CHAPTER 1

Quantiles of Sojourn Times

Peter G. Harrison and William J. Knottenbelt
Department of Computing, Imperial College London
South Kensington Campus, London SW7 2AZ, UK
E-mail: {pgh,wjk}@doc.ic.ac.uk

Fast response times and the satisfaction of response time quantile targets are important performance criteria for almost all transaction processing, computer-communication and other operational systems. However, response time quantiles tend to be difficult to obtain in stochastic models, even when the mean value of the response time has a relatively simple mathematical expression. Expressions have been obtained for the Laplace transform of the probability density function of sojourn times in many queueing models, including some complex single queues and networks of simple queues. These can sometimes be inverted analytically, giving an explicit expression for the density as a function of time, but more often numerical inversion is necessary. More generally, interesting sojourn times can be expressed in terms of passage times between states in continuous time Markov and semi-Markov chains. Quantiles for these can be computed in principle but can require extensive computational resources, both in terms of processing time and memory. Consequently, until recently, only trivial problems could be solved by this direct method. With recent technological advances, including the widespread use of clusters of workstations and limited availability of parallel supercomputers, very large Markov and semi-Markov chains can be solved directly for passage time densities, allowing many realistic systems to be investigated. This paper reviews the various approaches taken to compute sojourn time quantiles in systems ranging from simple queues to arbitrary semi-Markov chains, by the authors and others, over the past twenty years and more.

Keywords: Sojourn time; Response time quantile; response time density; Markov models; semi Markov models; queueing models; Laplace transform inversion

1. Introduction

The probability distribution of many response, or sojourn, times constitutes a vital quality of service (QoS) metric in many operational systems such as computer networks, logistical systems and emergency services. For example in the United Kingdom, ambulances must arrive at the scene of a life-threatening emergency within 8 minutes at least 75% of the time. For on-line transaction processing (OLTP) and other real-time systems, quantiles are often specified in Service Level Agreement contracts and industry standard benchmarks such as TPC-C, which specifies the 90^{th} percentile of response time [41].

The mean values of such time delays provide a good overall description of performance and are readily obtained by conventional techniques, but means alone are often insufficient. For example, we may wish to predict the variability of response time in a multi-access system or various reliability measures, such as the probability that a message transmission time exceeds a given value. The importance of obtaining quantiles of distributions – i.e. time intervals that are not exceeded with a specified probability – has therefore acquired increased recognition.

This paper is a review of various techniques used to compute quantiles numerically over the past quarter of a century and more. It is a mainly personal viewpoint and it should be remembered that there have been many excellent (and often mathematically more sophisticated) contributions by other researchers in the field, for example Onno Boxma and Hans Daduna [4].

Queueing network models which compute queue length distributions in a steady state network are well established; from the mean queue lengths, mean passage-time along a given path can be determined directly through Little's result [34]. Mathematically, the simplest type of network to analyse is open, acyclic and Markovian, i.e. has external arrivals from independent Poisson processes and exponentially distributed service times at the queueing nodes. The arrival process at every server is then independent and Poisson. Unfortunately, even these assumptions are too weak to allow the distribution of the sojourn time along an arbitrary path to be obtained in a simple form. For paths on which a task cannot be overtaken, we can consider sojourn time as the sum of waiting times at independent single-server (M/M/1) queues and obtain a simple solution. If any of these assumptions is violated (e.g. for any closed network of servers, independence is lost) the above approach fails. However, a more complex result can be derived for overtake-free paths in Markovian closed networks. Some analytical solutions

have been found for sojourn-time distributions in small networks of more general structure, but these are complex and often numerically intractable, see for example [36,26]. To derive sojourn-time distributions in more general networks usually requires either direct analysis of the underlying Markov chain, where numerically feasible, or else approximate methods.

Rather than the distributions themselves, it is often easier to work with their Laplace transforms, especially when a sojourn time is the sum of an independent set of smaller time delays. To obtain quantiles, of course, it is necessary to be able to invert the Laplace transform of the passage-time probability density function so as to recover the distribution itself. In general, inversion is by numerical methods which may be difficult to implement accurately over the whole time domain of interest – especially at the important high quantiles, i.e. in the tail of a distribution, although this is becoming less of a problem as increasingly sophisticated inverters become available and as computing technology advances. Furthermore, analytical inversion is possible in many of the solvable networks referred to above, including some complex, Markov modulated G-queues[a] with batches of customers and open and closed, overtake-free, Markovian networks. Where an analytical solution based on the inherent structure of the system under consideration is not possible, analysis of passage times between states in the underlying Markov chain solves the problem exactly when the state space is not excessively large – representing significant models of over 10 million states with the computing power available today.

In the next section we consider the sojourn-time distribution for a single-server queue, showing the remarkable effect of different queueing disciplines, which typically do not influence resource-oriented quantities like the queue length at equilibrium. Surprisingly, a Markov modulated queue with batch processing and negative customers [27] does have a tractable solution for the sojourn-time distribution in the time domain, provided customers are removed from the *front* of the queue by negative arrivals. This queue is considered in Section 3. Next, we look at sojourn-time distributions on paths in open, overtake-free or tree-like networks of Markovian queues in Section 4.1. The Laplace transform of the sojourn-time distribution on overtake-free paths in closed Markovian networks, together with its analytical inversion, is considered in Section 4.2. Section 5 reviews recent results on passage times in Markov chains, in particular, techniques developed for the parallel computation of quantiles. These include the application of the uniformiza-

[a] 'Gelenbe queues' which have negative customers [14,15]

tion technique in very large unstructured Markov models, using hypergraph partitioning to minimise interprocessor communication while maintaining a good load balance. We demonstrate this approach by calculating passage time densities and quantiles in a 10.8 million state Markov chain derived from a closed tree-like queueing network, for which we also have an analytical solution. Section 6 considers a technique for extracting passage times for large unstructured semi-Markov models using numerical Laplace transform inversion. Finally, in Section 7, we conclude, sketching out some current research in progress.

2. Time delays in the single server queue

In this section we first consider the waiting (response) and queueing times of a customer in an M/G/1 queue with FCFS discipline and then the *busy period* of the server, i.e. the interval between successive idle periods. Busy time analysis – a special case of delay-cycle analysis [40] – leads to the waiting time distribution of an M/G/1 queue with LCFS queueing discipline. The processor sharing (PS) queueing discipline is also considered, but only for M/M/1 queues where we can use the memoryless property at all times to exploit an analysis of infinitesimal intervals. In fact, we will see that this method can also be used for other queueing disciplines – even in the MM CPP/GE/c G-queue, with batches of customers and Markov modulation.

2.1. *Waiting time distribution in the M/G/1 queue*

The waiting time distribution for the FCFS discipline is traditionally obtained from the following observation. For $n \geq 1$, the queue, of length X_n, existing on the departure of the nth customer, C_n, comprises precisely the customers that arrived during that customer's waiting time. The distribution of the queue length at a departure instant, which is known to be the same as that at a random instant, is therefore equal to the distribution of the number of arrivals in a waiting time.

Hence, if, at equilibrium, we denote the waiting time random variable by W and the queue length random variable by X, then the generating function of the queue length may be expressed as

$$\Pi(z) = E[E[z^X \mid W]] = E[e^{-\lambda W(1-z)}] = W^*(\lambda(1-z))$$

where $W^*(s)$ is the Laplace-Stieltjes transform of $W(t)$, the probability distribution function of the waiting time (we use the same naming convention for all random variables). This is because X, conditional on W, has

a Poisson distribution with parameter λW. Writing $s = \lambda(1 - z)$ so that $z = (\lambda - s)/\lambda$, we now have

$$W^*(s) = \Pi((\lambda - s)/\lambda) = \frac{(1-\rho)sB^*(s)}{s - \lambda[1 - B^*(s)]}$$

using the Pollacek-Khintchine formula for Π, where B is the service time random variable and the load $\rho = -\lambda B^{*\prime}(0)$. We can now easily demonstrate Little's result for the M/G/1 queue since $-W^{*\prime}(0) = -\lambda^{-1}\Pi'(1)$. Notice too that we get the required exponential distribution in the case of an M/M/1 queue where Π is the generating function of the geometric random variable with parameter ρ.

2.2. *Busy periods*

To investigate the busy period, we first observe that its distribution is the same for all queueing disciplines that are work-conserving and for which the server is never idle when the queue is non-empty. Suppose that, at equilibrium, whilst an initial customer C_1 is being served, customers $C_2, \ldots, C_{Z+1}$ arrive, where the random variable Z, conditional on service time B for C_1, is Poisson with mean λB. Without loss of generality, we assume a LCFS queueing discipline with no preemption so that, if $Z \neq 0$, the second customer to be served is C_{Z+1}. Any other customers that arrive while C_{Z+1} is being served will also be served before C_Z. Now let N be the random variable for the number of customers served during a busy period and let N_i be the number of customers served between the instants at which C_{i+1} commences service and C_i commences service ($1 \leq i \leq Z$). Then $N_1, \ldots, N_Z$ are independent and identically distributed as N. This is because the sets of customers counted by $N_Z, N_{Z-1}, \ldots, N_1$ are disjoint and (excluding $C_{Z+1}, C_Z, \ldots, C_2$ respectively) arrive consecutively after C_{Z+1}. Thus,

$$N \sim \begin{cases} 1 + N_Z + N_{Z-1} + \ldots + N_1 & \text{if } Z \geq 1 \\ 1 & \text{if } Z = 0 \end{cases}$$

