}{2\pi i}\int_{a-i\infty}^{a+i\infty} e^{st} f^*(s)\,\mathrm{d}s \tag{3.3}$$

where a is a real number which lies to the right of all the singularities of $f^*(s)$. This is also known as the *Bromwich contour inversion integral.*

As with Laplace transforms, inverse Laplace transforms display linearity:

$$L^{-1}\{af^*(s) + bg^*(s)\} = aL^{-1}\{f^*(s)\} + bL^{-1}\{g^*(s)\}$$

The work in this chapter centres around the calculation and *numerical* inversion of Laplace transforms. There are a number of numerical Laplace transform inversion algorithms in the literature, for example the Euler technique [3, 4] and the Laguerre method [1] (also known as Weeks' method [52]). We now summarise the key features of these methods.

3.1.1. Summary of Euler Inversion. It is possible to rewrite Equation (3.3) such that it is possible to obtain $f(t)$ from $f^*(s)$ by integrating a real-valued function of a real variable rather than requiring contour integration to be performed in complex space. Substituting $s = a + iu$ allows Equation (3.3) to be rewritten as [2]:

$$f(t) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} e^{(a+iu)t} f^*(a+iu)\,\mathrm{d}u$$

Making use of the fact that:

$$e^{(b+iu)t} = e^{bt}(\cos ut + i\sin ut)$$

yields [2]:

$$f(t) = \frac{2e^{at}}{\pi} \int_0^\infty \mathrm{Re}(f^*(a+iu))\, \cos(ut)\, \mathrm{d}u$$

This integral can be evaluated numerically using the trapezoidal rule. This approximates the integral of a function $f(t)$ over the interval $[a, b]$ as:

$$\int_a^b f(t)\, \mathrm{d}t \approx h \left(\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a+kh) \right)$$

where $h = (b-a)/n$. Setting the step-size $h = \pi/2t$ and $a = A/2t$ (where A is a constant that controls the discretisation error and is set to 19.1 in [4]) results in the alternating series [4, 26]:

$$f(t) \approx \frac{e^{A/2}}{2t} \mathrm{Re}\left(f^* \left(\frac{A}{2t} \right) \right) + \frac{e^{A/2}}{2t} \sum_{k=1}^{\infty} (-1)^k\, \mathrm{Re}\left(f^* \left(\frac{A+2k\pi i}{2t} \right) \right)$$

Euler summation can be employed to accelerate the convergence of this alternating series [49]. That is, we calculate the sum of the first n terms explicitly and use Euler summation to calculate the next m. The mth term after the first n is given by [3]:

$$E(t, m, n) = \sum_{k=0}^{m} \binom{m}{k} 2^{-m}\, s_{n+k}(t) \tag{3.4}$$

In Equation (3.4):

$$s_n(t) = \sum_{k=0}^{n} (-1)^k\, \mathrm{Re}\left(f^* \left(\frac{A+2k\pi i}{2t} \right) \right)$$

An estimate of the truncation error incurred in using Euler summation can be calculated by comparing the magnitudes of the nth and $(n+1)$th terms, i.e. [3]:

$$|E(t, m, n) - E(t, m, n+1)|$$

3.1.2. Summary of Laguerre Inversion. The Laguerre method [1] makes use of the Laguerre series representation of $f(t)$:

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

where the Laguerre polynomials l_n are given by:

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

starting with $l_0 = e^{t/2}$ and $l_1 = (1-t)e^{t/2}$, and:

$$q_n = \frac{1}{2\pi r^n} \int_0^{2\pi} Q(re^{iu}) e^{-inu}\, \mathrm{d}u \tag{3.5}$$

where $r = (0.1)^{4/n}$ and $Q(z) = (1-z)^{-1} f^*\big((1+z)/2(1-z)\big)$.

The integral in the calculation of Equation (3.5) can be approximated numerically using the trapezoidal rule, giving:

$$q_n \approx \frac{1}{2nr^n} \left(Q(r) + (-1)^n Q(-r) + 2 \sum_{j=1}^{n-1} (-1)^j \mathrm{Re}\left(Q(re^{\pi j i/n}) \right) \right) \tag{3.6}$$

As described in [30], the Laguerre method can be modified by noting that the Laguerre coefficients q_n are independent of t. Since $|l_n(t)| \leq 1$ for all n, the convergence of the Laguerre series depends on the decay rate of q_n as $n \to \infty$ which is in turn determined by the smoothness of $f(t)$ and its derivatives [1]. Slow convergence of the q_n coefficients can often be improved by exponential dampening and scaling using two real parameters σ and b [52]. Here the inversion algorithm is applied to the function

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

with $f(t)$ being recovered as:

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

Each q_n coefficient is computed as in Equation (3.6), using the trapezoidal rule with $2n$ trapezoids. However, if we apply scaling 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 when calculating each q_n. This allows us to calculate each q_n with high accuracy 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 $f^*(s)$, we obtain $f(t)$ at an arbitrary number of t-values at the fixed cost of evaluating $Q(z)$ (and hence $f^*(s)$) $2p_0$ times.

Suitable scaling parameters can be automatically determined using the algorithm in [30]. 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 the observation in [30] that excessive values of the damping parameter σ can lead to numerical instability in finite precision arithmetic.

This fixed computational cost for any number of t-points is in contrast to the Euler method, where the number of different s-values at which $f^*(s)$ must be evaluated is a function of the number of points at which the value of $f(t)$ is required (in fact, it is $(n + m + 1)$ times the number of t-points). It must be noted, however, that the Euler method can be used to invert Laplace transforms which are not sufficiently smooth to permit the modified Laguerre method to be used. This situation typically arises when the original function $f(t)$ has discontinuities (e.g. if $f(t) = \det(x)$).

4. Models. In this section, we present the Markov and semi-Markov models which are used as running examples throughout the rest of the chapter.

4.1. Web Content Authoring System. Figure 4.1(a) represents an SM-SPN model of a web server with RR clients (readers), WW web content authors (writers), SS parallel web servers and a write-buffer of BB in size [13]. As with the Voting model, the size and complexity of the underlying semi-Markov chain can be varied by altering these four parameters as shown in Table 4.1.

Clients can make read requests to one of the web servers for content (represented by the movement of tokens from p_8 to p_7). Web content authors submit page updates into the write buffer (represented by the movement of tokens from p_1 onto p_2 and p_4), and whenever there are no outstanding read requests all outstanding write requests in the buffer (represented by tokens on p_4) are applied to all functioning web servers (represented by tokens on p_6). Web servers can fail (represented by the movement of tokens from p_6 to p_5) and institute self-recovery unless all servers fail, in which case a high-priority recovery mode is initiated to restore all servers to a fully functional state. Complete reads and updates are represented by tokens on p_9 and p_2 respectively.

4.2. Flexible Manufacturing System. Figure 4.1(b) shows a 22-place GSPN model of a flexible manufacturing system [17]. The model describes an assembly line with three types of machines ($M1$, $M2$ and $M3$) which assemble four types of parts ($P1$, $P2$, $P3$ and $P12$). Initially, there are k unprocessed parts of each type $P1$, $P2$ and $P3$ in the system. There are no parts of type $P12$ at start-up since these are assembled from processed parts of type $P1$ and $P2$ by the machines of type $M3$. When parts of any type are finished, they are stored for shipping on places $P1s$, $P2s$, $P3s$ and $P12s$.

System	RR	WW	SS	BB	States
1	45	22	4	8	107 289
2	52	26	5	10	248 585
3	60	30	6	12	517 453
4	65	30	7	13	763 680
5	70	35	7	14	1 044 540
6	100	50	18	20	15 445 919

Table 4.1. Number of states generated by the Web-server SM-SPN in terms of the number of clients (RR), authors (WW), parallel web servers (SS) and write-buffers (BB).

4.3. Courier Communications Protocol. The GSPN shown in Figure 4.1(c) (originally presented in [53]) models the ISO Application, Session and Transport layers of the Courier sliding-window communication protocol.

Data flows from a sender (p_1 to p_{26}) to a receiver (p_{27} to p_{46}) via a network. The sender's transport layer fragments outgoing data packets; this is modelled as two paths between p_{13} and p_{35}. The path via t_8 carries all fragments before the last one through the network to p_{33}. Acknowledgements for these fragments are sent back to the sender (as signalled by the arrival of a token on p_{20}), but no data is delivered to the higher layers on the receiver side. The path via t_9 carries the last fragment of each message block. Acknowledgements for these fragments are generated and a data token is delivered to higher receiver layers via t_{27}.

The average number of data packets sent is determined by the ratio of the weights on the immediate transitions t_8 and t_9. This ratio, known as the fragmentation ratio, is given by $q_1 : q_2$ (where q_1 and q_2 are the weights associated with transitions t_8 and t_9 respectively). This number of data packets is geometrically distributed, with parameter $q_1/(q_1 + q_2)$. In our case study, we use a fragmentation ratio of one.

The transport layer is further characterised by two important parameters: the sliding window size n (p_{14}) and the transport space m (p_{17}). For our example, we set $m = 1$ and $n = 1$.The transition rates $r_1, r_2, \ldots, r_{10}$ used in the original model [53] were obtained by benchmarking a working implementation of the protocol.

4.4. Tree-like Queueing Network. Figure 4.1(d) shows a tree-like queueing network which has six servers with rates $\mu_1, \ldots, \mu_6$ and non-zero routing probabilities as shown. 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}. For this example, we set:

$$\{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6\} = \{3, 5, 4, 6, 2, 1\} \text{ and } \{p_{12}, p_{13}, p_{14}\} = \{0.2, 0.5, 0.3\}$$

Analytical results for the cycle time density in this type of overtake-free, tree-like queueing network with M servers and population n are known [29, 30]. To compute the cycle time density in this network in terms of its underlying Markov Chain using the uniformization technique described in this chapter requires the state vector to be augmented by 4 extra components so that a "tagged" customer can be followed through the system. The extra components are: the queue containing the tagged customer m, the position of the tagged customer in that queue k (with $k \geq 0$), the cycle sequence number c (an alternating bit, flipped whenever the tagged customer joins $q1$) and a flag p indicating whether or not a passage has started.

5. Passage Times in Markov Models. This section first describes the calculation of passage time densities and quantiles in continuous-time Markov chains using the Laplace transform technique presented in [30]. We also describe a second technique for the calculation of passage time densities in Markov models known as uniformization. We present a comparison of the run-time behaviour of the Laplace transform and uniformization techniques and contrast them both with simulation.

