Terminating Passage-Time Calculations on Uniformised Markov Chains

Allan Clark* Stephen Gilmore†

Abstract

Uniformisation[1, 2] is a key technique which allows modellers to extract passage-time quantiles/densities which in turn permits the plotting of probability density and cumulative distribution functions. Uniformisation converts a CTMC (Continuous-Time Markov Chain) into a DTMC (Discrete-Time Markov Chain) with equivalent semantics. This can be used to calculate the probability of completing a passage within a given time t by calculating the probability of completing the passage within a number of iterations, n, of the DTMC and then calculating the probability that the nth iteration is performed within time t. However to calculate the passage-time quantiles we desire we must theoretically perform this calculation for values of n from zero to infinity and sum the probabilities. This can be approximated by calculating for values of n from zero to some finite value if we know that larger values of n will yield negligible probabilities and hence add nothing significant to the summation. This paper discusses two important conditions which ensure that the approximation is appropriate while also reducing the amount of negligible values calculated.

1 Introduction

Passage-time quantiles are often desirable measurements to be made from a performance model. In the case of a passage-time measurement we measure from a set of source states to a set of target states. A passage-time quantile is simply a point taken along the cumulative distribution function (cdf) of the passage in question. The cumulative distribution function maps time (usually along the x-axis) against the probability of completing the passage at or before that time. This allows the modeller to answer such questions as: "What is the probability that a request is responded to within 4 seconds?" It is also possible to plot the probability density function (pdf) where the cdf is the integral of the pdf. The pdf then maps time against the probability density of completing the passage at exactly that time.

This paper describes the calculation of passage-time quantiles/densities from a CTMC based model. The technique used is known as *uniformisation* – though it is sometimes called *randomisation*. This paper is chiefly concerned with the steps which follow the actual uniformisation of a CTMC. This is the process

*LFCS, University of Edinburgh, a.d.clark@ed.ac.uk
†LFCS, University of Edinburgh, stg@inf.ed.ac.uk

of extracting from the uniformised Markov chain the probabilities which we desire. For this reason our first example concerns a CTMC which is already in a uniformised state due to all of the rates involved being equal.

The paper is organised as follows; in Section 2 we give an overview of the whole process of uniformisation. Section 3 then introduces an example uniform CTMC which is used to analyse the process of extracting our quantiles from the uniformised Markov chain. Section 4 compares with existing approaches, Section 5 details some finer implementation points and finally in Section 6 we conclude.

The major contribution of this paper is the identification of two properties which allow the accurate halting of the calculation of quantiles. However this paper is sufficiently detailed to serve as an introduction to the key technique of uniformisation in general.

2 Uniformisation

This section details the steps used to derive the cdf (and/or pdf) from a model represented as a CTMC. We first detail the prerequisites:

- The CTMC may be represented by the generator matrix Q. The generator matrix is an n x n matrix where n is the number of states in the Markov chain. Each row corresponds to one state and the value in one cell of a row corresponds to the rate at which the Markov chain may transition from the state given by the row number to the state given by the column number. The diagonal values are given by subtracting from zero the sum of the other values in the row. Hence if we write $r(i, j)$ to mean the rate at which the Markov chain may transition from state i to j then the generator matrix Q is written as:

  $$Q_{i,j} = \begin{cases} r(i,j) & : \quad i \neq j \\ 0 - (\sum_{k=0}^{n-1} r(i,k)) & : \quad otherwise \end{cases}$$

  This requires that for any state i, the rate $r(i, i)$ is zero – this condition is usually stated by insisting that the model contains no self-loops.

- The generator matrix may be solved to obtain the steady-state probability distribution π where π_i is the long-term probability of being in state i. This requires that the model be deadlock-free.

- The set of source states $\mathcal{S}$ and the set of target states $\mathcal{T}$. The probability at time t that we compute is the probability of moving from any of the states in $\mathcal{S}$ to any of the states in $\mathcal{T}$ within time t.

The steps in the computation of the pdf and the cdf of a particular passage within a CTMC are summarised as follows:

- Uniformise the generator matrix to obtain the new matrix P by:

  $$P = Q/q + I$$

  where Q is the generator matrix, I is the identity matrix and q is a rate value which is chosen to be greater than the magnitude of all of the rates within the generator matrix including the values along the diagonal.