(The symbol $\sim$ denotes 'equal in distribution'.) Now, denoting the busy time random variable by H, we have

$$H \sim \begin{cases} B + H_Z + H_{Z-1} + \ldots + H_1 & \text{if } Z \geq 1 \\ B & \text{if } Z = 0 \end{cases}$$

where H_i is the length of the interval between the instants at which C_{i+1} and C_i commence service, $1 \leq i \leq Z$. Moreover, the H_i are independent

random variables, each distributed as H, and also independent of B. This is because the customers that arrive and complete service during the intervals H_i are disjoint. Thus

$$\begin{aligned} H^*(s) &= E[E[E[e^{-sH} \mid Z, B] \mid B]] \\ &= E[E[E[e^{-s(B+H_1+\ldots+H_Z)} \mid Z, B] \mid B]] \\ &= E[E[e^{-sB}E[e^{-sH}]^Z \mid B]] \\ &= E[e^{-sB}E[H^*(s)^Z \mid B]] \\ &= E[e^{-sB}e^{-\lambda B(1-H^*(s))}] \end{aligned}$$

since Z (conditioned on B) is Poisson with mean λB. Thus we obtain

$$H^*(s) = B^*(s + \lambda(1 - H^*(s)))$$

Although this equation cannot be solved in general for $H^*(s)$, we can obtain the moments of busy time by differentiating at $s = 0$. For example, the mean busy period b is given by $b = -H^{*\prime}(0) = -B^{*\prime}(0)[1 + \lambda(-H^{*\prime}(0))] = (1 + \lambda b)\mu^{-1}$ since $H^*(0) = 1$, and so $b = (\mu - \lambda)^{-1}$, yielding the M/M/1 queue result (where μ is the reciprocal of mean service time). The above technique, in which a time delay is defined in terms of independent, identically distributed time delays, is often called 'delay cycle analysis' and is due to Takacs [40].

2.3. *Waiting times in LCFS queues*

Now consider waiting times under LCFS disciplines. For the preemptive-resume variant, we note that a task's waiting time is independent of the queue length it faces on arrival, since the whole of the queue already there is suspended until after this task completes service. Thus, without loss of generality, we may assume that the task arrives at an idle server. Waiting time then becomes identical to the busy period. We therefore conclude that the waiting time distribution in a LCFS-PR M/G/1 queue has Laplace-Stieltjes transform $H^*(s)$. For LCFS without preemption we can modify the busy period analysis. First, if a task arrives at an empty queue, its waiting time is the same as a service time. Otherwise, its queueing time Q is the sum of the residual service time R of the customer in service and the service times of all other tasks that arrive before it commences service. This definition is almost the same as that of a busy period given above. The only differences are that the time spent in service by the initial customer C_1' (C_1 above) is not a service time but a residual service time and the random

variable Z' (Z above) is the number of customers that arrive whilst C_1' is in residual service. Proceeding as before, we obtain

$$Q \sim \begin{cases} R + H'_Z + H_{Z'-1} + \ldots + H_1 & \text{if } Z' \geq 1 \\ R & \text{if } Z' = 0 \end{cases}$$

We therefore derive

$$Q^*(s) = R^*(s + \lambda(1 - H^*(s)))$$

But since R is a forward recurrence time [24], $R^*(s) = \mu[1 - B^*(s)]/s$. Thus,

$$Q^*(s) = \frac{\mu(1 - H^*(s))}{s + \lambda(1 - H^*(s))}$$

Finally, since a customer arrives at an empty queue with probability $1 - \rho$ at equilibrium, we obtain

$$\begin{aligned} W^*(s) &= (1 - \rho)B^*(s) + \rho B^*(s)Q^*(s) \\ &= B^*(s)\left(1 - \rho + \frac{\lambda(1 - H^*(s))}{s + \lambda(1 - H^*(s))}\right) \end{aligned}$$

since waiting time is the sum of the independent queueing time and service time random variables.

To illustrate, compare the response time variability in an M/G/1 queue, under FCFS and non-preemptive LCFS queueing disciplines. We can do this to a great extent by comparing the first two moments, which are obtained by differentiating the respective formulae for $W^*(s)$ at $s = 0$. We obtain the same result for the mean waiting time, as expected from Little's result since the mean queue lengths are the same under each discipline[b]. However, it turns out that the second moment of waiting time for FCFS discipline is $(1 - \rho)$ times that for LCFS. Thus, LCFS discipline suffers a much greater variability as ρ approaches 1, i.e. as the queue begins to saturate.

[b]Notice that LCFS-PR gives a different mean waiting time in general. This is reasonable because we cannot expect the mean queue length (and hence, using Little's result, the mean waiting time) to be the same since the preemption invalidates the argument used to derive the queue length distribution. Intuitively, the average amount of work left to do in the queue should be the same, but since the queue will, in general, contain partially served customers, its expected length should be different. In fact, as we saw above, mean waiting time is the same as for the case of an M/M/1 queue. This is a consequence of the memoryless property of the exponential distribution: a partially served customer is stochastically identical to one that has received no service.

2.4. *Waiting times with Processor-Sharing discipline*

The problem with the PS discipline is that the rate at which a customer receives service during his sojourn at a server varies as the queue length changes due to new arrivals and other departures. Thus, we begin by analysing the waiting time of a customer with a given service time requirement in a PS M/M/1 queue.

Proposition 1: *In a PS M/M/1 queue with fixed arrival rate λ and fixed service rate μ, the Laplace transform of the waiting time probability density function, conditional on a customer's service time being x, is*

$$W^*(s \mid x)\frac{(1-\rho)(1-\rho r^2)e^{-[(1-r)\lambda+s]x}}{(1-\rho r)^2-\rho(1-r)^2e^{-[\mu/r-\lambda r]x}}$$

where r is the smaller root of the equation $\lambda r^2-(\lambda+\mu+s)r+\mu=0$ and $\rho=\lambda/\mu$.

This result, proved in [24], was first derived in [10]. We can now obtain the Laplace transform of the unconditional waiting time density as

$$W^*(s)=\int_0^\infty W^*(s \mid x)\mu e^{-\mu x}\mathrm{d}x$$

The essential technique used in the proof of Proposition 1 splits the waiting time in an M/M/1 queue into an infinitesimal initial interval and the remaining waiting time. In fact, the technique is quite general, applying to more disciplines than PS. In particular, it can be used to find the Laplace transform of the waiting time density in an M/M/1 queue with 'random' discipline, FCFS discipline with certain queue-length-dependent service rates and in M/M/1 G-queues (with negative customers) [22,25]. We outline the method in the following section, with a more general, recent application.

3. MM CPP/GE/c G-queues: semi-numerical Laplace transform inversion

The infinitesimal initial interval (III) approach to finding passage time densities in Markov processes is well illustrated in a recent result that determines the waiting time density in the MM CPP/GE/c G-queue, abbreviated to MBG (modulated, batched G-queue) [21,27]. The queue is Markovian, has c servers, FCFS queueing discipline and RCH 'killing strategy' whereby the customer at the head of the queue (i.e. in service), if any, is removed

by a negative arrival[c]. The queue's parameters are modulated by an N-phase Markov process with generator matrix Q. In phase k, the arrival rate of positive (respectively negative) customers is λ_k (respectively κ_k) and the parameters of their geometric batch sizes are θ_k (respectively ρ_k), $1 \leq k \leq N$. Similarly, the service time is generalised exponential with parameter μ_k and batch size parameter ϕ_k in phase k. The following diagonal matrices are defined:

- Λ : the positive arrival rate matrix, $\Lambda_{kk} \equiv \lambda_k$;
- Θ : the positive batch size geometric distribution parameter matrix, $\Theta_{kk} \equiv \theta_k$;
- M : the service rate matrix, $M_{kk} \equiv \mu_k$;
- Φ : the (truncated) service completion batch size geometric distribution parameter matrix, $\Phi_{kk} \equiv \phi_k$;
- K : the negative arrival rate matrix, $K_{kk} \equiv \kappa_k$;
- R : the negative batch size geometric distribution parameter matrix, $R_{kk} \equiv \rho_k$.

The equilibrium state probability for state (j, k), representing queue length $j \geq 0$ and phase k is denoted $[\mathbf{v}_j]_k$, i.e. the kth component of the state probability vector $\mathbf{v}_j$. The solution for $\{\mathbf{v}_j \mid j \geq 0\}$ is assumed to have been determined by the method of spectral analysis [37], which yields a solution for $j \geq c$ of the form:

$$\mathbf{v}_j = \sum_{k=1}^{N} a_k \xi_k^j \boldsymbol{\psi}_k$$

where $(\xi_k, \boldsymbol{\psi}_k)$ is the kth eigenvalue-eigenvector pair of the method and the a_k are scalar constants – see [9].

The III method yields the following solution for the Laplace transform of the sojourn-time density, after simplification.

Proposition 2: *The sojourn-time density in the above MBG-queue has Laplace transform:*

$$L(s) = \left(\sum_{j=0}^{c-1} \left(\sum_{q=0}^{j} \mathbf{v}_q \Theta^{-q} \right) \Theta^j \right) (I - \Theta)(\Lambda/\lambda^*)\mathbf{L}_0(s) + \sum_{k=1}^{N} a_k \xi_k \boldsymbol{\psi}_k (\xi_k I - \Theta)^{-1} (I - \Theta)(\Lambda/\lambda^*)\mathbf{D}(\xi_k, s)$$

[c]Other killing strategies are also considered in [21].

for certain vector-functions of s given in [21].