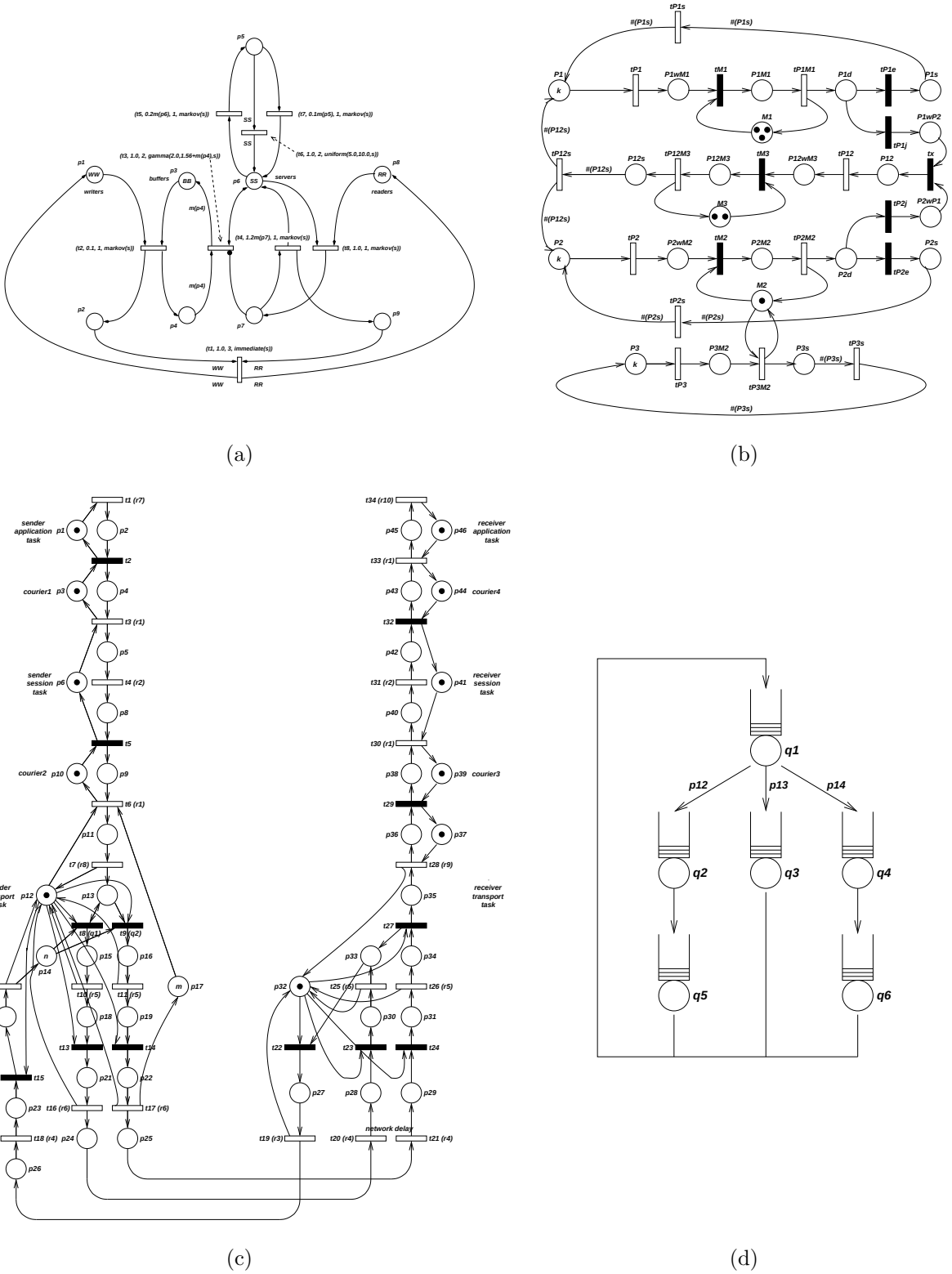

Fig. 4.1. (a) The Web-server Model SM-SPN [13]. (b) The GSPN model of a Flexible Manufacturing System [17]. (c) The Courier communications protocol GSPN model [53]. (d) The tree-like queueing network [29, 30].

5.1. The Laplace Transform Method for CTMCs. Consider a finite, irreducible CTMC with N states $\{1, 2, \ldots, N\}$ and generator matrix $\mathbf{Q}$ as defined in Section 2.1.1. As $\chi(t)$ denotes the states of the CTMC at time $t \geq 0$ and $N(t)$ denotes the number of state transitions which have occurred by time t, the first passage time from a single source marking i into a non-empty set of target markings $\vec{j}$ is:

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

When the CTMC is stationary and time-homogeneous this quantity is independent of t:

$$P_{i\vec{j}} = \inf\{u > 0 : \chi(u) \in \vec{j}, N(u) > 0, \chi(0) = i\} \tag{5.1}$$

That is, the first time the system enters a state in the set of target states $\vec{j}$, given that the system began in the source state i and at least one state transition has occurred. $P_{i\vec{j}}$ is a random variable with probability density function $f_{i\vec{j}}(t)$ such that:

$$\mathbb{P}(a < P_{i\vec{j}} < b) = \int_a^b f_{i\vec{j}}(t)\,\mathrm{d}t \quad : 0 \leq a < b$$

In order to determine $f_{i\vec{j}}(t)$ it is necessary to convolve the state holding-time density functions over all possible paths (including cycles) from state i to all of the states in $\vec{j}$.

As described in Section 3, the calculation of the convolution of two functions in t-space (cf. Equation (3.2)) can be more easily accomplished by multiplying their Laplace transforms together in s-space and inverting the result. The calculation of $f_{i\vec{j}}(t)$ is therefore achieved by calculating the Laplace transform of the convolution of the state holding times over all paths between i and $\vec{j}$ and then numerically inverting this Laplace transform.

In a CTMC all state sojourn times are exponentially distributed, so the density function of the sojourn time in state i is $\mu_i e^{-\mu_i t}$, where $\mu_i = -q_{ii}$ for $1 \leq i \leq N$ as defined in Section 2.1.1. The Laplace transform of an exponential density function with rate parameter λ is

$$L\{\lambda e^{-\lambda t}\} = \frac{\lambda}{s+\lambda}$$

Denoting the Laplace transform of the density function $f_{i\vec{j}}(t)$ of the passage time random variable $P_{i\vec{j}}$ as $L_{i\vec{j}}(s)$, we proceed by means of a first-step analysis. That is, to calculate the first passage time from state i into the set of target states $\vec{j}$, we consider moving from state i to its set of direct successor states $\vec{k}$ and thence from states in $\vec{k}$ to states in $\vec{j}$. This can be expressed as the following system of linear equations:

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} p_{ik} \left(\frac{-q_{ii}}{s - q_{ii}} \right) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} p_{ik} \left(\frac{-q_{ii}}{s - q_{ii}} \right) \tag{5.2}$$

The first term (i.e. the summation over non-target states $k \notin \vec{j}$) convolves the sojourn time density in state i with the density of the time taken for the system to evolve from state k into a target state in the set $\vec{j}$, weighted by the probability that the system transits from state i to state k. The second term (i.e. the summation over target states $k \in \vec{j}$) simply reflects the sojourn time density in state i weighted by the probability that a transition from state i into a target state k occurs.

Given that $p_{ij} = -q_{ij}/q_{ii}$ in the context of a CTMC (cf. Section 2.1.1), Equation (5.2) can be rewritten as:

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} \frac{q_{ik}}{s - q_{ii}} L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} \frac{q_{ik}}{s - q_{ii}} \tag{5.3}$$

This set of linear equations can be expressed in matrix–vector form. For example, when $\vec{j} = \{1\}$ we have:

$$\begin{pmatrix} s - q_{11} & -q_{12} & \cdots & -q_{1n} \\ 0 & s - q_{22} & \cdots & -q_{2n} \\ 0 & -q_{32} & \cdots & -q_{3n} \\ 0 & \vdots & \ddots & \vdots \\ 0 & -q_{n2} & \cdots & s - q_{nn} \end{pmatrix} \begin{pmatrix} L_{1\vec{j}}(s) \\ L_{2\vec{j}}(s) \\ L_{3\vec{j}}(s) \\ \vdots \\ L_{n\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} 0 \\ q_{21} \\ q_{31} \\ \vdots \\ q_{n1} \end{pmatrix} \tag{5.4}$$

Our formulation of the passage time quantity in Equation (5.1) states that we must observe at least one state-transition during the passage. In the case where $i \in \vec{j}$ (as for $L_{1\vec{j}}(s)$ in the above example), we therefore calculate the density of the cycle time to return to state i rather than requiring $L_{i\vec{j}}(s) = 1$.

Given a particular (complex-valued) s, Equation (5.4) can be solved for $L_{i\vec{j}}(s)$ by standard iterative numerical techniques for the solution of systems of linear equations in $\mathbf{Ax} = \mathbf{b}$ form. In the context of the inversion algorithms described in Section 3.1, both Euler and Laguerre can identify in advance the values of s at which $L_{i\vec{j}}(s)$ must be calculated in order to perform the numerical inversion. Therefore, if the algorithm requires m different values of $L_{i\vec{j}}(s)$ to calculate $f_{i\vec{j}}(t)$, Equation (5.4) will need to be solved m times.

The corresponding cumulative distribution function $F_{i\vec{j}}(t)$ of the passage time is obtained by integrating the density function. As described in Section 3, this integration can be achieved in terms of the Laplace transform of the density function with $F^*_{i\vec{j}}(s) = L_{i\vec{j}}(s)/s$. In practice, if Equation (5.4) is solved as part of the inversion process for calculating $f_{i\vec{j}}(t)$, the m values of $L_{i\vec{j}}(s)$ can be cached for future computation of $F_{i\vec{j}}(t)$ as required.

When there are multiple source markings, denoted by the vector $\vec{i}$, the Laplace transform of the response time density at equilibrium is:

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

where the weight α_k is the equilibrium probability that the state is $k \in \vec{i}$ at the starting instant of the passage. This instant is the moment of entry into state k; thus α_k is proportional to the equilibrium probability of the state k in the underlying embedded (discrete-time) Markov chain (EMC) of the CTMC with one-step transition matrix $\mathbf{P}$ as defined in Section 2.1.1. That is:

$$\alpha_k = \begin{cases} \pi_k / \sum_{j \in \vec{i}} \pi_j & : \text{if } k \in \vec{i} \\ 0 & : \text{otherwise} \end{cases} \tag{5.5}$$

where the vector $\boldsymbol{\pi}$ is any non-zero solution to $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$. The row vector with components α_k is denoted by $\boldsymbol{\alpha}$.