Therefore we have $q > \max_{ij} |Q_{ij}|$ which can be reduced to $q > \max_i |Q_{ii}|$ since the magnitude of the diaganol values in each row are the sums of the other values in the row which cannot be negative. Since q is of greater magnitude than any of the (negative) diagonal values dividing by q returns a negative number $x : -1 < x < 0$. This means that adding the identity matrix ensures that all rate values are positive.

- Add to this uniformised matrix P an absorbing state. This state has no out-going edges to any state other than itself which it loops to with probability 1.

- Modify all target states (states in $\mathcal{T}$) to transition with probability one to the absorbing state. Call this new matrix P'. The reason for our absorbing state is to ensure that we compute the probability of the first passage and not subsequent completions of the passage. That is; if we are in state $i \in \mathcal{T}$ at time t then we know we are completing the passage at time t and it is not the case that we completed the passage at some earlier time and remained in or returned to state i.

- Compute the probability distribution after n hops of the uniformised Markov chain; given by $\pi^{(n)}$ where $\pi^{(n+1)} = \pi^{(n)}P'$ And $\pi^{(0)}$ is computed using the steady-state probabilities in the embedded Markov chain of the source states by:

  $$\pi_k^{(0)} = \begin{cases} 0 & : \quad k \notin \mathcal{S} \\ \pi_k/\pi_{\mathcal{S}} & : \quad k \in \mathcal{S} \end{cases}$$

  where π_k is the steady-state probability in the Embedded Markov Chain of being in state k and $\pi_{\mathcal{S}}$ is the steady-state probability in the EMC of being in any one of the source states, that is $\sum_{k \in \mathcal{S}} \pi_k$. Where there is exactly one source state then the steady-state probability distribution need not be calculated and $\pi^{(0)}$ is given by:

  $$\pi_k^{(0)} = \begin{cases} 0 & : \quad k \notin \mathcal{S} \\ 1 & : \quad k \in \mathcal{S} \end{cases}$$

  since for the one source state j, $\pi_j = \pi_{\mathcal{S}}$.

- For each time t compute: $\sum_{n=0}^{n=\infty} Er_t^{(n)} \pi_{\mathcal{T}}^{(n)}$ where $Er_t^{(n)}$ is the probability that the nth hop will be performed at or before time t and $\pi_{\mathcal{T}}^{(n)}$ is the probability of being in any of the target states after exactly n hops of the uniformised matrix, P'.

In the final step above we have computed the cdf of the passage by multiplying the probability of being in a target state after exactly n hops by the probability of performing n within the time t. This probability is given by: $\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right)$. For the pdf we substitute this for the probability of performing n hops at exactly time t. This is given by: $\frac{q^n t^{n-1} e^{-qt}}{(n-1)!}$.

For completeness we provide the full formulae for computing the cdf and pdf of the passage respectively given by:

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

and

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

Transient Measures A transient measure seeks to obtain information about a model from the initial state of the model. The intention is to answer such questions as "What is the expected time before the server first breaks?" Because we are asking about the model in the short term and not the long term the steady-state distribution need not be calculated. The measurement reduces to a passage-time measurement in which there is only one source-state (namely the initial system configuration). A transient measure is often of use if the model is not free from deadlock – asking the probability that the system is deadlocked after a given time is a common transient measure. Depending on the particular query the transient measure may or may not require the addition of the absorbing state. In this paper we focus on passage-time measurements.

3 Snakes And Ladders

In this section we detail an example Markovian analysis of the simple board game "Snakes and Ladders". Before we proceed with the analysis a brief revision of the rules. Players start by placing a counter on the start square and take it in turns to roll a dice. When a player roles the dice they move their counter forward the number of squares equal to the number they have rolled on the dice. If a player lands directly on a square on which rests the bottom of a ladder they can immediately move their counter to the square at the top of the ladder. Similarly if a player lands directly on a square on which rests the head of a snake then that player must move their counter down to the square at the tail of the snake.

Analysing this game using a DTMC to assess the percentage chance of winning any game within a number of turns N has already been done. If we wish to analyse how long in wall-clock time it will take to complete a game then we must combine the information about how likely it is to win the game after N turns together with the probability of performing N turns within a given time.