The proof idea is the following. Consider the passage of a special 'tagged' customer through the queue. Ultimately, this customer will either be served or killed by a negative customer; we require the probability distribution function of the sojourn time of customers that are not killed, i.e. the time elapsed between the arrival instant and the completion of service. Let the random variables $I(x)$ and $A(x)$ denote respectively the phase of the modulating Markov chain and the number of customers ahead of the tagged customer at time x and let T denote the time remaining, without loss of generality at time 0, up to the departure of the tagged customer. For $j \geq 0$, we define the probability distributions $\mathbf{F}_j(t) = (F_{1j}(t), \ldots, F_{Nj}(t))$ where, for $1 \leq k \leq N$,

$$F_{kj}(t) = P(T \leq t \mid I(0) = k, A(0) = j)$$

Now, when the state is (j, k), we consider an initial small interval of length h and derive an expression for $\mathbf{F}_j(t+h)$ in terms of $\{\mathbf{F}_a(t) \mid a \geq 0\}$. By the Markov property and stationarity, we can write, for $j \geq c$:

$$\begin{aligned}
\mathbf{F}_j(t+h) = (I + Qh - Kh - cMh)\mathbf{F}_j(t) & \\
+\ h \sum_{s=1}^{j-c+1} K(I-R)R^{s-1}\mathbf{F}_{j-s}(t) & \\
+\ hKR^{j-c+1}\mathbf{0} & \\
+\ h \sum_{s=1}^{j-c+1} cM(I-\Phi)\Phi^{s-1}\mathbf{F}_{j-s}(t) & \\
+\ hcM\Phi^{j-c+1}\mathbf{e} + o(h) & \qquad (1)
\end{aligned}$$

where $\mathbf{0}$ and $\mathbf{e}$ are the zero-vector and $(1, \ldots, 1)$ of length N respectively. The details appear in [21], but to clarify, consider the special case of the Markov modulated M/M/1 G-queue with no batches, obtained by setting $c = 1, \Theta = \Phi = R = 0$. This gives the much simpler equation

$$\begin{aligned}
\mathbf{F}_j(t+h) = (I + Qh - Kh - Mh)\mathbf{F}_j(t) & \\
+\ hK\mathbf{F}_{j-1}(t) + h\delta_{j0}K\mathbf{0} + hM\mathbf{F}_{j-1}(t) + h\delta_{j0}M\mathbf{e} + o(h) &
\end{aligned}$$

Returning to the general case, taking the term $I\mathbf{F}_j(t)$ to the left hand side in equation 1 and dividing by h now yields the vector differential-

difference equation:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{F}_j(t)}{\mathrm{d}t} &= (Q-K-cM)\mathbf{F}_j(t) \\ &+ K(I-R)\sum_{s=1}^{j-c+1} R^{s-1}\mathbf{F}_{j-s}(t) \\ &+ cM(I-\Phi)\sum_{s=1}^{j-c+1} \Phi^{s-1}\mathbf{F}_{j-s}(t) \\ &+ cM\Phi^{j-c+1}\mathbf{e} \end{aligned} \tag{2}$$

For $j < c$, the tagged customer is in service and so his progress is influenced solely by the negative arrivals and the service completion time at his particular server. Thus, $\mathbf{F}_j(t) = \mathbf{F}_0(t)$ for $0 \le j < c$ and we have:

$$\mathbf{F}_0(t+h) = (I + Qh - Mh - (K/c)h)\mathbf{F}_0(t) + hM\mathbf{e} + h(K/c)\mathbf{0} + o(h)$$

This yields the vector differential equation

$$\frac{\mathrm{d}\mathbf{F}_0(t)}{\mathrm{d}t} = (Q - M - K/c)\mathbf{F}_0(t) + M\mathbf{e} \tag{3}$$

which has solution

$$\mathbf{F}_0(t) = \left(1 - e^{-(M+K/c-Q)t}\right)(M + K/c - Q)^{-1}M\mathbf{e}$$

We can now derive recurrence formulas for the Laplace transforms $\mathbf{L}_j(s)$ of the $\mathbf{F}_j(t)$, which we solve using the generating function method – the vector $\mathbf{D}$ is defined by: $\mathbf{D}(z,s) = \sum_{j=c}^{\infty} \mathbf{L}_j(s) z^j$.

The surprising result obtained for this complex queue is that the Laplace transform can be inverted analytically for any given numerical parameterisation.

Proposition 3: *The Laplace transform $L(s)$ is analytically invertible and gives an unconditional sojourn-time density function which is a mixture of exponential and Erlang densities as well as those of the types $e^{at}\cos(bt)$ and $e^{at}\sin(bt)$, for real numbers $a \le 0$ and $b > 0$.*

Proof The expression of Proposition 2 for $L(s)$ is a sum of terms, each of which depends on s through either the function $\mathbf{L}_0(s)$ or $\mathbf{D}(z,s)$, for some particular value of z which is an eigenvalue in the spectral analysis solution. These eigenvalues are either real or occur in complex conjugate pairs. It is shown in [27] that both of these functions are sums of rational functions of the form $u/(sv)$, where u and v are polynomials in s with real

coefficients. Consequently, v has real roots, $\{x_1, \ldots, x_k\}$ say, and roots that form complex conjugate pairs, $\{y_1 \pm iz_1, \ldots, y_l \pm iz_l\}$, where all $x_i, y_j < 0$ for $1 \leq i \leq k$ and $1 \leq j \leq l$. The polynomial v is of higher order than u and it is therefore possible to re-write each term using partial fractions. Further routine analysis yields the stated result. ♠

This is a semi-numerical inversion in that the parameters of the resulting mixtures of exponential and Erlang densities must be computed numerically, but subsequently a solution can be found directly as a function of time. There is no need to invert the Laplace transform numerically.

4. Time delays in networks of queues

The analysis of networks of queues is entirely different to that of a single server queue – even a stationary Markovian network. This is because, although we may know the distribution of the queue lengths at the time of arrival of a given (tagged) customer at the first queue in his path (by the Random Observer Property or the Job Observer Property), we cannot assume this stationary distribution exists upon arrival at subsequent queues. The reason is that the arrival times at the subsequent queues are only finitely later than the arrival time at the first queue. Hence, the state existing at the subsequent arrival times must be conditioned on the state that existed at the time of arrival at the first queue. Effectively, a new time origin is set at the first arrival instant, with known initial joint queue length probability distribution – usually a stationary distribution. Even in open networks with no feedback, where it is easy to see that all arrival processes are Poisson, this conditioning cannot be overlooked and we cannot assume all queues on a path are independent and in an equilibrium state at the arrival times of the tagged customer. This is in contrast to Jackson's Theorem because we are not considering the queues at the same time instant. However, things are not quite as hopeless as they seem. First, we can prove that the FCFS queues in an *overtake-free* path in a Markovian open network behave as if they were independent and in equilibrium when observed at the successive arrival times of a tagged customer. By an overtake-free path, or a path with *no overtaking*, we mean that a customer following this path will depart from its last queue before any other customer that joins any queue in that path after the said customer. Surprisingly, a similar result holds for overtake-free paths in closed networks, e.g. all paths in networks with a tree-like structure – see Figure 3. In the next two subsections, we

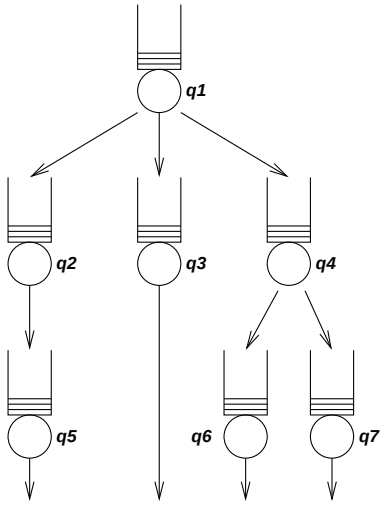

Fig. 1. An open tree-like network of queues

consider respectively those open and closed networks for which a solution for the time delay density along a given path can be derived.

4.1. *Open networks*

The simplest open network structure we can consider is a series of tandem queues. In this case, the distribution of the time delay of a customer passing through them is the convolution of the stationary waiting time distributions at each queue in the series considered in isolation. This follows from the following stronger result.

Proposition 4: *In a tandem series of stationary M/M/1 queues with fixed-rate servers and FCFS discipline, the waiting times of a given customer in each queue are independent.*

Proof First we claim that the waiting time of a tagged customer, C say, in a stationary M/M/1 queue is independent of the departure process before the departure of C. This is a direct consequence of the reversibility of the M/M/1 queue.

To complete the proof, let A_i and T_i denote C's time of arrival and waiting time respectively at queue i in a series of m queues ($1 \leq i \leq m$). Certainly, by our claim, T_1 is independent of the arrival process at queue

2 before A_2 and so of the queue length faced by C on arrival at queue 2. Thus, T_2 is independent of T_1. Now, we can ignore customers that leave queue 1 after C since they cannot arrive at (nor influence the rate of) any queue in the series before C, again because all queues have single servers and FCFS discipline. Thus, T_1 is independent of the arrival process at queue i before A_i and so of T_i for $2 \leq i \leq m$. Similarly, T_j is independent of T_k for $2 \leq j < k \leq m$. ♠

Observe that if service rates varied with queue length, we could not ignore customers behind a given tagged customer, even though they could not overtake, because they would influence the service rate received by the tagged customer.

From the proposition above it follows that, since the waiting time probability density at the stationary queue i with service rate μ_i (considered in isolation) has Laplace transform $(\mu_i - \lambda)/(s + \mu_i - \lambda)$ when the external arrival rate is λ, the probability density of the time to pass through the whole series of m queues is the convolution of these densities, with Laplace transform $\prod_{i=1}^{m}(\mu_i - \lambda)/(s + \mu_i - \lambda)$. There is one obvious generalisation of this result: the final queue in the series need not be M/M/1 since we are not concerned with its output. Also, the same result holds, by the same reasoning, when the final queue is M/G/c for $c > 1$. Moreover, Proposition 4 generalises to *tree-like networks* which are defined as follows and illustrated in Figure 1. A tree-like network consists of:

- a linear *trunk segment* containing one or more queues in tandem, the first being called the *root* queue;
- a number (greater than or equal to zero) of disjoint *subtrees*, i.e. tree-like subnetworks, such that customers can pass to the roots of the subtrees from the last queue in the trunk segment or else leave the network with specified routing probabilities.