5.2. Extension to GSPNs. The analysis described above for Markov chains can be extended to the analysis of the underlying state spaces of GSPNs [23]. The situation is complicated, however, by the existence of two types of state (tangible and vanishing). As we are dealing with Petri nets, the analysis is described in terms of markings rather than states (although the two terms are equivalent – the state of a GSPN is defined by its marking). The underlying stochastic process for a GSPN is $\{(\chi_n, T_n) : n \geq 0\}$ where χ_n is the marking of the GSPN after the nth transition and T_n is the time of the nth transition. As with SMPs, $N(t) = \max\{n : T_n \leq t\}$ is the number of transitions that have taken place by time t.

In a GSPN, the first passage time from a single source marking i into a non-empty set of target markings $\vec{j}$ has to take into account the fact that vanishing states may exist in the set of target markings:

$$P_{i\vec{j}} = \inf\{u > 0 : N(u) \geq M_{i\vec{j}}\} \tag{5.6}$$

where $M_{i\vec{j}} = \min\{m > 0 : \chi_m \in \vec{j}, \chi_0 = i\}$ is the transition defining the terminating marking of the passage.

We proceed by means of a first-step analysis as described above for the purely Markovian case. The Laplace transform of the (exponential) sojourn time density function of tangible marking i is $\mu_i/(s+\mu_i)$, but for a vanishing marking the sojourn time is 0 with probability 1, giving a corresponding Laplace transform of 1 for all values of s. We must therefore distinguish between passage times which start in a tangible state and those which begin in a vanishing state:

$$L_{i\vec{j}}(s) = \begin{cases} \sum_{k \notin \vec{j}} \frac{q_{ik}}{s - q_{ii}} L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} \frac{q_{ik}}{s - q_{ii}} & : \text{if } i \in \mathcal{T} \\ \sum_{k \notin \vec{j}} p_{ik} L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} p_{ik} & : \text{if } i \in \mathcal{V} \end{cases} \tag{5.7}$$

Again, this system of linear equations can be expressed in matrix–vector form. For example, when $\vec{j} = \{1\}$, $\mathcal{V} = \{2\}$ and $\mathcal{T} = \{1, 3, \ldots, n\}$ the above equations can be written as:

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

This system of linear equations can then be solved by the same techniques as for the Markov case above.

As described above in Section 5.1, this formulation can easily be generalised to the case where multiple source states are required. This is accomplished by weighting the $L_{k\vec{j}}(s)$ values with the renormalised steady-state probabilities for state $k \in \vec{i}$ from the embedded Markov chain (EMC) defined by the marking of the GSPN at firing instants. Therefore:

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

where the vector $\boldsymbol{\pi}$ is any non-zero solution to $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$.

Note also that if vanishing states are eliminated from the underlying state space during its generation, the result is a continuous-time Markov chain which can then be analysed for passage times as per Section 5.1. Doing so reduces the size of the state space to be analysed but removes the ability to reason about source or target states which are vanishing.

5.3. Uniformization. As well as the Laplace transform approach described above, passage time densities and quantiles in CTMCs may also be computed through the use of *uniformization* (also known as *randomization*). This transforms a CTMC into one in which all states have the same mean holding time $1/q$, by allowing *invisible* transitions from a state to itself. This is equivalent to a discrete-time Markov chain, after normalisation of the rows, together with an associated Poisson process of rate q. The one-step transition probability matrix $\mathbf{P}$ which characterises the one-step behaviour of the uniformized DTMC is derived from the generator matrix $\mathbf{Q}$ of the CTMC as:

$$\mathbf{P} = \mathbf{Q}/q + \mathbf{I} \tag{5.8}$$

where the rate $q > \max_i |q_{ii}|$ ensures that the DTMC is aperiodic by guaranteeing that there is at least one single-step transition from a state to itself.

5.3.1. Uniformization for Transient Analysis of CTMCs. Uniformization has classically been used to conduct transient analysis of finite-state CTMCs [28, 47]. The transient state distribution of a CTMC is the probability that the process is in a state in $\vec{j}$ at time t, given that it was in state i at time 0:

$$\pi_{i\vec{j}}(t) = \mathbb{P}(\chi(t) \in j \mid \chi(0) = i)$$

where $\chi(t)$ denotes the state of the CTMC at time t.

In a uniformized CTMC [47], the probability that the process is in state j at time t is calculated by conditioning on $N(t)$, the number of transitions in the DTMC that occur in a given time interval [0,t]:

$$\pi_{i\vec{j}}(t) = \sum_{m=0}^{\infty} \mathbb{P}\big(\chi(t) \in \vec{j} \mid N(t) = m\big)\, \mathbb{P}\big(N(t) = m\big)$$

where $N(t)$ is given by a Poisson process with rate q and the state of the uniformized process at time t is denoted $\chi(t)$. Therefore:

$$\pi_{i\vec{j}}(t) = \sum_{n=1}^{\infty} \frac{(qt)^n e^{-qt}}{n!} \sum_{k \in \vec{j}} \pi_k^{(n)}$$

where:

$$\boldsymbol{\pi}^{(n+1)} = \boldsymbol{\pi}^{(n)}\mathbf{P} \quad : \text{for } n \geq 0$$

and $\boldsymbol{\pi}^{(0)}$ is the initial probability distribution from which the transient measure will be measured (typically, for a single initial state i, $\pi_k = 1$ if $k = i$, 0 otherwise).

5.3.2. Uniformization for Passage Time Analysis of CTMCs. Uniformization can also be employed for the calculation of passage time densities in Markov chains as described in [9, 39, 40, 42]. We ensure that only the first passage time density is calculated and that we do not consider the case of successive visits to a target state by making the target states absorbing. We denote by $\mathbf{P}'$ the one-step transition probability matrix of the modified, uniformized chain.

The calculation of the first passage time density between two states has two main components. The first considers the time to complete n hops ($n = 1, 2, 3, \ldots$). Recall that in the uniformized chain all transitions occur with rate q. The density of the time taken to move between two states is found by convolving the state holding-time densities along all possible paths between the states. In a standard CTMC, convolving holding times in this manner is non-trivial as, although they are all exponentially distributed, their rate parameters are different. In a CTMC which has undergone uniformization, however, all states have exponentially-distributed state holding-times with the same parameter q. This means that the convolution of n of these holding-time densities is the convolution of n exponentials all with rate q, which is an n-stage Erlang density with rate parameter q.

Secondly, it is necessary to calculate the probability that the transition between a source and target state occurs in exactly n hops of the uniformized chain, for every value of n between 1 and a maximum value m. The value of m is determined when the value of the nth Erlang density function (the left-hand term in Equation (5.9)) drops below some threshold value. After this point, further terms are deemed to add nothing significant to the passage time density and so are disregarded.

The density of the time to pass between a source state i and a target state j in a uniformized Markov chain can therefore be expressed as the sum of m n-stage Erlang densities, weighted with the probability that the chain moves from state i to state j in exactly n hops ($1 \leq n \leq m$). This can be generalised to allow for multiple target states in a straightforward manner; when there are multiple source states it is necessary to provide a probability distribution across this set of states (such as the renormalised steady-state distribution calculated below in Equation (5.11)).

The response time between the non-empty set of source states $\vec{i}$ and the non-empty set of target states $\vec{j}$ in the uniformized chain therefore has probability density function:

$$\begin{aligned} f_{\vec{i}\vec{j}}(t) &= \sum_{n=1}^{\infty} \frac{q^n t^{n-1} e^{-qt}}{(n-1)!} \sum_{k \in \vec{j}} \pi_k^{(n)} \\ &\simeq \sum_{n=1}^{m} \frac{q^n t^{n-1} e^{-qt}}{(n-1)!} \sum_{k \in \vec{j}} \pi_k^{(n)} \end{aligned} \tag{5.9}$$

where:

$$\boldsymbol{\pi}^{(n+1)} = \boldsymbol{\pi}^{(n)}\mathbf{P}' \quad : \text{for } n \geq 0 \tag{5.10}$$

with:

$$\pi_k^{(0)} = \begin{cases} 0 & : \text{if } k \notin \vec{i} \\ \pi_k / \sum_{j \in \vec{i}} \pi_j & : \text{if } k \in \vec{i} \end{cases} \tag{5.11}$$

The π_k values are the steady-state probabilities of the corresponding state k from the CTMC's embedded Markov chain. When the convergence criterion:

$$\frac{\|\boldsymbol{\pi}^{(n)} - \boldsymbol{\pi}^{(n-1)}\|_\infty}{\|\boldsymbol{\pi}^{(n)}\|_\infty} < \varepsilon \tag{5.12}$$

is met, for given tolerance ε, the vector $\boldsymbol{\pi}^{(n)}$ is considered to have converged and no further multiplications with $\mathbf{P}'$ are performed. Here, $\|\mathbf{x}\|_\infty$ is the infinity-norm given by $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

The corresponding cumulative distribution function for the passage time, $F_{\vec{i}\vec{j}}(t)$, can be calculated by substituting the cumulative distribution function for the Erlang distribution into Equation (5.9) in place of the Erlang density function term, that is:

$$\begin{aligned} F_{\vec{i}\vec{j}}(t) &= \sum_{n=1}^{\infty} \left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right) \sum_{k \in \vec{j}} \pi_k^{(n)} \\ &\simeq \sum_{n=1}^{m} \left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right) \sum_{k \in \vec{j}} \pi_k^{(n)} \end{aligned}$$

where $\boldsymbol{\pi}^{(n)}$ is defined as in Equations (5.10) and (5.11).

5.4. Comparison of Methods. As both the Laplace transform method and uniformization can be used to calculate passage time densities in Markov models, it is instructive to compare the run-time performance of the two methods along with simulation. Table 5.1 shows the run-times in seconds taken to compute the passage time densities shown in Figure 5.1 using uniformization, Laplace transform inversion and simulation. The run-times were produced on a network of PC workstations linked together by 100Mbps switched Ethernet, each PC having an Intel Pentium 4 2.0GHz processor and 512MB RAM.

Model Name	No. of States	Uniform. Run-time	Laplace Run-times						Sim. Run-time (1 run)
			1 PC	2 PCs	4 PCs	8 PCs	16 PCs	32 PCs	
Courier	11 700	1.9	42.1	30.0	22.4	19.6	18.0	23.1	3 656.0
FMS	537 768	64.8	5 096.0	2 582.6	1 298.4	675.8	398.4	182.1	2 729.6
Tree	542 638	126.2	7 555.3	4 719.3	1 921.9	993.0	550.9	398.6	1 976.8

Table 5.1. Comparison of run-time in seconds for uniformization, Laplace transform inversion and simulation passage time analysis.