So our situation is exactly as in the case that we had started with a CTMC and used uniformisation to obtain a uniform CTMC except that our CTMC was already neatly uniformised to begin with.

We will use a simplified version of the game with only sixteen squares and a dice with only three sides, a player can only roll a 1, a 2 or a 3. We further simplify our task by analysing how long we can expect one player playing by themselves to complete the game.

The game board looks like the one drawn in Figure 1.

The DTMC representing this can be shown in Table 1, note that there are only thirteen game states as opposed to sixteen, this is because the state representing square 5 is equivalent to the state representing square 12 since when a player lands on square 5 their token is automatically moved to square 12. In particular notice that the states representing squares $2, 3$ and 4 each have an out-going edge straight to square 12 rather than square 5. Should a player on square 2 roll a 3 their token will end up on square 12, no token can therefore rest on square 5 so we omit it from the state space. Similarly for the two squares, 13 and 14, on which there is a snake. There are fourteen states in total because the absoring state which we will require is already shown. In the table transitions representing a move to a ladder or snake square and the resulting jump have their rates written in green and red respectively.

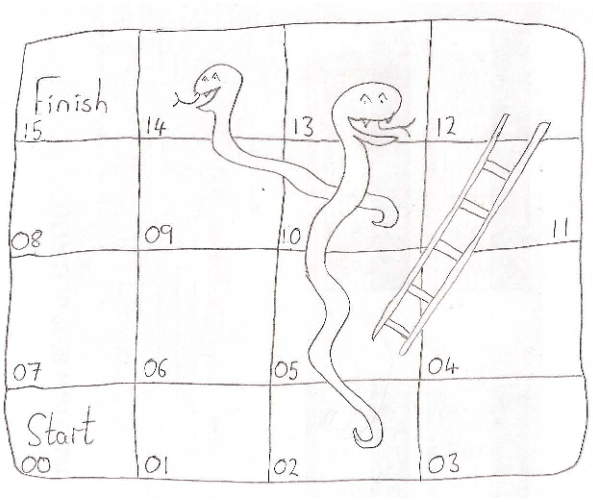

Figure 1: The simplified Snakes and Ladders board game.

	0	1	2	3	4	6	7	8	9	10	11	12	15	Abs
0		$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$										
1			$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$									
2				$\frac{1}{3}$	$\frac{1}{3}$							$\frac{1}{3}$		
3					$\frac{1}{3}$	$\frac{1}{3}$						$\frac{1}{3}$		
4						$\frac{1}{3}$	$\frac{1}{3}$					$\frac{1}{3}$		
6							$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$					
7								$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$				
8									$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$			
9										$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$		
10			$\frac{1}{3}$								$\frac{1}{3}$	$\frac{1}{3}$		
11			$\frac{1}{3}$							$\frac{1}{3}$		$\frac{1}{3}$		
12			$\frac{1}{3}$							$\frac{1}{3}$			$\frac{1}{3}$	
15														1
Abs														1

Table 1: The DTMC of a simple snakes and ladders game

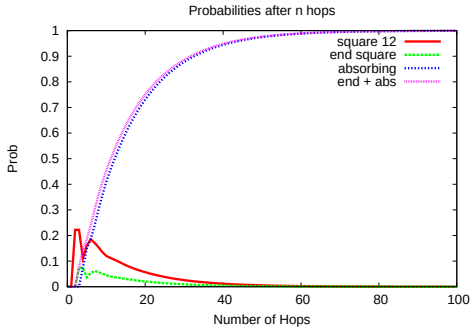

Figure 2: The probability of completing the game after N hops

We can measure from the set of source states: $\{0\}$ to the set of target states: $\{15\}$, note that this means we do not require a steady-state distribution since there is exactly one source state. The graph in Figure 2 shows the probability of being in particular states after N hops. The probability flows from the other states through the target state and into the absorbing state which is why the probability of being in the absorbing state increases as N increases and will equal one in the limit. The line for the target state depicts the probability density function (against number of hops rather than time). A fourth line is drawn which adds the probability of being in either the end state or the absorbing state. This depicts the cumulative distribution function of the game as it adds the probability completing the passage at exactly N hops (the probability of being in the target state) to the probability of completing the game before N hops (the probability of being in the absorbing state).