The *leaf* queues (or just *leaves*) are those from which customers leave the network.

The proof of Proposition 4, extended to tree-like networks, carries through unchanged since every path in the network is overtake-free. Hence we can ignore the customers that leave any queue on the path after the tagged customer. Indeed, we can generalise further to overtake-free paths in any Markovian open network for the same reason. Conditional on the choice of path of queues numbered, without loss of generality, $1, \ldots, m$, the Laplace transform of the passage time density is the same as for the tandem

queue of m servers considered above, but with each arrival rate λ_i at the ith queue adjusted to the actual traffic rate.

To generalise the network structure further leads to serious problems and solutions have been obtained only for special cases. The simplest case of a network with overtaking is the network of Figure 2.

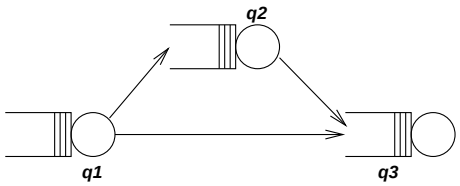

Fig. 2. A three-node network with overtaking

In this network the path of queues numbered $\{q1, q3\}$ is overtake-free and so the passage time density can be obtained as described above. However, overtaking is possible on the path $\{q1, q2, q3\}$ since when the tagged customer C is at queue 2, any customers departing queue 1 (after C) can reach queue 3 first. The arrival processes to every queue in this network are independent Poisson, by Burke's theorem together with the decomposition and superposition properties of Poisson processes. However, this is not sufficient for the passage time distribution to be the convolution of the stationary sojourn-time distributions at each queue on a path with overtaking: and so the proof of Proposition 4 breaks down. This particular problem has been solved by considering the state of the system at the departure instant of the tagged customer from server 1 and using complex variable methods in an III analysis [36]. A similar analysis is required – for similar reasons – to analyse a tandem pair of queues with negative customers [26]. In this case, negative arrivals at the second queue allow the first queue to influence the sojourn time of a tagged customer in the second; departures from the first queue offer a degree of 'protection'. Although these networks are solved problems, their results are almost numerically intractable; more general networks appear mathematically intractable as well.

4.2. *Closed networks*

As for the case of open networks, we begin with the simplest case, a cyclic network that comprises a tandem network with departures from its last queue fed back into the first queue. There are no external arrivals and

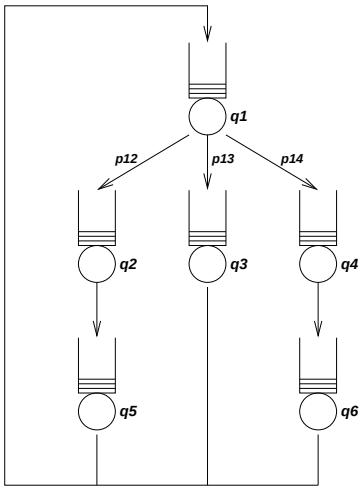

Fig. 3. A closed tree-like network and its routing probabilities

hence a constant population. Again, all service disciplines are FCFS and all service rates are constant.

We solve for the Laplace transform of the cycle time probability density function by considering a dual network, viz. the tandem, open network consisting of the same servers $1, \ldots, m$ with no external arrivals. Eventually, therefore, the dual network has no customers, i.e. its state is $\mathbf{0} = (0, \ldots, 0)$, the empty state, with probability 1. All other states with one or more customers are transient. Now, given that the state immediately after the arrival of the tagged customer at queue 1 is u, the ensuing cycle time in the closed network is the same as the time interval between the dual network entering states u and $\mathbf{0}$ – the (first) passage time from u to $\mathbf{0}$. This is so because there is no overtaking and service rates are constant. Thus the progress of the tagged customer in its cycle cannot be influenced by any customer behind it. We only need consider customers ahead of the tagged customer and can ignore those recycling after leaving the last queue. We therefore seek the density of the passage time from state u to $\mathbf{0}$ in the dual network, $f(t \mid u)$, where i is a state of the form $(i_1, \ldots, i_m)$ with $i_1 > 0$, corresponding to the tagged customer having just arrived at server 1. We know the probability distribution of the state seen by the tagged customer on arrival at the first queue by the Job Observer Property [24] and so can calculate the cycle time density by deconditioning f.

Given a cyclic network of population n, let the state space of the dual network be $S_n = \{(u_1, \ldots, u_m) \mid 0 \leq u_i \leq n, 1 \leq i \leq m; \sum_{i=1}^m u_i \leq n\}$ and define, for $u \in S_n$,

$$\lambda_u = \sum_{i=1}^{m} \mu_i \epsilon(u_i)$$

where μ_i is the service rate of server i, $\epsilon(n) = 1$ if $n > 0$ and $\epsilon(0) = 0$. Thus, λ_u is the total service rate in state u, i.e. the total instantaneous rate out of state u in the Markov process defining the dual network. The holding time in state u is an exponential random variable with parameter λ_u and so has a density with Laplace transform $\lambda_u/(s + \lambda_u)$. Given that the network next enters state v after u, the passage time from u to $\mathbf{0}$ is the sum of the holding time in state u and the passage time from v to $\mathbf{0}$. Thus the density of the passage time from u to $\mathbf{0}$ has Laplace transform $L(s \mid u)$ given by the equations

$$\begin{aligned} L(s \mid u) &= \sum_{v \in S_n} q_{uv} \frac{\lambda_u}{s + \lambda_u} L(s \mid v) \qquad (u \neq \mathbf{0}) \\ L(s \mid \mathbf{0}) &= 1 \end{aligned}$$

where q_{uv} is the one-step transition probability from state u to v. Now let $\mu(u, v)$ be the rate of the server from which a departure causes the state transition $u \to v$. Then $q_{uv} = \mu(u, v)/\lambda_u$ and, writing $q^*_{uv} = \mu(u, v)/(s + \lambda_u)$, we have the matrix-vector equation

$$\mathbf{L} = Q^* \mathbf{L} + \mathbf{1}_0$$

where $\mathbf{L} = (L(s \mid u) \mid u \in S_n), Q^* = (q^*_{uv} \mid u, v \in S_n)$ and $\mathbf{1}_w$ is the unit vector with component corresponding to state w equal to 1, the rest 0. Using this equation and deconditioning on the initial state u, we obtain a product-form for the Laplace transform of cycle time density.

This approach extends directly to cycle times in closed tree-like queueing networks. Such networks are defined in the same way as open tree-like networks except that customers departing from leaf-queues next visit the root queue. Clearly such networks have the no-overtaking property and if paths are restricted to start at one given server (here the root), they define the most general class for which it holds. Consider a closed tree-like network with M nodes and population N, in which node i has service rate μ_i and visitation rate (proportional to) v_i, $1 \leq i \leq M$, defined by $v_i = \sum_{j=1}^M v_j p_{ji}$ where p_{ji} is the probability that a customer leaving node j next visits

node i. Let G be the network's normalising constant function for the joint equilibrium state probabilities, i.e. at population k,

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

where $x_i = v_i/\mu_i$. Without loss of generality, the root node is numbered 1 and we define the cycle time random variable to be the elapsed time between a customer's successive arrival instants at node 1. Then we have the following result [18,29,13].

Proposition 5: *For the above closed tree-like network, the Laplace transform of the cycle time density function, conditional on choice of path* $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ *(*$m \leq M, z_1 = 1$*) is*

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

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

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

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

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

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

Now, the Laplace transforms in the inner sum are products of the Laplace transforms of Erlang densities. Moreover, their coefficients are geometric. Such transforms can be inverted analytically. In the simplest case, all the

servers on the overtake-free path are identical, i.e. have the same rate, and the inversion can be done by inspection, as in Section 4.2.2 and [24]. In the case that the μ_i are all distinct ($1 \leq i \leq m$), the density function is given by the following proposition, derived in [19]; in [20], the question of degenerate μ_i (when not all are equal) is considered[d].

Proposition 6: *If the servers in an overtake-free path* $(1, 2, \ldots, m)$ *have distinct service rates* $\mu_1, \mu_2, \ldots, \mu_m$*, the passage time density function, conditional on the choice of path, is*

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

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

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

$K^m(j,l)$ is just a normalising constant that may be computed efficiently, along with $G_m(n-c-1)$ and $G(n-1)$, by Buzen's algorithm [7]. Thus we define the recursive function k, for real vector $\mathbf{y} = (y_1, \ldots, y_a)$ and integers a, b $(0 \leq a \leq M,\ \ 0 \leq b \leq N-1)$ by:

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

Then we have

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

[d]Essentially, we start with the case of distinct rates and successively combine any two servers with equal rates. The combination involves manipulation of the summations and reduces the problem to two similar problems on networks with one less node in the overtake-free path. Thus, in each step, one degenerate server is removed until all the remaining problems are on paths with distinct rates.

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

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

In fact Propositions 5 and 6 hold for any overtake-free path in a closed Jackson queueing network – recall the preceding discussion.