Model Name	No. of States	Laplace Run-times					
		1 PC	2 PCs	4 PCs	8 PCs	16 PCs	32 PCs
Courier	29 010	542.7	293.6	170.6	145.6	166.6	232.8
FMS	2 519 580	27 593.8	13 790.2	6 961.3	3 548.9	1 933.4	1 079.7

Table 5.2. Run-time in seconds for the Laplace transform inversion method on GSPN state spaces without vanishing state elimination.

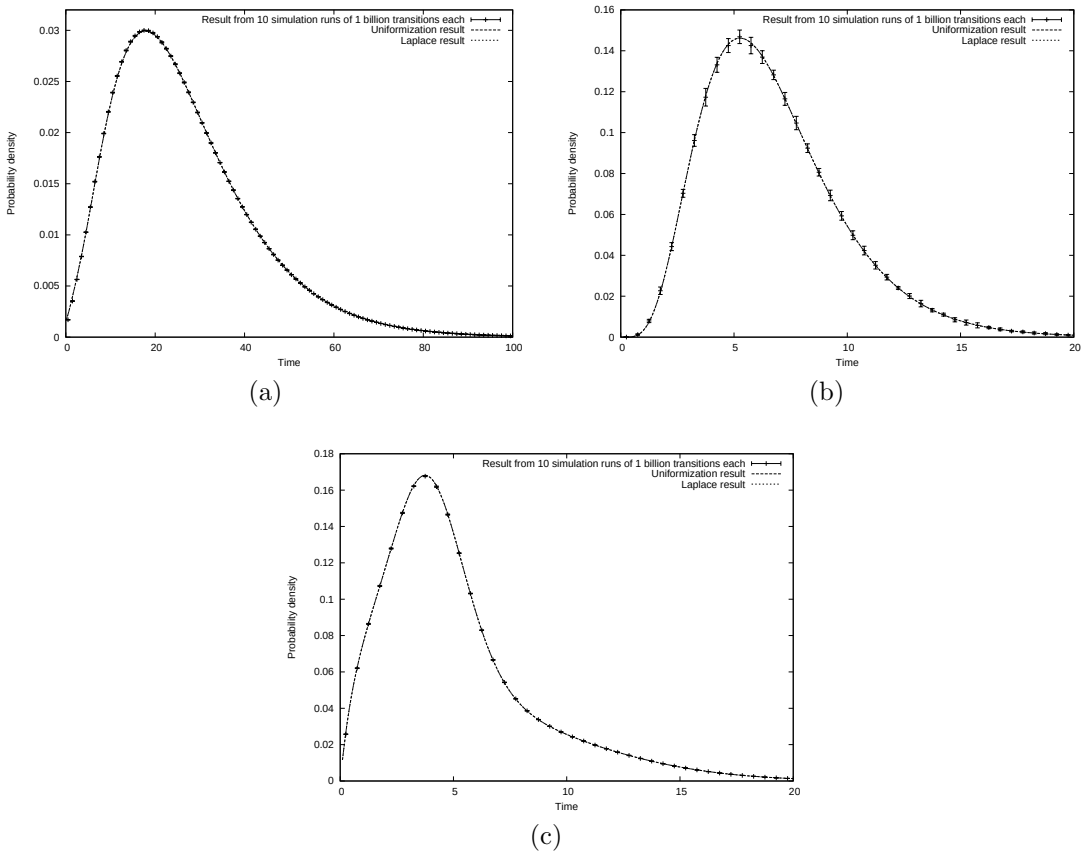

Fig. 5.1. Numerical and simulated (with 95% confidence intervals) passage time densities for the Courier (a), FMS model (b) and tree-like queueing models (c).

5.4.1. Models Studied. Results are presented for three models: a GSPN model of a flexible manufacturing system, a GSPN model of a communication protocol and a tree-like queueing network. These models are described in detail in Sections 4.2, 4.3 and 4.4 respectively. In order for uniformization to be used and compared fairly with the Laplace transform method, it was necessary in the case of the two GSPN models to generate the state spaces with the vanishing states eliminated. Therefore, the number of states given in the second column of Table 5.1 is the size of each model's underlying CTMC.

For the Courier protocol model, the passage of interest is from markings for which $M(p_{11}) > 0$ to those markings for which $M(p_{20}) > 0$, where $M(p)$ is the number of tokens in place p. This corresponds to the end-to-end response time from the initiation of a transport layer transmission to the arrival of the corresponding acknowledgement packet; as the sliding window size n is set to 1, there can be only one outstanding unacknowledged packet at any time. In the Flexible Manufacturing System (FMS) model we calculate the density of the time taken to produce a finished part of type $P12$ starting from any state in which there are 6 unprocessed parts of type $P1$ and 6 unprocessed parts of type $P2$. That is, the source markings are those where $M(P1) = M(P2) = 6$ and the target markings are those where $M(P12s) = 1$. Finally, for the tree-like queueing network, results are presented for a model with 15 customers and show the density of the cycle time of a customer from when it arrives at the back of the queue for server $q1$ to when it reaches that point again.

For uniformization and simulation, the results were produced using a single PC, but for the Laplace transform method a parallel solver illustrated in Figure 5.2 was used. This has a distributed master–slave architecture, where the master processor calculates in advance at which

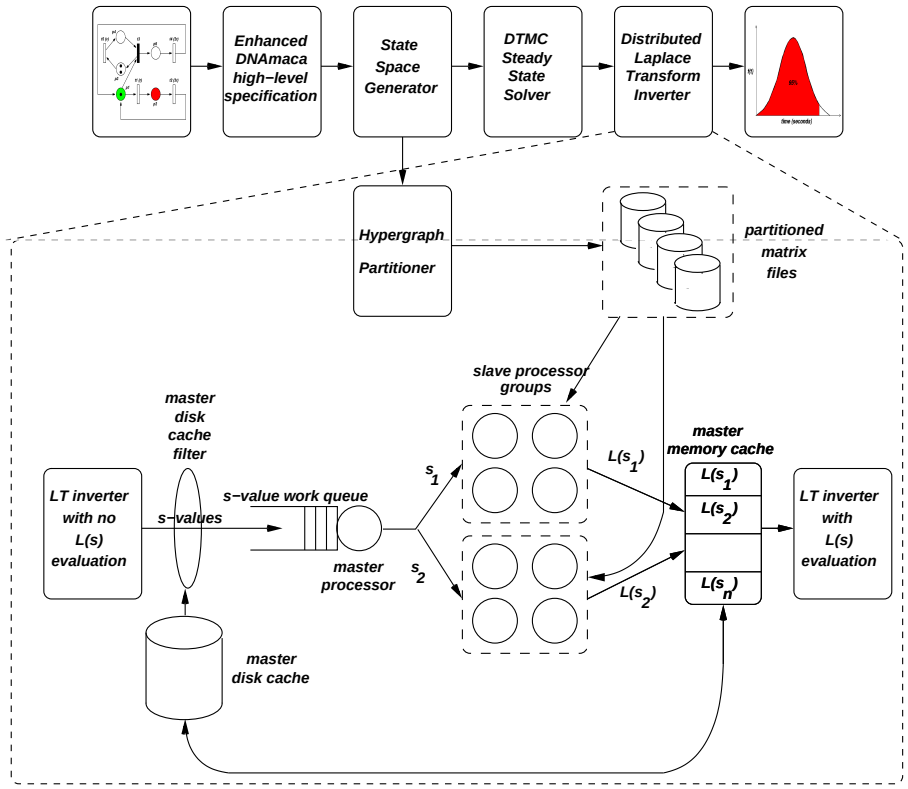

Fig. 5.2. SMP passage time density calculation pipeline [23].

values of s Equation (5.3) or Equation (5.7) will need to be solved in order to perform the numerical inversion. These values of s are placed in a queue to which slave processors make requests. They are allocated the next s value available and then construct and solve the set of linear equations for that value of s, returning the result to the master to be cached. When all the results have been returned, the master processor then uses the cached values to perform the inversion and returns the value of the passage time density or distribution at the required values of t.

The simulation results presented in the graphs are the combined results from 10 runs, each run consisting of 1 billion (10^9) transition firings. The timing information for simulation in Table 5.1 is the average time taken to perform one of these runs: as the runs are independent of each other they can be executed in parallel and so 10 runs on 10 machines should take no longer than 1 run on 1 machine.

5.4.2. Discussion. From Table 5.1 it can be seen that uniformization (running on a single processor) is much faster than the Laplace transform method (for all number of processors up to 32) except in the case of the smallest model considered (Courier with 11 700 states). Using the Laguerre method required the solution of 402 sets of equations of the form of Equation (5.4) for the tree-like queueing network and FMS models, and 804 sets for Courier. The reason that Courier required more equations to be solved was the use of the scaling technique referred to in Section 3.1.2. We also note that for the Courier model the solution took longer on 32 machines than it did on 16. This can be attributed to increased contention for the global work queue.

In contrast, the uniformization implementation needed only to perform a single set of sparse matrix–vector multiplications of the form shown in Equation (5.10). It must be noted, however, that the Laplace transform method is easier to extend to systems with generally-distributed state holding-time distributions (see Section 6 below) and preserves the ability to reason about source and target states which are vanishing. This ability is lost in the uniformization method as vanishing

states must be eliminated when generating the state space for the method to function.

Table 5.1 also shows that a single simulation run took much longer than either uniformization or the Laplace transform method for all three models and 10 runs only produced inexact results bounded by confidence intervals (although the intervals are fairly tight in two of the three cases). This run-time could, however, have been reduced by performing fewer transition firings but this may have resulted in wider confidence intervals. Simulation may not be suitable for passage time calculation in all models, particularly those which are very large but have very few initial states. In such cases, many more transition firings may have to be performed in order to achieve meaningful results as the number of observed passages would otherwise be too low. By way of example, the CTMC underlying the FMS model has 537 638 states, of which only 28 are source states (0.005% of the total states) and 136 584 are target states. The CTMC of the Courier model with 11 700 states has 3 150 source states (26.9% of the total states) and 900 target states. We observe that the confidence intervals on the Courier passage time density of Figure 5.1 are much tighter than those on the FMS density in Figure 5.1(a).

When the method of Section 5.2 is used for the analysis of GSPN models, the underlying state spaces are larger as vanishing states are not eliminated. Table 5.2 shows the sizes of the state spaces for the two GSPN models and the time taken to analyse them for the same passage time quantities as Table 5.1 using the Laplace transform method for GSPNs. Note the large increase in the size of the underlying process when vanishing states are not eliminated (it has more than doubled in the case of Courier and increased by a factor of 5 for the FMS model) and the consequent increase in time taken to compute the results. This illustrates that, although the Laplace transform method for GSPNs offers the opportunity to reason about vanishing source and target states, the modeller must be aware that it does so at increased computational cost.