Now that we know the probability of completing the game in exactly N hops, we may use this information to calculate the probability of completing the game at or by a given time. Since in our example each hop represents one turn or move of the game then we need only know the rate or average duration of a turn in the game. If we assume that each turn lasts about six seconds then the rate at which they occur is ten-per-minute. The graph (Fig. 3, left) shows the probability of performing exactly N hops at or by time t for various values of t. The graph (Fig. 3, right) shows the probability of performing N hops at or by time t seconds for various values of N.

3.1 Producing our final cdf

We can now combine the information in the graphs (Fig 2) and (Fig. 3, right) to produce the cumulative distribution function for the time it will take one player to complete the simplified snakes and ladders game. We know the probability of completing the passage in exactly N hops, $\pi_{\mathcal{T}}^{(n)}$, for all values of N. In addition we know the probability of performing N hops in a given amount of time t, $Er_t^{(n)}$. If we multiply these two values for a given N and a given t this gives us the probability of completing the game in the given t using exactly N hops – this value we designate as $P_n(t)$. Therefore to compute the general probability of completing the game within the given amount of time t we need to sum up

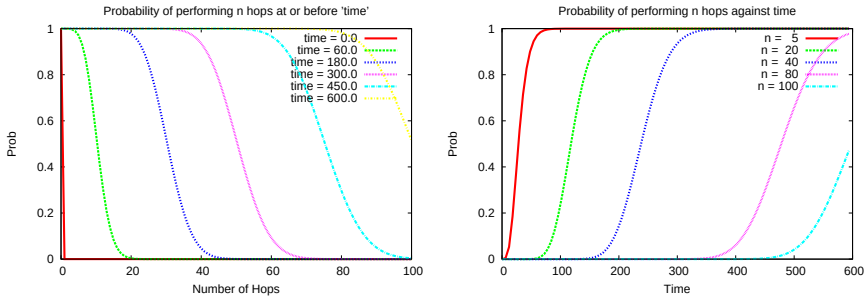

Figure 3: Graphs showing the probability of performing a number of hops by a given time.

Probability of:	value	name
completing the passage in exactly N hops	$\sum_{k \in \vec{j}} \pi_k^{(n)}$	$\pi_{\mathcal{T}}^{(n)}$
performing N hops within time t	$\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right)$	$Er_t^{(n)}$
completing the passage in N hops by time t	$\pi_{\mathcal{T}}^{(n)} Er_t^{(n)}$	$P_n(t)$
completing the passage by time t	$\sum_{N=0}^{N=\infty} P_n(t)$	cdf

Table 2: Relationships between the probability values

$P_n(t)$ for all values of N from zero to infinity. The relationships between these probabilities are shown in Table 2.

Clearly, summing all of these probability values from zero to infinity is impossible for a computer to do. However there will be some value X for which all values $N_X > X$ the probability of completing the passage within the given time in exactly N_X hops is negligible. Hence at this point we may stop computing probability values. The main contribution of this paper is determining the two conditions which suffice to find the value X.

Previously one method was to compute the probabilities for successive values of N and whenever the probability $(P_n(t))$ was sufficiently low we assume that subsequent values will also be sufficiently low. This method has problems when the passage we must complete has separate paths which vary greatly in their length of hops. In this instance it is possible for the probability to drop below the threshold value but to later climb above it. In this case the given method would stop calculating the probability values before they have a chance to rise above the threshold once more.

Our method is to monitor the probability of being in the absorbing state after N hops – we designate this value $Abs^{(n)}$. When this value climbs to within a suitable threshold of 1 then there is no probability left to flow through the target states. Hence the probability of being in a target state for all values of N greater than the current value must be below the threshold value, since this probability is multiplied by the probabilty of performing N hops within time t we know that all subsequent probabilities will be below the threshold.