4.2.1. *Cyclic networks*

For a cyclic network of M exponential servers with distinct rates $\mu_1, \ldots, \mu_M$ and population N, cycle time distribution is

$$\frac{\left(\prod_{i=1}^{M} \mu_i\right) t^{N-1}}{(N-1)!G(N)} \sum_{j=1}^{M} \frac{e^{-\mu_j t}}{\prod_{i \neq j}(\mu_i - \mu_j)}$$

This follows by setting $v_1 = \ldots = v_M = 1$ in Proposition 6, so that all the terms $K^m(j,i)$ are zero except when $n_k = 0$ for all k, i.e. when $i = 0$. Finally, note there is only one partition of the state space, namely the one with all $N-1$ customers at the servers $1, \ldots, M$. Thus we have $G_M(n) = 1$ if $n = 0$ and $G_M(n) = 0$ if $n > 0$, so that only terms with $c = N - 1$ give a non-zero contribution.

4.2.2. *Paths with service rates all equal*

When all the service rates in the path are the same, equal to μ say, the Laplace transform of passage time is a mixed sum of Erlang densities of the form $[\mu/(s+\mu)]^{c+m}$. Each term can therefore be inverted by inspection and we get:

Proposition 7: *If the servers in overtake-free path $1, \ldots, m$ in the network of Proposition 5 all have service rate μ, the path's time delay density function is*

$$\frac{\mu^m e^{-\mu t}}{G(n-1)} \sum_{c=0}^{n-1} G_m(n-c-1) G^m(c) \mu^c \frac{t^{c+m-1}}{(c+m-1)!}$$

where $G^m(k)$ is the normalising constant for the subnetwork comprising servers $1, \ldots, m$ only, with population $k \geq 0$.

From this result we can immediately obtain formulae for moments higher than the mean of a customer's transmission time.

Corollary 1: *For a path of equal rate servers, message transmission time has kth moment equal to*

$$\frac{1}{\mu^k G(n-1)} \sum_{c=0}^{n-1} G_m(n-c-1) G^m(c)(c+m)\dots(c+m-k+1)$$

5. Passage times in continuous time Markov chains

In this section we consider the analysis of passage times in arbitrary finite continuous time Markov chains (CTMCs). While such chains can be specified manually, it is more usual for them to be generated automatically from one of several high-level modelling formalisms, such as queueing networks, stochastic Petri nets or stochastic process algebras. Since the memory and processing power of a single workstation is easily overwhelmed by realistic models (which may typically contain tens of millions of states or more), our approach is a parallel uniformization-based algorithm that uses hypergraph partitioning to minimise interprocessor communication [12].

5.1. *First Passage Times in CTMCs*

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

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

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

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

$T_{i\vec{j}}$ is a random variable with an associated probability density function $f_{i\vec{j}}(t)$. To determine $f_{i\vec{j}}(t)$ we must convolve the exponentially distributed state holding times over all possible paths (including cycles) from state i into any of the states in the set $\vec{j}$. As shown in the next subsection, the problem can also be readily extended to multiple initial states by weighting first passage time densities.

5.2. *Uniformization*

Uniformization [16,39] 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 (DTMC), after normalisation of the rows, together with an associated Poisson process of rate q. The one-step DTMC transition matrix P is given by:

$$P = Q/q + I$$

where $q > \max_i |q_{ii}|$ (to ensure that the DTMC is aperiodic). The number of transitions in the DTMC that occur in a given time interval is given by a Poisson process with rate q.

While uniformization is normally used for transient analysis, it can also be employed for the calculation of response time densities [35,38]. We add an extra, absorbing state to our uniformized chain, which is the sole successor state for all target states (thus ensuring we calculate the *first* passage time density). We denote by P' the one-step transition matrix of the modified, uniformized chain. Recalling that the time taken to traverse a path with n hops in this chain will have an Erlang distribution with parameters n and q, the density of the time taken to pass from a set of source states $\vec{i}$ into a set of target states $\vec{j}$ is given by:

$$f_{\vec{i}\vec{j}}(t) = \sum_{n=1}^{\infty} \frac{q^n t^{n-1} e^{-qt}}{(n-1)!} \sum_{k \in \vec{j}} \pi_k^{(n)} \tag{4}$$

where

$$\pi^{(n+1)} = \pi^{(n)} P' \qquad \text{for } n \geq 0$$

with

$$\pi_k^{(0)} = \begin{cases} 0 & \text{for } k \notin \vec{i} \\ \pi_k / \sum_{j \in \vec{i}} \pi_j & \text{for } k \in \vec{i} \end{cases} \tag{5}$$

and in which $\boldsymbol{\pi}$ is any non-zero solution to $\pi = \pi P$. The corresponding passage time cumulative distribution function is given by:

$$F_{\vec{i}\vec{j}}(t) = \sum_{n=1}^{\infty} \left\{ \left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!} \right) \sum_{k \in \vec{j}} \pi_k^{(n)} \right\}. \tag{6}$$

Truncation is employed to approximate the infinite sums in Eq. 4 and Eq. 6, terminating the calculation when the Erlang term drops below a specified threshold value. Concurrently, when the convergence criterion

$$\frac{||\pi^{(n+1)} - \pi^{(n)}||_\infty}{||\pi^{(n)}||_\infty} < \epsilon \tag{7}$$

is met[e], for given tolerance ϵ, the steady state probabilities of P' are considered to have been obtained with sufficient accuracy and no further multiplications with P' are performed.

5.3. *Hypergraph partitioning*

The key opportunity for parallelism in the uniformization algorithm is the sparse matrix-vector product $\pi^{(n+1)} = \pi^{(n)}P'$ (or equivalently $\pi^{(n+1)T} = P'^T\pi^{(n)T}$, where the superscript T denotes the transpose operator. To perform these operations efficiently it is necessary to map the non-zero elements of P' onto processors such that the computational load is balanced and communication between processors is minimised. To achieve this, we use hypergraph-based partitioning techniques to assign matrix rows and corresponding vector elements to processors in a row-striped decomposition.

Hypergraphs are extensions of graph data structures that, until recently, were primarily applied in VLSI circuit design. Formally, a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined by a set of vertices $\mathcal{V}$ and a set of nets (or hyperedges) $\mathcal{N}$, where each net is a subset of the vertex set $\mathcal{V}$ [8]. In the context of a row-wise decomposition of a sparse matrix, matrix row i $(1 \leq i \leq n)$ is represented by a vertex $v_i \in \mathcal{V}$ while column j $(1 \leq j \leq n)$ is represented by net $N_j \in \mathcal{N}$. The vertices contained within net N_j correspond to the row numbers of the non-zero elements within column j, i.e. for matrix A, $v_i \in N_j$ if and only if $a_{ij} \neq 0$. The weight of vertex i is given by the number of non-zero elements in row i, while the weight of a net is its contribution to the hyperedge cut, defined as one less than the number of different partitions (in the row-wise decomposition) spanned by that net.

The overall objective of a hypergraph sparse matrix partitioning is to minimize the total hyperedge cut while maintaining a load balancing criterion. Like graph partitioning, hypergraph partitioning is NP-complete. However, there are a small number of hypergraph partitioning tools which implement fast heuristic algorithms, for example PaToH [8] and hMeTiS [28].

5.4. *Parallel Algorithm and Tool Implementation*

The process of calculating a response time density begins with a high-level model, which we specify in an enhanced form of the DNAmaca Markov

[e] $||\pi||_\infty = \max_i |\pi_i|$

Chain Analyser interface language [30,31]. This language supports the specification of queueing networks, stochastic Petri nets, stochastic process algebras and other models that can be mapped onto Markov chains. Next, a probabilistic, hash-based state generator [33] uses the high-level model description to produce the generator matrix Q of the model's underlying Markov chain as well as a list of the initial and target states. Normalised weights for the initial states are then determined from Eq. 5, which requires us to solve $\pi Q = 0$. This is readily done using any of a variety of steady-state solution techniques (e.g. [11,32]). From Q, P'^T is constructed by uniformizing and transposing the underlying Markov chain and by adding the extra, terminal state that becomes the sole successor state of all target states. Having been converted into an appropriate input format, P'^T is then partitioned using a hypergraph or graph-based partitioning tool.

The pipeline is completed by our distributed response time density calculator, which is implemented in C++ using the Message Passing Interface (MPI) [17] standard. This means that it is portable to a wide variety of parallel computers and workstation clusters.

Initially each processor tabulates the Erlang terms for each t-point required (cf. Eq. 4). Computation of these terms terminates when they fall below a specified threshold value. In fact, this is safe to use as a truncation condition for the entire passage time density expression because the Erlang term is multiplied by a summation which is a probability. The terminating condition also determines the maximum number of hops m used to calculate the right-hand factor, a sum which is independent of t.

Each processor reads in the rows of the matrix P'^T that correspond to its allocated partition into two types of sparse matrix data structure and also computes the corresponding elements of the vector $\pi^{(0)}$. *Local* non-zero elements (i.e. those elements in the diagonal matrix blocks that will be multiplied with vector elements stored locally) are stored in a conventional compressed sparse row format. *Remote* non-zero elements (i.e. those elements in off-diagonal matrix blocks that must be multiplied with vector elements received from other processors) are stored in an ultrasparse matrix data structure – one for each remote processor – using a coordinate format. Each processor then determines which vector elements need to be received from and sent to every other processor on each iteration, adjusting the column indices in the ultrasparse matrices so that they index into a vector of received elements. This ensures that a minimum amount of communication takes place and makes multiplication of off-diagonal blocks with received vector elements very efficient.

The vector $\pi^{(n)}$ is then calculated for $n = 1, 2, 3, \ldots, m$ by repeated sparse matrix-vector multiplications of form $\pi^{(n+1)T} = P'^T \pi^{(n)T}$. Actually, fewer than m multiplications may take place since a test for steady state convergence is made after every iteration (cf. Eq. 7); if the convergence criterion is satisfied, the matrix-vector multiplication is not performed and we set $\pi^{(n+1)T} = \pi^{(n)T}$ in subsequent iterations.