We note that we have developed a parallel tool called HYDRA [24] which implements the uniformization method. This was not used here, however, as the size of the state spaces under consideration made it unnecessary. Its use could reduce the time taken to perform the uniformization calculations even further.

6. Passage Times in Semi-Markov Models. In this section, we present a central contribution of this chapter: an iterative algorithm for the calculation of passage time densities for very large semi-Markov processes (SMPs) [13]. Previous attempts to analyse SMPs for passage time measures have not been applicable to very large models due to the complexity of maintaining the Laplace transforms of state holding time distributions in closed form. In previous work [27, 33], the time complexity of the numerical calculation of passage time densities and quantiles for a semi-Markov system with N states is $O(N^4)$. Consequently, it has not been possible to analyse semi-Markov systems with more than a few thousand states.

This limitation is overcome here by the application of an efficient representation for the Laplace transforms of the state holding time density functions, which was developed with the demands of the numerical inversion algorithms described in Section 3.1 in mind. The resulting technique is amenable to a parallel implementation (thus allowing for the analysis of even larger semi-Markov models) and has a time complexity of $O(N^2 r)$ for r iterations (typically $r \ll N$).

We also present an iterative algorithm for computing transient state probabilities in SMPs which requires less computational effort than existing techniques. The algorithm builds on the iterative passage time algorithm and shows a similar time complexity. We present example passage time and transient results from models with up to 1.1 million states. The section concludes by considering the extraction of moments of passage times in semi-Markov systems.

6.1. Efficient Representation of General Distributions. The key to practical analysis of semi-Markov processes lies in the efficient representation of their general distributions. Without care the structural complexity of the SMP can be recreated within the representation of the distribution functions. This is especially true with the convolutions performed in the calculation of passage time densities. Many techniques have been used for representing arbitrary distributions – two of the most popular being *phase-type distributions* [43] 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. Attempts

to maintain a wholly symbolic representation are similarly hamstrung by space constraints. To get round these issues, we use a similar s-value lookahead technique as was used in Section 5.1 for Markovian analysis using Laplace functions.

As all the distribution manipulations in the algorithm take place in s-space, the distribution representation is linked to the Laplace inversion technique used. The two Laplace transform inversion algorithms which are applied in this chapter are described in Section 3.1. Both 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 $n = (k + m + 1)$, where k can vary between 15 and 50, depending on the accuracy of the inversion required. In the modified Laguerre case, $n = 400$ and is independent of m (cf. Section 3.1.2).

Whichever Laplace transform inversion technique is employed, it is important to note that calculating $s_i, 1 \leq i \leq n$ and storing all the state holding time distribution transform functions, sampled at these points, will be sufficient to provide a complete inversion. Key to this is the fact that convolution and weighted sum operations do not require any adjustment to the array of domain s-points required. In the case of a convolution, for instance, if $L_1(s)$ and $L_2(s)$ are stored in the form $\{(s_i, L_j(s_i)) : 1 \leq i \leq n\}$, for $j = 1, 2$, then the convolution, $L_1(s)L_2(s)$, can be stored using the same size array and using the same list of domain s-values, $\{(s_i, L_1(s_i)L_2(s_i)) : 1 \leq i \leq n\}$.

Storing the 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 requirement even after composition with other distributions. Finally, the function has sufficient information about a distribution to determine the required passage time, and no more.

6.2. The Laplace Transform Method for SMPs. The Laplace transform-based method described in Section 5 for the extraction of passage times from Markov models can be extended to the analysis of semi-Markov models. From Section 2.1.2, 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 $t \geq 0$, $N(t)$ denotes the number of transitions which have occurred by time t and $\{(\chi_n, T_n) : n \geq 0\}$ describe the state and time after the nth transition. The first passage time from a source state i at time t into a non-empty set of target states $\vec{j}$ is defined as:

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

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

$$P_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j}, N(u) > 0, Z(0) = i\} \tag{6.1}$$

This formulation of the random variable $P_{i\vec{j}}$ applies to an SMP with no immediate (that is, zero-time) transitions. If zero-time transitions are permitted in the model then the passage time can be stated as:

$$P_{i\vec{j}} = \inf\{u > 0 : N(u) \geq M_{i\vec{j}}\} \tag{6.2}$$

where $M_{i\vec{j}} = \min\{m > 0 : \chi_m \in \vec{j}, \chi_0 = i\}$ is the transition marking the terminating state of the passage.

$P_{i\vec{j}}$ has an associated probability density function $f_{i\vec{j}}(t)$. In a similar way to Section 5.1, the Laplace transform of $f_{i\vec{j}}(t)$, $L_{i\vec{j}}(s)$, can be computed by means of a first-step analysis. That is, we consider moving from the source state i into the set of its immediate successors $\vec{k}$ and must distinguish between those members of $\vec{k}$ which are target states and those which are not. This calculation can be achieved by solving a set of N linear equations of the form:

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} r^*_{ik}(s) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} r^*_{ik}(s) \qquad : \text{for } 1 \leq i \leq N \tag{6.3}$$

where $r^*_{ik}(s)$ is the Laplace-Stieltjes transform (LST) of $R(i,k,t)$ from Section 2.1.2 and is defined by:

$$r^*_{ik}(s) = \int_0^\infty e^{-st}\,\mathrm{d}R(i,k,t) \tag{6.4}$$

Equation (6.3) has a matrix–vector form $\mathbf{Ax} = \mathbf{b}$, where the elements of $\mathbf{A}$ are general functions of the complex variable s. For example, when $\vec{j} = \{1\}$, Equation (6.3) yields:

$$\begin{pmatrix} 1 & -r^*_{12}(s) & \cdots & -r^*_{1N}(s) \\ 0 & 1-r^*_{22}(s) & \cdots & -r^*_{2N}(s) \\ 0 & -r^*_{32}(s) & \cdots & -r^*_{3N}(s) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -r^*_{N2}(s) & \cdots & 1-r^*_{NN}(s) \end{pmatrix} \begin{pmatrix} L_{1\vec{j}}(s) \\ L_{2\vec{j}}(s) \\ L_{3\vec{j}}(s) \\ \vdots \\ L_{N\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} r^*_{11}(s) \\ r^*_{21}(s) \\ r^*_{31}(s) \\ \vdots \\ r^*_{N1}(s) \end{pmatrix} \tag{6.5}$$

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

$$L_{\vec{i}\vec{j}}(s) = \sum_{k\in\vec{i}} \alpha_k L_{k\vec{j}}(s) \tag{6.6}$$

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 $\boldsymbol{\alpha}$ is a normalised steady-state vector. That is, if $\boldsymbol{\pi}$ denotes the steady-state vector of the embedded discrete-time Markov chain (DTMC) with one-step transition probability matrix $\mathbf{P}$ with elements $p_{ij}, 1 \le i,j \le N$, then α_k is given by:

$$\alpha_k = \begin{cases} \pi_k / \sum_{j\in\vec{i}} \pi_j & : \text{if } k \in \vec{i} \\ 0 & : \text{otherwise} \end{cases} \tag{6.7}$$

The row vector with components α_k is denoted by $\boldsymbol{\alpha}$.

6.3. Iterative Passage Time Analysis. In this section, we present an iterative algorithm for generating passage time densities that creates successively better approximations to the SMP passage time quantity $P_{i\vec{j}}$ of Equation (6.1). We approximate $P_{i\vec{j}}$ as $P^{(r)}_{i\vec{j}}$, for a sufficiently large value of r, which is the time for r consecutive transitions to occur starting from state i and ending in any of the states in $\vec{j}$. We calculate $P^{(r)}_{i\vec{j}}$ by constructing and then inverting its Laplace transform $L^{(r)}_{i\vec{j}}(s)$.

This iterative method bears a loose resemblance to the uniformization technique described in Section 5.3 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, which we achieve using the representation technique described in Section 6.1.

6.3.1. Technical Overview. Modifying Equation (6.1), we define the rth transition first passage time to be:

$$P^{(r)}_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j}, 0 < N(u) \le r, Z(0) = i\} \tag{6.8}$$

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. If we have immediate transitions in our SMP model (as in Equation (6.2)) then the rth transition first passage time is:

$$P^{(r)}_{i\vec{j}} = \inf\{u > 0 : M_{i\vec{j}} \le N(u) \le r\}$$

$P^{(r)}_{i\vec{j}}$ is a random variable with associated Laplace transform $L^{(r)}_{i\vec{j}}(s)$. $L^{(r)}_{i\vec{j}}(s)$ is, in turn, the ith component of the vector:

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

representing the passage time for terminating in $\vec{j}$ for each possible start state. This vector may be computed as:

$$\mathbf{L}^{(r)}_{\vec{j}}(s) = \mathbf{U}\left(\mathbf{I} + \mathbf{U}' + \mathbf{U}'^2 + \cdots + \mathbf{U}'^{(r-1)}\right)\mathbf{e}_{\vec{j}} \tag{6.9}$$

where $\mathbf{U}$ is a matrix with elements $u_{pq} = r^*_{pq}(s)$ and $\mathbf{U}'$ is a modified version of $\mathbf{U}$ with elements $u'_{pq} = \delta_{p\notin\vec{j}}\, u_{pq}$, where states in $\vec{j}$ have been made absorbing. Here, $\delta_{p\notin\vec{j}} = 1$ if $p \notin \vec{j}$ and 0 otherwise. The initial multiplication with $\mathbf{U}$ in Equation (6.9) is included so as to generate cycle times for cases such as $L^{(r)}_{ii}(s)$ which would otherwise register as 0 if $\mathbf{U}'$ were used instead. The column vector $\mathbf{e}_{\vec{j}}$ has entries $e_{k\vec{j}} = \delta_{k\in\vec{j}}$, where $\delta_{k\in\vec{j}} = 1$ if k is a target state ($k \in \vec{j}$) and 0 otherwise.