This method performs well, however, for small values of t we find that we compute more hop-values than are required. This is because for small values of

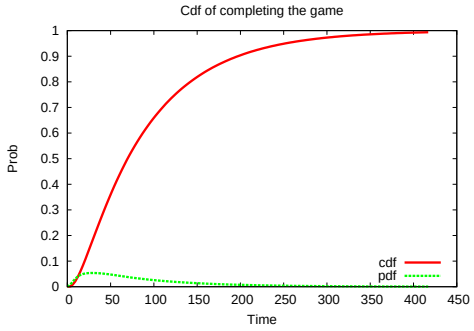

Figure 4: The probability of completing the game after N hops

t it is unlikely that we are able to perform a large number of hops. However if the passage is long then it may be that the probability of being in the absorbing state does not climb to within the threshold of one until N is large – whereby 'large' we mean "larger than the number of hops we could hope to perform within the time t". Therefore we also monitor the value of $P_n(t)$ – the probability of performing N hops within time t – whenever this value falls below the threshold, we know that any subsequent values of N will yield negligible probability at time t (since $P_n(t)$ is involved in the product to find the probability) and hence we have determined a suitable value of X.

Our algorithm may be summed up by a recursive function as:

$$cdf(n,t) = \begin{cases} 0 & : \quad Abs^{(n)} > (1 - threshold) \\ 0 & : \quad Er_t^{(n)} < threshold \\ \left(\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}\right) + (cdf(n+1,t)) & : \quad otherwise \end{cases}$$

and similarly for the pdf function:

$$pdf(n,t) = \begin{cases} 0 & : \quad Abs^{(n)} > (1 - threshold) \\ 0 & : \quad Er_t^{(n)} < threshold \\ \left(\pi_{\mathcal{T}}^{(n)} * Erp_t^{(n)}\right) + (pdf(n+1,t)) & : \quad otherwise \end{cases}$$

Where $Erp_t^{(n)}$ is the probability of performing the Nth hop at exactly time t and is given by: $\frac{q^n t^{n-1} e^{-qt}}{(n-1)!}$. Since there is a lot of shared computation our implementation computes both the cdf and the pdf of the passage together.

Finally then we may draw our graphs of the cumulative distribution and probability density functions of our snakes and ladders game. These are depicted in Figure 4.

4 Comparison with existing techniques

The naïve approach which we briefly illustrated in section 3.1 is to compute values for successive values of N until such values drop below a threshold. This method may be summed up by:

$$cdf(n,t) = \begin{cases} 0 & : \quad \left(\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}\right) < threshold \\ \left(\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}\right) + (cdf(n+1,t)) & : \quad otherwise \end{cases}$$

As we mentioned above this algorithm suffers from a problem if the input passage has multiple paths to completion of varying lengths. In this case the value at some N may drop below the threshold but may later rise above the threshold again. The simple solution would stop after the first time it drops below the threshold.

As an improvement on this technique the Markovian response-time analyser Hydra[3, 4, 5, 6] monitors the value of the erlang distribution with a q rate parameter and N hop parameter. The Hydra solution can therefore be summarised by:

$$cdf(n,t) = \begin{cases} 0 & : \quad Er_t^{(n)} < threshold \\ \left(\pi_{\mathcal{I}}^{(n)} * Er_t^{(n)}\right) + (cdf(n+1,t)) & : \quad otherwise \end{cases}$$

Therefore this solution will compute the same values as our solution in all cases because our solution contains the same condition. However our solution is a further refinement which allows us to avoid needless computation for some values of N. In particular where the t-range — that is the times for which we should compute the passage-time quantiles — specified is too large. Suppose the user has specified a t-range of $1-1000$ but the passage has a probability very close to one of completing by time 500. Because there is a large probability of completing the passage by time 500 this means that there is a large probability of completing the passage within a number of hops X and that X hops are very likely to be performed within 500 time units. This means that for time values over 500 there will be a possibility to perform more than X hops and the Hydra solution will continue to compute probabilities for these hop values. However our solution would recognise that such values cannot add anything to the cdf because you are very like to have completed the passage before X hops. In the case of the cdf this could be mitigated by incorporating the naïve solution but this is not as effective for computing the pdf.

Our solution has a further, related, advantage; the user need not specify the upper-bound on the t-range at all. The user need only give the start of the t-range and the steps in which we wish to increase the value of t. This is because using our technique we can calculate the value X at which performing more than X number of hops will not significantly add to the probability of completing the passage (because there is a probabilty within the threshold of one of being in the absorbing state by X hops). We can then use this to work out the upper-bound on the t-range by calculating the value of t such that performing X hops within time t is significantly likely. In order that the user need not specify a t-range at all we default to a starting time of zero and a time-step of the calculated stop-time divided by one hundred. The user may then override any of; the start-time, the stop-time, the time-increments and the number to divide the t-range by in order to obtain the time-increments.