For each matrix-vector multiplication, each processor begins by using non-blocking communication primitives to send and receive remote vector elements, while calculating the product of local matrix elements with locally stored vector elements. The use of non-blocking operations allows computation and communication to proceed concurrently on parallel machines where dedicated network hardware supports this effectively. The processor then waits for the completion of non-blocking operations (if they have not already completed) before multiplying received remote vector elements with the relevant ultrasparse matrices and adding their contributions to the local matrix-vector product cumulatively.

From the resulting local matrix-vector products each processor calculates and stores its contribution to the sum $\sum_{k \in \vec{j}} \pi_k^{(n)}$. After m iterations have completed, these sums are accumulated onto an arbitrary master processor where they are multiplied with the tabulated Erlang terms for each t-point required for the passage time density. The resulting points are written to a disk file and are displayed using the GNUplot graph plotting utility.

5.5. *Numerical Example*

As an example we consider the cycle time in the closed tree-like queueing network of Figure 3. This network 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: $1,\ p_{12},\ p_{13},\ p_{14},\ p_{12},\ p_{14}$. Here we set $\{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6\} = \{3, 5, 4, 6, 2, 1\}$ and $\{p_{12}, p_{13}, p_{14}\} = \{0.2, 0.5, 0.3\}$. As described in Section 4.2, 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, and can be rapidly computed.

To compute the cycle time density in this network in terms of its underlying Markov Chain using the uniformization technique described in this paper requires the state vector to be augmented by 3 extra components so that a "tagged" customer can be followed through the system. The extra components are: the queue containing the tagged customer l, the position of the tagged customer in that queue k, and the cycle sequence number

c (an alternating bit, flipped whenever the tagged customer joins $q1$). For this augmented system with n customers, the underlying Markov chain has $12\binom{n+5}{6}$ states. Source states are those in which $l = 1$, $k = n_1 - 1$ and $c = 0$ while target states are those in which $l = 1$, $k = n_1 - 1$ and $c = 1$.

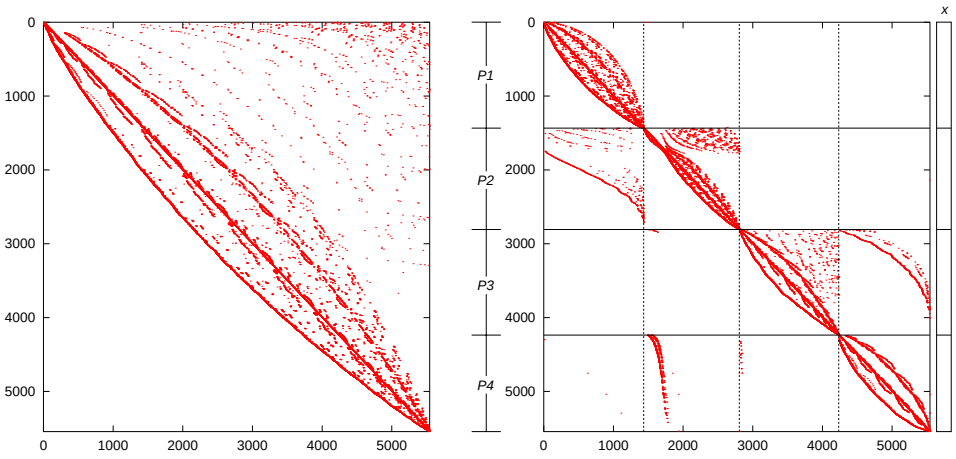

Fig. 4. Transposed P' matrix (left) and hypergraph-partitioned matrix (right) for the tree-like queueing network with 6 customers (5 544 states).

Table 1. Communication overhead in the queueing network model with six customers (left) and interprocessor communication matrix (right).

processor	non-zeros	local %	remote %	reused %
1	7 022	99.96	0.04	0
2	7 304	91.41	8.59	34.93
3	6 802	88.44	11.56	42.11
4	6 967	89.01	10.99	74.28

	1	2	3	4
1	-	407	-	4
2	3	-	16	181
3	-	-	-	12
4	-	1	439	-

For a small six customer system with 5 544 states, Figure 4 shows the resulting transposed P' matrix and associated hypergraph decomposition produced by hMeTiS. Statistics about the per-iteration communication associated with this decomposition are presented in Table 1. Around 90% of the non-zero elements allocated to each processor are local, i.e. they are multiplied with vector elements that are stored locally. The remote non-

zero elements are multiplied with vector elements that are sent from other processors. However, because the hypergraph decomposition tends to align remote non-zero elements in columns (well illustrated in the second block belonging to processor 4), reuse of received vector elements is good (up to 74%) with correspondingly lower communication overhead. The communication matrix on the right in Table 1 shows the number of vector elements sent between each pair of processors during each iteration (e.g. 181 vector elements are sent from processor 2 to processor 4).

Moving to a more sizeable model, the queueing network with 27 customers has an underlying Markov Chain with 10 874 304 states and 82 883 682 transitions. This model is too large to partition using a hypergraph partitioner on a single machine (even one with 2GB RAM), and there are currently no parallel hypergraph partitioning tools available. Consequently a lesser quality graph-based decomposition produced by the parallel graph partitioner ParMeTiS (running on the PC cluster) was chosen. It must be noted that this decomposition still offers a great reduction in communication costs over other methods available: a 16-way partition has an average of 95.8% local non-zero elements allocated to each processor and a reused received non-zero element average of 30.4%. Table 2 shows the per-iteration communication overhead for randomised (i.e. random assignment of rows to partitions), linear (i.e. simple in-order allocation of rows to processors such that the number of non-zeros assigned to each processor is the same) and graph-based allocations. The graph-based method is clearly superior, both in terms of number of messages sent and (especially) communication volume.

Table 2. Per-iteration communication overhead for various partitioning methods for the queueing network model with 27 customers on 16 processors.

Partitioning Method	Communication Overhead	
	Messages	Volume (MB)
randomised	240	450.2
linear	134	78.6
graph-based	110	19.7

Figure 5 compares the numerical and analytical cycle time densities (computed by Proposition 6) for the queueing network with 27 customers. Agreement is excellent and the results agree to an accuracy of 0.00001% over the time range plotted. The numerical density is computed in 968 seconds (16 minutes 8 seconds) for 875 iterations using 16 PCs. The memory used on each PC is just 84MB. It was not possible to compute the density on a single

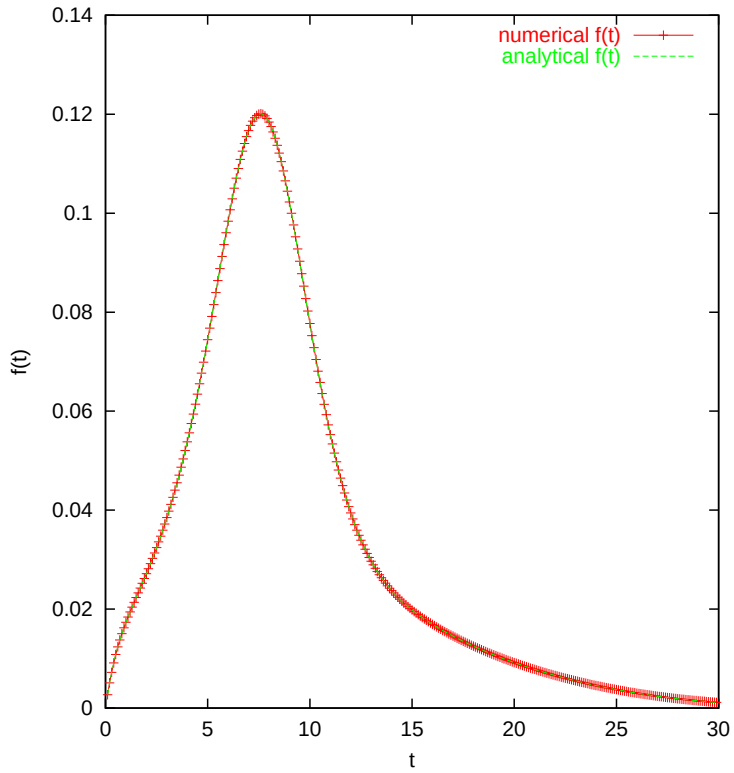

Fig. 5. Numerical and analytical cycle time densities for the tree-like queueing network of Figure 3 with 27 customers (10 874 304 states).

PC (with 512MB RAM) but the same computation on a dual-processor server machine (with 2GB RAM) required 5580 seconds (93 minutes).

6. Passage times in continuous time semi-Markov processes

Semi-Markov processes (SMPs) are a generalisation of Markov processes that allow for arbitrarily distributed state sojourn times, so that more realistic models can be described while still maintaining some of the analytical tractability associated with Markov models. This section summarises an iterative technique for passage time analysis of large, structurally unrestricted semi-Markov processes [23,5,6]. Our method is based on the calculation and subsequent numerical inversion of Laplace transforms and is amenable to a highly scalable distributed implementation. One of the biggest problems involved in working with semi-Markov processes is how to store the Laplace

transform of state sojourn times in an effective way, such that accuracy is maintained but representation explosion does not occur. We address this issue with a constant-space representation of a general distribution function based on the evaluation demands of the numerical inversion algorithm employed. Results for a distributed voting system model with up to 1.1 million states are presented and compared against simulation.