From Equation (6.1) and Equation (6.8):

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

This can be generalised to multiple source states $\vec{i}$ using, for example, the normalised steady-state vector $\boldsymbol{\alpha}$ of Equation (6.7):

$$\begin{aligned} L^{(r)}_{\vec{i}\vec{j}}(s) &= \boldsymbol{\alpha}\mathbf{L}^{(r)}_{\vec{j}}(s) \\ &= \left(\boldsymbol{\alpha}\mathbf{U} + \boldsymbol{\alpha}\mathbf{U}\mathbf{U}' + \boldsymbol{\alpha}\mathbf{U}\mathbf{U}'^2 + \cdots + \boldsymbol{\alpha}\mathbf{U}\mathbf{U}'^{(r-1)}\right)\mathbf{e}_{\vec{j}} \\ &= \sum_{k=0}^{r-1} \boldsymbol{\alpha}\mathbf{U}\mathbf{U}'^k\,\mathbf{e}_{\vec{j}} \end{aligned} \tag{6.10}$$

The sum of Equation (6.10) can be computed efficiently using sparse matrix–vector multiplications with a vector accumulator, $\boldsymbol{\mu}_r = \sum_{k=0}^{r} \boldsymbol{\alpha}\mathbf{U}'^k$. At each step, the accumulator (initialised as $\boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{U}$) is updated as $\boldsymbol{\mu}_{r+1} = \boldsymbol{\alpha}\mathbf{U} + \boldsymbol{\mu}_r\mathbf{U}'$. 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 $r \ll N$ for large N (see Section 6.3.3 below).

6.3.2. Example Passage Time Results. In this section, we display passage time densities produced by our iterative passage time algorithm and validate these results by simulation. Readers are referred to Section 4.1 for full details of the Web-server model in which these passage times are measured.

Figure 6.1 (left) shows the density of the time taken to process 45 reads and 22 writes in system 1 of the Web-server model (107 289 states). This corresponds to the movement of 45 tokens from p_1 to p_8 and 22 tokens from p_2 to p_9. The graph shows results computed by both the iterative technique and the combined results from 10 simulations of 1 billion transition firings each. The close agreement provides mutual validation of the analytical method, with its numerical approximation, and the simulation.

Figure 6.1 (right) shows the cumulative distribution for the same passage as Figure 6.1 (left). An example response time quantile for this measure would be:

$$\mathbb{P}(\text{all reads and all writes are processed in under 470 seconds}) = 0.954$$

Figure 6.2 (left) shows the density of the time taken to perform 100 reads and 50 page updates in the Web-server model 6 (15 445 919 states). Calculation of the 35 t-points plotted required 2 days, 17 hours and 30 minutes using 64 slave processors (in 8 groups of 8). Our algorithm evaluated

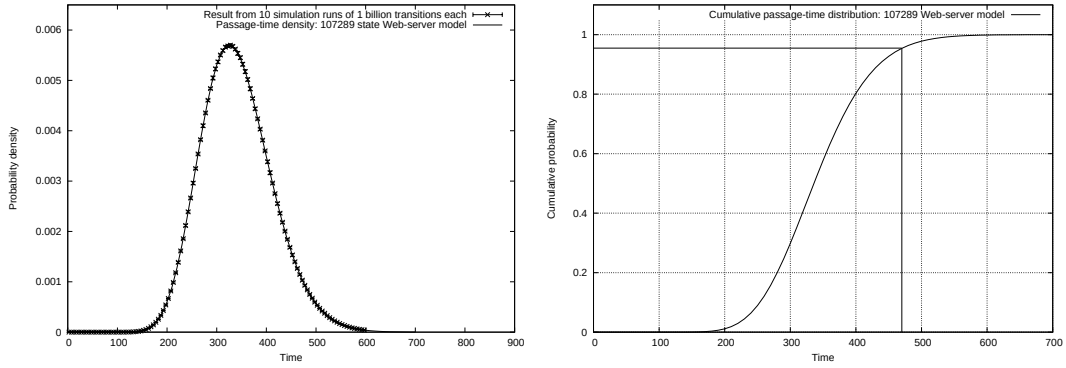

Fig. 6.1. Numerical and simulated (with 95% confidence intervals) density (left) and cumulative distribution function and quantile (right) for the time taken to process 45 reads and 22 writes in the Web-server model system 1 (107 289 states).

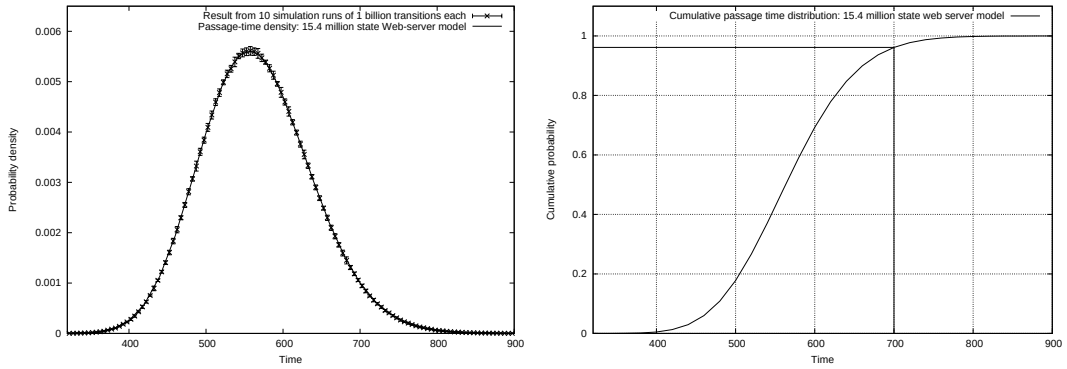

Fig. 6.2. Numerical and simulated (with 95% confidence intervals) density for the time taken to process 100 reads and 50 page updates in the Web-server model system 6 (15.4 million states).

$L_{\vec{ij}}(s)$ at 1 155 s-points, each of which involved manipulating sparse matrices of rank 15 445 919. Again, the numerical result is validated against the combined results from 10 simulations, each of which consisted of 1 billion transition firings. We observe excellent agreement. Figure 6.2 (right) shows the corresponding cumulative distribution function with a reliability quantile superimposed, in this case showing:

$$\mathbb{P}(\text{system 6 can process 100 reads and 50 page updates in less than 700 seconds}) = 0.9613$$

6.3.3. Practical Convergence of the Iterative Passage Time Algorithm. In practice, convergence of the sum $L^{(r)}_{\vec{ij}}(s) = \sum_{k=0}^{r-1} \boldsymbol{\alpha}\mathbf{U}\mathbf{U}'^{k}$ can be said to have occurred if, for a particular r and s-point:

$$|\mathrm{Re}(L^{(r+1)}_{\vec{ij}}(s) - L^{(r)}_{\vec{ij}}(s))| < \varepsilon \quad \text{and} \quad |\mathrm{Im}(L^{(r+1)}_{\vec{ij}}(s) - L^{(r)}_{\vec{ij}}(s))| < \varepsilon \tag{6.11}$$

where ε is chosen to be a suitably small value, say $\varepsilon = 10^{-16}$. Empirical observations on the convergence behaviour of this technique (i.e. the order of r) are presented below.

Figure 6.3(a) shows the average number of iterations the algorithm takes to converge per s-point for the Web-server model (see Appendix 4.1 for full details) for two different values of ε (10^{-8} and 10^{-16}). It is noted that the number of iterations required for convergence as the model size grows is sub-linear; that is, as the model size doubles the number of iterations less than doubles. This suggests the algorithm has good scalability properties.

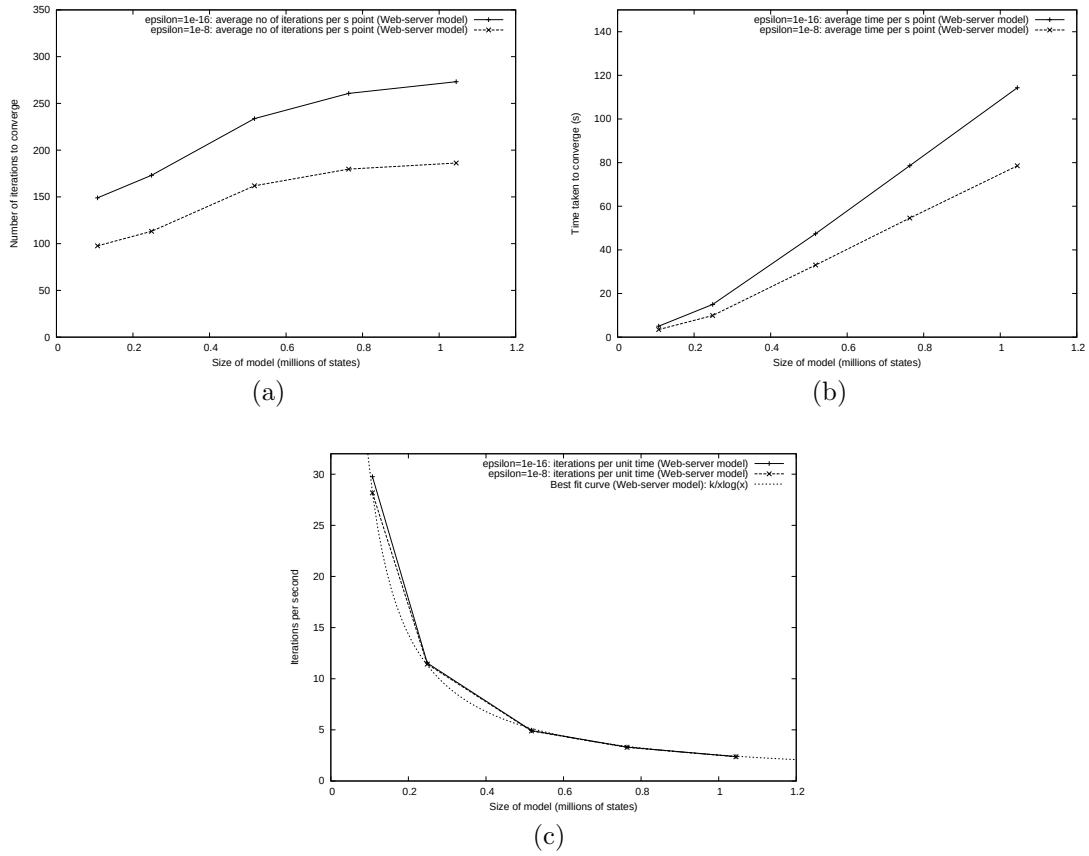

Fig. 6.3. (a) Average number of iterations to converge per s point for two different values of ε over a range of model sizes for the iterative passage time algorithm. (b) Average time to convergence per s point for two different values of ε over a range of model sizes for the iterative passage time algorithm. (c) Average number of iterations per unit time over a range of model sizes for the iterative passage time algorithm.