5 Implementation

The techniques described in this paper have been fully implemented in the International PEPA Compiler (ipc) based on the ipclib[7]. This is a compiler for the Performance Evaluation Process Algebra (PEPA)[8], is open source software and may be downloaded from: `http://www.dcs.ed.ac.uk/pepa/tools/ipc/`

5.1 Technical Points

This paper has shown how to compute passage-time quantiles from continuous time Markov chains. However we have left the actual numerical computation as a given, though this is non-trivial. For the cumulative distribution function we must compute:

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

Notice in particular that we must compute $k!$ for what may be large values of k. In addition we must compute qt^k, also for potentially large values of k. The large values here are in the order of the number of hops, if we have large differences between the rates this value may be quite high — a value in the order of thousands is not uncommon (in [9] this number is said to be of the order of qt). Hence we can expect to encounter a problem with overflow. Even if some arbitrary precision library is used (at a performance cost) computing the cdf in this way is inefficient. Our first observation is that:

$\frac{(qt)^k}{k!}$ is equal to: $\prod_{i=1}^{i=k} \frac{qt}{i}$

which allows us to avoid the computation of the large power and factorial values.

Now for each value of N we must compute $\sum_{k=0}^{N} \frac{(qt)^k}{k!}$, we need not compute each term separately we can instead compute the infinite list of values by the recursive function:

$$\begin{aligned} sumvalues(n, current) &= current : rest \\ \text{where } rest &= sumvalues\left(n+1, current + \sum_{k=0}^{N} \frac{(qt)^k}{k!}\right) \end{aligned}$$

Because our implementation is in the lazy programming language Haskell we need not worry about the computation of an infinite list since we will only ever examine a finite number of elements from it. For a strict language this laziness can be easily simulated. We now observe that even this computation does a large amount of re-computation. Namely the successive values of $\sum_{k=0}^{N} \frac{(qt)^k}{k!}$ recompute all previous values. However we can use the same trick:

$$\begin{aligned} prodvalues(k, current) &= current : rest \\ \text{where } rest &= prodvalues\left(k+1, current \times \frac{qt}{k}\right) \end{aligned}$$

This means that we can now update our *sumvalues* function to take advantage of this. It now becomes a list transformation function which takes in the list of product values computed by the above *prodvalues* function.

$$\begin{aligned} sumvalues(current, (n, p) : rest) &= (n, current) : restsum \\ \text{where } restsum &= sumvalues(current + p, rest) \end{aligned}$$

We can also factor out the code to calculate the probability of being in the absorbing state and/or a target state after exactly N hops. Since otherwise we will recompute these values for each time value we desire. Once we have factored out all the common computation we have a set of infinite lists which map N from $N = 0$ to $N = \infty$ to values used in the computation of the cdf and pdf. We need only operate on these lists for the values of t.

5.2 Computing hops

We must compute the probability of completing the passage in a given number of hops. Recall that the probability of completing the passage in exactly N hops is given by: $\sum_{k \in \mathcal{T}} \pi_k^N$

Further recall that each hop is computed via the previous hop as:

$\pi^{(n+1)} = \pi^{(n)} P'$

Therefore we are required to perform successive matrix multiplications. If we have a large matrix P' these matrix multiplications may be expensive. However we note that P' is a modified version of the matrix P which is a uniformised version of the original generator matrix Q. We have modified P by adding an absorbing state and each state in the target set $\mathcal{T}$ has been mutated to target only the absorbing state. This means that, even in the case that all states in the matrix Q were reachable there may be a set of states which are now unreachable, call this set $\mathcal{U}$. In fact this set is likely to be non-empty. It represents all the states in the original which are not on any path from $\mathcal{S}$ to $\mathcal{T}$ but are on some path from $\mathcal{T}$ to $\mathcal{S}$.