6.1. *First Passage Times in SMPs*

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

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

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

$$Z(t) = X_{N(t)}$$

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

$$\begin{aligned} R(i, j, t) &= \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \\ &= p_{ij} H_{ij}(t) \end{aligned}$$

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

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 ($t \geq 0$), the first passage time from a source state i at time t into a non-empty set of target states $\vec{j}$ is:

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

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

$$P_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j} \mid Z(0) = i\} \tag{8}$$

$P_{i\vec{j}}$ is a random variable with an associated probability density function $f_{i\vec{j}}(t)$. In general, the Laplace transform of $f_{i\vec{j}}$, $L_{i\vec{j}}(s)$, can be computed by solving a set of N linear equations:

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} r^*_{ik}(s) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} r^*_{ik}(s) \quad \text{for } 1 \leq i \leq N \tag{9}$$

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

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

Eq. (9) has a matrix-vector form, $A\tilde{x} = \tilde{b}$, where the elements of A are arbitrary complex functions; care needs to be taken when storing such functions for eventual numerical inversion (see Section 6.3). For example, when $\vec{j} = \{1\}$, Eq. (9) yields:

$$\begin{pmatrix} 1 & -r^*_{12}(s) & \cdots & -r^*_{1N}(s) \\ 0 & 1 - r^*_{22}(s) & \cdots & -r^*_{2N}(s) \\ 0 & -r^*_{32}(s) & \cdots & -r^*_{3N}(s) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -r^*_{N2}(s) & \cdots & 1 - r^*_{NN}(s) \end{pmatrix} \tilde{x} = \begin{pmatrix} r^*_{11}(s) \\ r^*_{21}(s) \\ r^*_{31}(s) \\ \vdots \\ r^*_{N1}(s) \end{pmatrix} \tag{10}$$

where $\tilde{x} = (L_{1\vec{j}}(s), L_{2\vec{j}}(s), \ldots, L_{N\vec{j}}(s))^T$. When there are multiple source states, denoted by the vector $\vec{i}$, the Laplace transform of the passage time distribution at steady-state is:

$$L_{\vec{i}\vec{j}}(s) = \sum_{k \in \vec{i}} \alpha_k L_{k\vec{j}}(s) \tag{11}$$

where the weight α_k is the probability at equilibrium that the system is in state $k \in \vec{i}$ at the starting instant of the passage. If $\tilde{\pi}$ denotes the steady-state vector of the embedded discrete-time Markov chain (DTMC) with one-step transition probability matrix $P = [p_{ij} \mid 1 \leq i,j \leq N]$, then α_k is given by:

$$\alpha_k = \begin{cases} \pi_k / \sum_{j \in \vec{i}} \pi_j & \text{if } k \in \vec{i} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

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

6.2. *Iterative passage time algorithm*

Recall the semi-Markov process, $Z(t)$, of Section 6.1, where $N(t)$ is the number of state transitions that have taken place by time t. We define the rth transition first passage time to be:

$$P_{i\vec{j}}^{(r)} = \inf\{u > 0 : Z(u) \in \vec{j} \mid N(u) \leq r, Z(0) = i\} \tag{13}$$

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

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

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

which may be computed as:

$$\tilde{L}_{\vec{j}}^{(r)}(s) = U(I + U' + U'^2 + \cdots + U'^{(r-1)})\,\tilde{e} \tag{14}$$

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

We include the initial U term in Eq. (14) so as to generate cycle times for cases such as $L_{ii}^{(r)}(s)$ which would otherwise register as 0, if U' were used instead.

From Eqs. (8) and (13):

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

Now, $L_{i\vec{j}}^{(r)}(s)$ can be generalised to multiple source states $\vec{i}$ using the normalised steady-state vector, $\tilde{\alpha}$, of Eq. (12):

$$\begin{aligned} L_{\vec{i}\vec{j}}^{(r)}(s) &= \tilde{\alpha}\tilde{L}_{\vec{j}}^{(r)}(s) \\ &= (\tilde{\alpha}U + \tilde{\alpha}UU' + \tilde{\alpha}UU'^2 + \ldots \\ &\qquad \ldots + \tilde{\alpha}UU'^{(r-2)} + \tilde{\alpha}UU'^{(r-1)})\,\tilde{e} \end{aligned} \tag{15}$$

The sum of Eq. (15) can be computed efficiently using sparse matrix-vector multiplications with a vector accumulator. At each step, the accumulator (initialised to αU) is postmultiplied by U' and αU is added. The worst-case time complexity for this sum is $O(N^2 r)$ versus the $O(N^3)$ of typical matrix inversion techniques.

Convergence of the sum in Eq. (15) is said to have occurred at a particular r, if for a given s-point:

$$|\mathrm{Re}(L_{\vec{ij}}^{(r+1)}(s) - L_{\vec{ij}}^{(r)}(s))| < \epsilon \text{ and}$$
$$|\mathrm{Im}(L_{\vec{ij}}^{(r+1)}(s) - L_{\vec{ij}}^{(r)}(s))| < \epsilon \quad (16)$$

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

6.3. *Laplace Transform Inversion*

The key to practical analysis of semi-Markov processes lies in the efficient representation of their generally distributed functions. Without care the structural complexity of the SMP can be recreated within the representation of the distribution functions.

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

As all our distribution manipulations take place in Laplace-space, we link our distribution representation to the Laplace inversion technique that we ultimately use. Our implementation supports two Laplace transform inversion algorithms: the Euler technique [2] and the Laguerre method [1] with modifications summarised in [21].

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

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

6.4. *Implementation*

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

Our implementation employs a distributed master-slave architecture similar to that of the Markovian passage time calculation tool of [21]. The master processor computes in advance the values of s at which it will need to know the value of $L_{\vec{ij}}(s)$ in order to perform the inversion. The s-values are then placed in a global work-queue to which the slave processors make requests. On making a request, slave processors are assigned the next available s-value and use this to construct the matrices U and U'. The iterative algorithm is then applied to calculate the truncated sum of Eq. (15) for that s-value. The result is returned to the master and cached (both in memory and on disk so that all computation is checkpointed), and once all values have been computed and returned, the final Laplace inversion calculations are made by the master. The resulting t-points can then be plotted on a graph. As inter-slave communication is not required, the algorithm exhibits excellent scalability.

6.5. *Numerical example*

We demonstrate the SMP analysis techniques of the previous sections with a semi-Markov Petri net model of a distributed voting system. Semi-Markov stochastic Petri nets are extensions of GSPNs [3], which can handle arbitrary state-dependent holding-time distributions and which generate an underlying semi-Markov process rather than a Markov process.

The semi-Markov stochastic Petri net of a distributed voting system is shown in Figure 6. Voting agents vote asynchronously, moving from place p_1 to p_2 as they do so. A restricted number of polling units which receive their votes transit t_1 from place p_3 to place p_4. At t_2, the vote is registered with as many central voting units as are currently operational in p_5.

The system is considered to be in a failure mode if either all the polling units have failed and are in p_7 or all the central voting units have failed

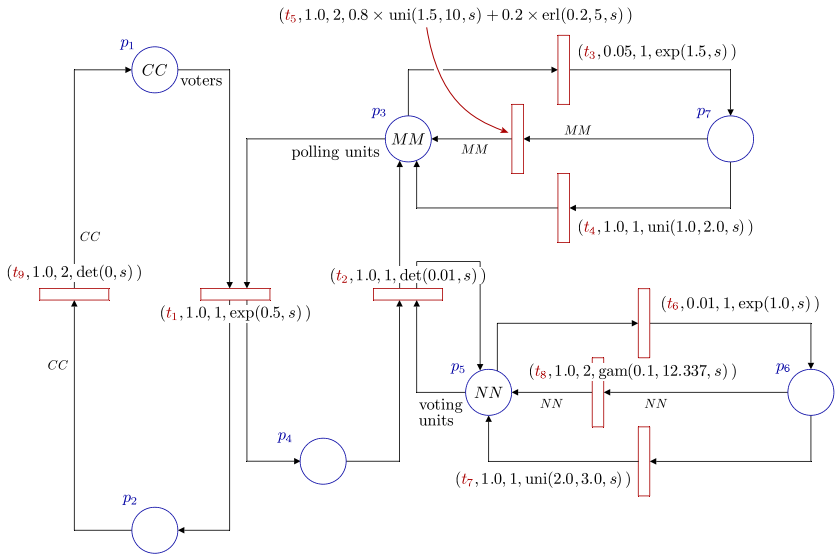

Fig. 6. A semi-Markov stochastic Petri net of a voting system

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

For the voting system described in Figure 6, Table 6.5 shows how the size of the underlying SMP varies according to the configuration of the variables CC, MM, and NN, which are the number of voters, polling units and central voting units, respectively.

Table 3. Different configurations of the voting system as used to present results

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

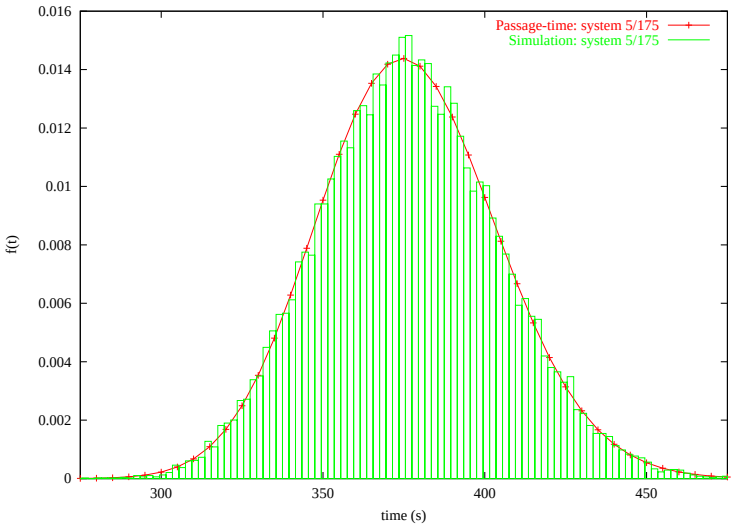

Fig. 7. Analytic and simulated density for the time taken to process 175 voters in system 5 (1.1 million states).