Figure 6.3(b) shows the average amount of time to convergence per s-point, while Figure 6.3(c) shows how the number of iterations per unit time decreases as model size increases. The curves are almost identical for both values of ε, suggesting that the time spent per iteration remains constant, irrespective of the number of iterations performed. The rate of computation (iterations per unit time) is $O\big(1/(N\log(N))\big)$ for system size N. This gives a time per iteration of $O\big(N\log(N)\big)$, suggesting an overall practical complexity of better than $O\big(N^2\log(N)\big)$ (given the better than $O(N)$ result for the number of iterations required).

6.4. Iterative Transient Analysis. This section presents a numerical advance for SMPs first published in [12]. It is reproduced here as it is similar to the iterative passage time method in construction. Another important modelling result is the transient state distribution $\pi_{ij}(t)$ of a stochastic process:

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

From Pyke's seminal paper on SMPs [46], we have the following relationship between passage time densities and transient state distributions, in Laplace form:

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

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

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

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 Equation (6.5). 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 per s-value.

From Pyke's formula for the transient state distribution between two states [46, Eq. (3.2)], we can derive:

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

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. $R(i,k,t-\tau)$ is the probability that a transition from state i to an adjacent state k occurs 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:

$$\pi^*_{i\vec{j}}(s) = \delta_{i\in\vec{j}}\overline{F}^*_i(s) + \sum_{k=1}^{N} r^*_{ik}(s)\pi^*_{k\vec{j}}(s) \tag{6.12}$$

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:

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

Equation (6.12) can be written in matrix–vector form; for example, when $\vec{j} = \{1,3\}$, we have:

$$\begin{pmatrix} 1-r^*_{11}(s) & -r^*_{12}(s) & \cdots & -r^*_{1N}(s) \\ -r^*_{21}(s) & 1-r^*_{22}(s) & \cdots & -r^*_{2N}(s) \\ -r^*_{31}(s) & -r^*_{32}(s) & \cdots & -r^*_{3N}(s) \\ \vdots & \vdots & \ddots & \vdots \\ -r^*_{N2}(s) & -r^*_{N2}(s) & \cdots & 1-r^*_{NN}(s) \end{pmatrix} \begin{pmatrix} \pi^*_{1\vec{j}}(s) \\ \pi^*_{2\vec{j}}(s) \\ \pi^*_{3\vec{j}}(s) \\ \vdots \\ \pi^*_{N\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} \overline{F}^*_1(s) \\ 0 \\ \overline{F}^*_3(s) \\ \vdots \\ 0 \end{pmatrix} \tag{6.13}$$

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

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

6.4.1. Technical Overview. Our iterative transient state distribution generation technique builds on the passage time computation technique of Section 6.3. 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 and then inverting $\pi^{(r)}_{i\vec{j}}(s)$, which is the rth iterative approximation to the Laplace transform of the transient state distribution function, for a sufficiently large value of r.

We note that Equation (6.13) can be written as:

$$(\mathbf{I} - \mathbf{U})\,\boldsymbol{\pi}_{\vec{j}}(s) = \mathbf{v} \tag{6.14}$$

where matrix $\mathbf{U}$ has elements $u_{pq} = r^*_{pq}(s)$ and column vector $\mathbf{v}$ has elements $v_i = \delta_{i\in\vec{j}}\overline{F}^*_i(s)$. The vector $\boldsymbol{\pi}_{\vec{j}}(s)$ has elements $\pi_{i\vec{j}}(s)$:

$$\boldsymbol{\pi}_{\vec{j}}(s) = \Big(\pi_{1\vec{j}}(s), \pi_{2\vec{j}}(s), \ldots, \pi_{N\vec{j}}(s)\Big)$$

Equation (6.14) can be rewritten (see [14] for the proof that $(\mathbf{I} - \mathbf{U})$ is invertible) as:

$$\begin{aligned}\boldsymbol{\pi}_{\vec{j}}(s) &= (\mathbf{I} - \mathbf{U})^{-1}\,\mathbf{v} \\ &= \Big(\mathbf{I} + \mathbf{U} + \mathbf{U}^2 + \mathbf{U}^3 + \cdots\Big)\,\mathbf{v}\end{aligned}$$

This infinite summation may be approximated as:

$$\boldsymbol{\pi}_{\vec{j}}(s) \simeq \boldsymbol{\pi}^{(r)}_{\vec{j}}(s) = \Big(\mathbf{I} + \mathbf{U} + \mathbf{U}^2 + \cdots + \mathbf{U}^r\Big)\,\mathbf{v}$$

for a suitable value of r such that the approximation is good. See Section 6.4.3 below for observations regarding typical values of r.

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 $\mathbf{U}$. This reflects the fact that the transient state distribution accumulates probability from all passages through the system and not just the first one.

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

$$\pi^{(r)}_{\vec{i}\vec{j}}(s) = \boldsymbol{\alpha}\Big(\mathbf{I} + \mathbf{U} + \mathbf{U}^2 + \cdots + \mathbf{U}^r\Big)\,\mathbf{v}$$

Having calculated $\pi^{(r)}_{\vec{i}\vec{j}}(s)$ in this manner, the same numerical inversion techniques which are used in passage time analysis can be employed to compute $\pi_{\vec{i}\vec{j}}(t)$.

6.4.2. Example Transient Results. We demonstrate our iterative transient technique on the small two-state example with two sates, 0 and 1. The distribution of transitions from 0 to 1 is X, the one for transitions from 1 to 0 is Y. Figure 6.4(a) 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 distributions of the transitions are $X \sim \exp(2)$ and $Y \sim \det(2)$. 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 up to 8 iterations of the algorithm which exhibit increasing accuracy in approximating the transient curve.

Figure 6.4(b) shows the transient state distribution $\pi_{00}(t)$ for the two state system with $X \sim \det(3)$ and $Y \sim \exp(0.5)$. 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 numerical instabilities (Gibb's Phenomena) in the Laplace inversion algorithm [3]. Also shown on the graph are 4 iterations of the algorithm which exhibit increasing accuracy in approximating the transient curve, as before.

We also use the two state system to highlight when numerical Laplace transform inversion does not perform well and how such problems can be avoided. Figure 6.4(c) shows the transient probability of being in state 0 having started in state 0 when both X and Y are $\det(2)$ transitions. We would expect to see the probability equalling 1 for $0 < t < 2$, $4 < t < 6$ and so forth, and 0 at $2 < t < 4$, $6 < t < 8$ and so on, but the numerically computed result becomes increasingly unstable as t increases. This is because discontinuities in $f(t)$ and its derivatives result in instabilities in the numerical inversion. Even the Euler algorithm (which was used to produce these results) performs

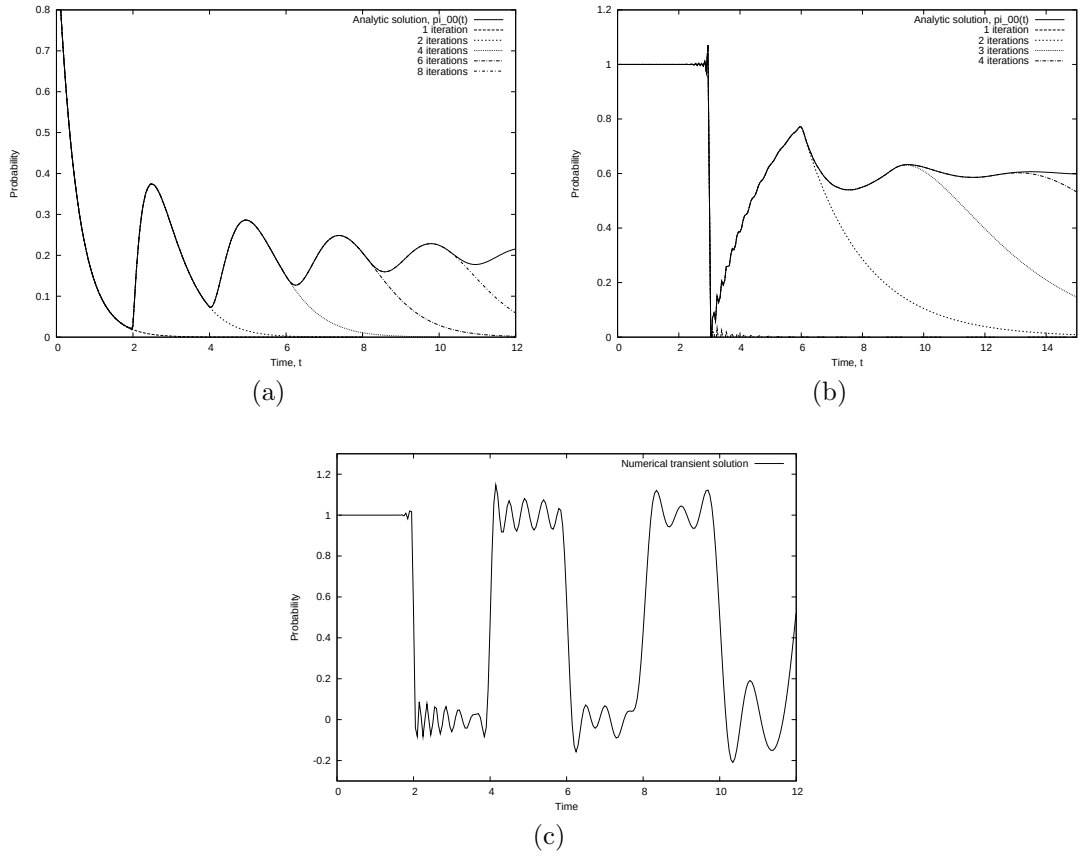

Fig. 6.4. (a) Example iterations towards a transient state distribution in a system with successive exponential and deterministic transitions. (b) Example iterations towards a transient state distribution in a system with successive deterministic and exponential transitions. (c) Where numerical inversion performs badly: transient state distribution in a system with two deterministic transitions.

badly when inverting entirely deterministic probability distributions. This example, with two such transitions and no source of randomness, is the worst case we could expect to deal with.

The presence of a small amount of randomness is, however, enough to remove this instability. We modify the two state system by adding a new state with a single exp(0.5) transition into state 0 and calculate the transient probability of being in state 0 having started in the newly added predecessor state. There is no transition from state 0 back to the new state, so the exp(0.5) transition fires only once. The resulting transient state distribution is shown in Figure 6.4(c). Note that the numerical instability has disappeared. This demonstrates that only a small amount of randomness in the model can be sufficient for numerical inversion to be applied successfully.