Consider the model of a system containing 20 clients and 2 servers. Each server may accept *requests* and subsequently make a *response*. The servers therefore have two states: *Available* and *Busy*. Each client synchronises with one of the two servers over a *request* and then waits for a *response*. Between subsequent communications with a server each client must do two pieces of work. The client therefore has three states which it cycles through; $Working$, $Waiting$ and $Using$. The $Using$ state corresponds to the using of the data returned by the server and the $Working$ state corresponds to the generation of a new request to the server.

If we wish to measure the response-time of this model we must measure the response-time as observed by a single client, suppose we choose the first client named $Client_0$. The set of source states is the set of states in which the first client is in the $Waiting$ state and is a target of a transition in which the source state of the transition has the first client in the $Working$ state. Conversely the set of target states is the set of states in which the first client is in the $Using$ state and is a target of some transition from a state in which the first client is in the state $Waiting$.

For this measurement all states in which the first client is in the $Waiting$ state are along some path from $\mathcal{S}$ to $\mathcal{T}$ however all states in which the first client is the in $Working$ state are not in $\mathcal{T}$ (since the client must go through the $Using$ state) or in $\mathcal{S}$ or on some path between $\mathcal{S}$ and $\mathcal{T}$. These states are all in the unreachable set $\mathcal{U}$. In this particular case this corresponds to approximately half of the entire state space of the model. Where there are more client states not within the passage this ratio can increase such that the size of the set $\mathcal{U}$ is much larger than the set of states not in $\mathcal{U}$.

Because we know that for any n and $i \in \mathcal{U}$:

$\pi_i^{(n)} = 0$

we can avoid a lot of calculation by transforming the matrix P' into a smaller matrix removing the unreachable states. Since we perform many matrix multiplications using this matrix to obtain the hops, this is a potentially very large saving.

6 Conclusions

In this paper we have given a detailed account of the calculation of passage-time quantiles and densities from continuous-time Markov chains. Although we have shown how to obtain the uniformised matrix from the original generator matrix of the CTMC, we have focussed on the calculations that occur once the matrix has already been uniformised. To this extent our main contribution has been

the identification of two important properties which allow the otherwise infinite calculation to terminate. The two properties have the desired feature that we are conservative — meaning that we never terminate too early producing an erroneous answer. However the combination of the two provides early detection to avoid some needless calculations. In addition we feel that our paper is a good introduction to the topic of uniformisation in general.

Acknowledgements: The authors are supported by the EU FET-IST Global Computing 2 project SENSORIA ("Software Engineering for Service-Oriented Overlay Computers" (IST-3-016004-IP-09)) and the EPSRC PerformDB project (EP/D054087/1). The ipc/Hydra tool chain has been developed in co-operation with Jeremy Bradley, Will Knottenbelt and Nick Dingle of Imperial College, London.

References

[1] Grassmann, W.: Transient solutions in Markovian queueing systems. Computers and Operations Research **4** (1977) 47–53

[2] Gross, D., Miller, D.: The randomization technique as a modelling tool and solution procedure for transient Markov processes. Operations Research **32** (1984) 343–361

[3] Dingle, N.J., Knottenbelt, W.J., Harrison, P.G.: HYDRA: HYpergraph-based Distributed Response-time Analyser . In Arabnia, H.R., Man, Y., eds.: PDPTA'03, Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications. Volume 1., Las Vegas, NV (2003) 215–219

[4] Knottenbelt, W.J.: Generalised Markovian Analysis of Timed Transition Systems. Master's thesis, Department of Computer Science, University of Cape Town (1996)

[5] Dingle, N.J.: Parallel Computation of Response Time Densities and Quantiles in Large Markov and Semi-Markov Models. PhD thesis, Department of Computing, Imperial College London. University of London. (2004)

[6] Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Extracting passage times from PEPA models with the HYDRA tool: A case study. In Jarvis, S., ed.: Proceedings of the Nineteenth annual UK Performance Engineering Workshop, University of Warwick (2003) 79–90

[7] Clark, A.: The ipclib PEPA Library. In Harchol-Balter, M., Kwiatkowska, M., Telek, M., eds.: Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST), IEEE (2007) 55–56

[8] Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)

[9] Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model checking continuous-time markov chains by transient analysis. In: CAV '00: Proceedings of the 12th International Conference on Computer Aided Verification, London, UK, Springer-Verlag (2000) 358–372