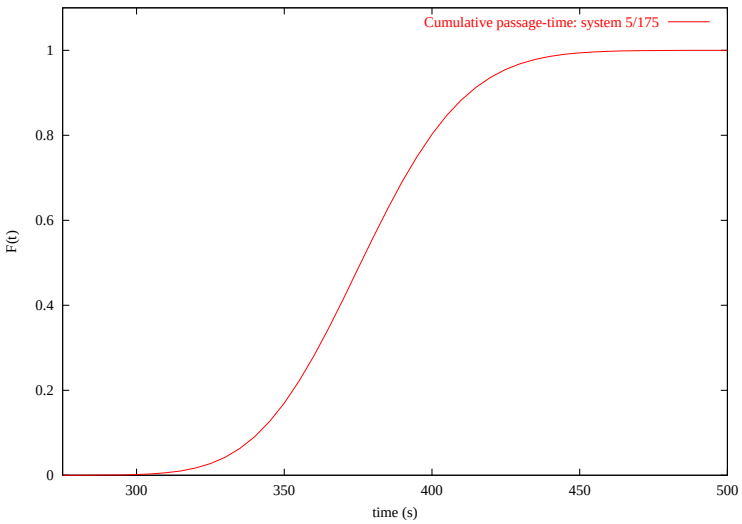

Fig. 8. Cumulative distribution function for the time taken to process 175 voters in system 5 (1.1 million states).

Figure 7 shows the density of the time taken for the passage of 175 voters from place p_1 to p_2 in system 5 as computed by both our (truncated) iterative technique and by simulation. The close agreement provides mutual validation of the analytical method, with its numerical approximation, and the simulation. It is interesting that, qualitatively, the density appears close to Normal. Certainly, the passage time random variable is a (weighted) sum of a large number of independent random variables, but these are, in general, not identically distributed.

Figure 8 shows a cumulative distribution for the same passage as Figure 7. This is easily obtained by inverting the Laplace transform $L_{\vec{ij}}(s)/s$; it allows us to extract response time quantiles, for instance:

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

7. Conclusion

We have seen that finding time delay densities is a hard problem, often with with complex and computationally expensive solutions when they can be solved at all. Consequently, in most practical applications, the performance engineer requires approximate methods. There is no single established methodology for such approximation and most of the techniques used are ad hoc.

The computation of quantiles of sojourn times in the performance engineering of diverse operational systems, such as internet communication, has been recognised for many years and is increasing with present day benchmarks. We have presented a personal perspective on this subject, revealing the increasing difficulties in computing the probability distribution functions of times delays – and hence quantiles – as models become more complex. From the single FCFS M/G/1 queue, through queues with more complex queueing disciplines and negative customers, to tree-like open and closed queueing networks of Markovian queues, it was shown how analytical solutions for response time distributions could be found in the time domain. More sophisticated cases, not considered here, can compute Laplace transforms of response time probability densities, but these require numerical inversion to get distribution functions, using methods such as Euler and Laguerre as discussed, in Section 6.3.

For many years, other problems, perhaps with no particular structure, could only be solved approximately by analytical methods, the approach of direct analysis of the Markov chain being numerically infeasible. However, in recent years, this direct approach has become viable for small-to-medium

sized problems by exploiting parallel computation and the availability of very large storage systems at all levels form cache to disk file stores. We explained how the resulting matrix analytic problems can be structured to take advantage of these technological advances, in particular using hypergraph partitioning of matrices and a fixed storage representation of probability distribution functions (or rather their Laplace-Stieltjes transforms).

Quantiles can indeed now be generated numerically, exactly or nearly exactly, for many problems but even quite moderately sized ones are still intractable. The future will require approximate and asymptotic methods, e.g. for tail probabilities; this is a growing research area.

References

1. J. Abate, G.L. Choudhury, and W. Whitt. On the Laguerre method for numerically inverting Laplace transforms. *INFORMS Journal on Computing*, 8(4):413–427, 1996.
2. J. Abate and W. Whitt. Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, 7(1):36–43, 1995.
3. 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*, 2:93–122, 1984.
4. O.J. Boxma and H. Daduna. Sojourn times in queueing networks. In H. Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*, pages 401–450. North Holland, 1990.
5. 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 *Proc. International Workshop of Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems (PMEO-PDS 2003)*, page 281, Nice, France, April 2003.
6. 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. In *Proc. 4th International Meeting on the Numerical Solution of Markov Chains (NSMC 2003)*, pages 99–120, Chicago, USA, September 2003.
7. J.P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16:527–531, 1973.
8. U.V. Catalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, July 1999.
9. R. Chakka and P.G. Harrison. A Markov modulated multi-server queue with negative customers – the MM CPP/GE/c/L G-queue. *Acta Informatica*, 37(11–12):881–919, 2001.
10. E.G. Coffman Jnr, R.R. Muntz, and H. Trotter. Waiting time distribution for processor-sharing systems. *Journal of the ACM*, 17:123–130, 1970.
11. D.D. Deavours and W.H. Sanders. An efficient disk-based tool for solving

large Markov models. *Performance Evaluation*, 33(1):67–84, June 1998.
12. N.J. Dingle, P.G. Harrison, and W.J. Knottenbelt. HYDRA: HYpergraph-based Distributed Response-time Analyser. In *Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003)*, pages 215–219, Las Vegas, Nevada, USA, June 2003.
13. H. Duduna. Passage times for overtake-free paths in Gordon-Newell networks. *Advances in Applied Probability*, 14:672–686, 1982.
14. E. Gelenbe. Queueing networks with negative and positive customers. *Journal of Applied Probability*, 28:656–663, 1991.
15. E. Gelenbe (ed.). Feature issue on G-networks. *European Journal of Operations Research*, 126, 2000.
16. W. Grassman. Means and variances of time averages in Markovian environments. *European Journal of Operational Research*, 31(1):132–139, 1987.
17. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, Massachussetts, 1994.
18. P.G. Harrison. The distribution of cycle times in tree-like networks of queues. *Computer Journal*, 27(1):27–36, 1984.
19. P.G. Harrison. Laplace transform inversion and passage-time distributions in Markov processes. *Journal of Applied Probability*, 27:74–87, 1990.
20. P.G. Harrison. On non-uniform packet switched delta networks and the hot-spot effect. *IEE Proceedings E*, 138(3):123–130, 1991.
21. P.G. Harrison. The MM CPP/GE/c/L G-queue: sojourn time distribution. *Queueing Systems: Theory and Applications*, 41:271–298, 2002.
22. P.G. Harrison and A.J. Field. Sojourn times in a random queue with and without preemption. *European Journal of Operations Research*, 112:646–653, 1999.
23. P.G. Harrison and W.J. Knottenbelt. Passage time distributions in large Markov chains. In *Proc. ACM SIGMETRICS 2002*, Marina Del Rey, California, June 2002.
24. P.G. Harrison and N.M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. International Computer Science Series. Addison Wesley, 1993.
25. P.G. Harrison and E. Pitel. Sojourn times in single server queues with negative customers. *Journal of Applied Probability*, 30:943–963, 1993.
26. P.G. Harrison and E. Pitel. Response time distributions in tandem G-networks. *Journal of Applied Probability*, 32:224–246, 1995.
27. P.G. Harrison and H. Zatschler. Sojourn time distributions in modulated G-queues with batch processing. In *Proc. 1st Quantitative Evalution of Systems Conference (QEST '04)*, Twente, the Netherlands, September 2004.
28. George Karypis and Vipin Kumar. Multilevel k-way hypergraph paritition-ing. Technical Report #98-036, University of Minnesota, 1998.
29. F.P. Kelly and P.K. Pollett. Sojourn times in closed queueing networks. *Advances in Applied Probability*, 15:638–656, 1983.
30. W.J. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, Cape Town, South Africa,

July 1996.
31. W.J. Knottenbelt. *Parallel Performance Analysis of Large Markov Models*. PhD thesis, Imperial College, London, United Kingdom, February 2000.
32. W.J. Knottenbelt and P.G. Harrison. Distributed disk-based solution techniques for large Markov models. In *Proceedings of the 3rd International Meeting on the Numerical Solution of Markov Chains (NSMC '99)*, pages 58–75, Zaragoza, Spain, September 1999.
33. W.J. Knottenbelt, P.G. Harrison, M.A. Mestern, and P.S. Kritzinger. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation*, 39(1–4):127–148, February 2000.
34. J.D.C. Little. A proof of the queueing formula $L = \lambda * W$. *Operations Research*, 9:383–387, 1961.
35. B. Melamed and M. Yadin. Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes. *Operations Research*, 32(4):926–944, July–August 1984.
36. I. Mitrani. Response time problems in communication networks. *Journal of the Royal Statistical Society B*, 47(3):396–406, 1985.
37. I. Mitrani. *Probabilistic Modelling*. Cambridge University Press, 1998.
38. J.K. Muppala and K.S. Trivedi. Numerical transient analysis of finite Markovian queueing systems. *Queueing and Related Models, Bhat, U.N.; Basawa, I.V. (eds.)*, pages 262–284, 1992.
39. A. Reibman and K.S. Trivedi. Numerical transient analysis of Markov models. *Computers and Operations Research*, 15(1):19–36, 1988.
40. L. Takacs. *Introduction to the theory of queues*. Oxford University Press, 1962.
41. Transaction Processing Performance Council. *TPC benchmark C: Standard specification revision 5.2*. 2003.