6.4.3. Practical Convergence of the Iterative Transient Algorithm. As in the iterative passage time algorithm, convergence of the sum $\pi_{\vec{ij}}^{(r)}(s)$ is said to have occurred if, for a particular r and s-point:

$$|\mathrm{Re}(\pi_{\vec{ij}}^{(r+1)}(s) - \pi_{\vec{ij}}^{(r)}(s))| < \varepsilon \quad \text{and} \quad |\mathrm{Im}(\pi_{\vec{ij}}^{(r+1)}(s) - \pi_{\vec{ij}}^{(r)}(s))| < \varepsilon \tag{6.15}$$

where ε is chosen to be a suitably small value such as 10^{-16}.

7. Conclusion. This chapter has explored the numerical computation of passage time densities, quantiles and transient distributions in large Markov and semi-Markov models. Prior work in this area has focused mainly on the analysis of Markov systems using the two techniques described

in Section 5: the Laplace transform method and uniformization. We have expanded on this by presenting techniques which enable the computation of passage time densities and quantiles in Markov and semi-Markov models with state spaces of 10^7 states and above.

Key to this is the iterative passage time analysis algorithm described in Section 6. This calculates the Laplace transform of the passage time quantity by convolving the Laplace transforms of the state holding-times across all possible paths between source and target states. This Laplace transform is then inverted using numerical inversion techniques to yield the value of the density or quantile function at a range of user-specified time points. This iterative algorithm has also been extended to permit the efficient calculation of transient state distributions.

In order to describe large semi-Markov models succinctly, we have devised a high-level modelling formalism called Semi-Markov Stochastic Petri Nets (SM-SPNs) [11]. This is an extension of stochastic Petri nets which includes transitions with generally-distributed firing delays. In the event that two or more general transitions are concurrently enabled, the selection of the next to fire is done by probabilistic choice and the delay is then sampled from that transition's firing distribution. This means that the underlying stochastic process is isomorphic to a semi-Markov chain. While SM-SPNs do not attempt to model true Generalised Semi-Markov Process-style concurrency, they do support Markovian concurrency provided that generally-distributed (non-exponential) transitions are exclusively enabled.

Using these techniques, passage time density and quantile results have been calculated for Markov and semi-Markov models with up to 15.4 million states.

REFERENCES

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

[2] ———, *An introduction to numerical transform inversion and its application to probability models*, in Computational Probability, W Grassman, ed., Kluwer, Boston, 2000, pp. 257–323.

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

[4] ———, *Numerical inversion of Laplace transforms of probability distributions*, ORSA Journal on Computing, 7 (1995), pp. 36–43.

[5] M Ajmone-Marsan, G Conte, and G Balbo, *A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems*, ACM Transactions on Computer Systems, 2 (1984), pp. 93–122.

[6] Australian Capital Territory Commissioner for the Environment, *Indicator: Emergency services*, February 2004.

[7] F Bause and P S Kritzinger, *Stochastic Petri Nets – An Introduction to the Theory*, Verlag Vieweg, Wiesbaden, Germany, 1995.

[8] M Benzi and M Tuma, *A parallel solver for large-scale Markov chains*, Applied Numerical Mathematics, 41 (2002), pp. 135–153.

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

[10] 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 Proceedings of the International Workshop on Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems (PMEO-PDS'03), Nice, April 26th 2003.

[11] ———, *Performance queries on semi-Markov stochastic Petri nets with an extended Continuous Stochastic Logic*, in Proceedings of 10th International Workshop on Petri Nets and Performance Models (PNPM'03), Urbana-Champaign IL, USA, September 2nd–5th 2003, pp. 62–71.

[12] ———, *Distributed computation of transient state distributions and passage time quantiles in large semi-Markov models*, Future Generation Computer Systems, (2005). (to appear).

[13] 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*, Linear Algebra and Its Applications, 386 (2004), pp. 311–334.

[14] J T Bradley and H J Wilson, *Iterative convergence of passage-time densities in semi-Markov performance models*, Performance Evaluation, 60 (2004), pp. 237–254.

[15] P Buchholz, M Fischer, and P Kemper, *Distributed steady state analysis using Kronecker algebra*, in Proceedings of the 3rd International Conference on the Numerical Solution of Markov Chains (NSMC'99), Zaragoza, Spain, September 1999, pp. 76–95.

[16] G Ciardo and A S Miner, *A data structure for the efficient Kronecker solution of GSPNs*, in Proceedings of the 8th International Conference on Petri Nets and Performance Models (PNPM'99), Zaragoza, Spain,

September 1999, IEEE Computer Society Press, pp. 22–31.
[17] G Ciardo and K S Trivedi, *A decomposition approach for stochastic reward net models*, Performance Evaluation, 18 (1993), pp. 37–59.
[18] Commission for Health Improvement, *Final key targets and performance indicators for primary care trusts (PCTs)*, December 2003.
[19] County of Oxford Board of Health, *2001 annual report*, October 2002.
[20] Cross-Industry Working Team, *Customer view of Internet service performance: Measurement methodology and metrics*, October 1998.
[21] D D Deavours and W H Sanders, *An efficient disk-based tool for solving large Markov models*, Performance Evaluation, 33 (1998), pp. 67–84.
[22] ———, *"On-the-fly" solution techniques for stochastic Petri nets and extensions*, IEEE Transactions on Software Engineering, 24 (1998), pp. 889–902.
[23] N J Dingle, P G Harrison, and W J Knottenbelt, *Response time densities in Generalised Stochastic Petri Net models*, in Proceedings of the 3rd International Workshop on Software and Performance (WOSP'02), Rome, July 24th–26th 2002, pp. 46–54.
[24] ———, *HYDRA: HYpergraph-based Distributed Response-time Analyser*, in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03), Las Vegas NV, USA, June 23rd–26th 2003, pp. 215–219.
[25] DLT Solutions Inc., *Capacity planning for e-commerce systems with Benchmark Factory™*, February 2004.
[26] H Dubner and J Abate, *Numerical inversion of Laplace transforms by relating them to the finite Fourier cosine transform*, Journal of the ACM, 15 (1968), pp. 115–123.
[27] R German, D Logothetis, and K S Trivedi, *Transient analysis of Markov regenerative stochastic Petri nets: A comparison of approaches*, in Proceedings of the 6th International Workshop on Petri Nets and Performance Models (PNPM'95), Durham, North Carolina, 1995, pp. 103–112.
[28] W K Grassman, *Means and variances of time averages in Markovian environments*, European Journal of Operational Research, 31 (1987), pp. 132–139.
[29] P G Harrison, *Laplace transform inversion and passage-time distributions in Markov processes*, Journal of Applied Probability, 27 (1990), pp. 74–87.
[30] P G Harrison and W J Knottenbelt, *Passage time distributions in large Markov chains*, in Proceedings of ACM SIGMETRICS 2002, Marina Del Rey, California, June 2002, pp. 77–85.
[31] H Hermanns, J Meyer-Kayser, and M Siegle, *Multi-terminal binary decision diagrams to represent and analyse continuous time Markov chains*, in Proceedings of the 3rd International Conference on the Numerical Solution of Markov Chains (NSMC'99), Zaragoza, Spain, September 1999, pp. 188–207.
[32] J Hillston, *A Compositional Approach to Performance Modelling*, PhD thesis, University of Edinburgh, 1994.
[33] G G Infante López, H Hermanns, and J-P Katoen, *Beyond memoryless distributions: Model checking semi-Markov chains*, in Lecture Notes in Computer Science 2165: Proceedings of Process Algebra and Probabilistic Methods (PAPM'01), Aachen, September 2001, Springer-Verlag, pp. 57–70.
[34] W J Knottenbelt, *Parallel Performance Analysis of Large Markov Models*, PhD thesis, Imperial College, London, United Kingdom, February 2000.
[35] W J Knottenbelt and P G Harrison, *Distributed disk-based solution techniques for large Markov models*, in Proceedings of the 3rd International Conference on the Numerical Solution of Markov Chains (NSMC'99), Zaragoza, Spain, September 1999, pp. 58–75.
[36] M Kwiatkowska and R Mehmood, *Out-of-core solutions of large linear systems of equations arising from stochastic modelling*, in Proceedings of Process Algebra and Performance Modelling (PAPM'02), Copenhagen, July 25th–26th 2002, pp. 135–151.
[37] M Kwiatkowska, G Norman, and D Parker, *PRISM: Probabilistic symbolic model checker*, in Lecture Notes in Computer Science 2324: Proceedings of the 12th International Conference on Modelling, Techniques and Tools (TOOLS'02), London, April 14th–17th 2002, Springer Verlag, pp. 200–204.
[38] London Ambulance Service, *Category A response times*, February 2004.
[39] B Melamed and M Yadin, *Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes*, Operations Research, 32 (1984), pp. 926–944.
[40] A S Miner, *Computing response time distributions using stochastic Petri nets and matrix diagrams*, in Proceedings of the 10th International Workshop on Petri Nets and Performance Models (PNPM'03), Urbana-Champaign, IL, September 2nd–5th 2003, pp. 10–19.
[41] Municipal Corporation of the County of Renfrew, *Health committee minutes*, May 2003.
[42] 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., Oxford University Press, 1992, pp. 262–284.
[43] M F Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Johns Hopkins University Press, Baltimore, MD, 1981.
[44] C H Ng, *Queueing Modelling Fundamentals*, John Wiley and Sons, 1996.
[45] A Puliafito, M Scarpa, and K S Trivedi, *Petri nets with k simultaneously enabled generally distributed timed transitions*, Performance Evaluation, 32 (1998), pp. 1–34.
[46] R Pyke, *Markov renewal processes with finitely many states*, Annals of Mathematical Statistics, 32 (1961), pp. 1243–1259.
[47] A Reibman and K S Trivedi, *Numerical transient analysis of Markov models*, Computers and Operations

Research, 15 (1988), pp. 19–36.
[48] San Francisco EMS Section Department of Public Health, *San Francisco EMS system activity summary*, December 1999.
[49] R M Simon, M T Stroot, and G H Weiss, *Numerical inversion of Laplace transforms with application to percentage labeled mitoses experiments*, Computers and Biomedical Research, 5 (1972), pp. 596–607.
[50] Township of Rideau Lakes, *Leeds Grenville emergency medical services frequently asked questions*, February 2004.
[51] Transaction Processing Performance Council, *TPC benchmark C: Standard specification revision 5.2*, December 2003.
[52] W T Weeks, *Numerical inversion of Laplace transforms using Laguerre functions*, Journal of the ACM, 13 (1966), pp. 419–429.
[53] C M Woodside and Y Li, *Performance Petri net analysis of communication protocol software by delay-equivalent aggregation*, in Proceedings of the 4th International Workshop on Petri nets and Performance Models (PNPM'91), Melbourne, Australia, 2–5 December 1991, IEEE Computer Society Press, pp. 64–73.