From *Communications in Statistics: Stochastic Models, vol. 10, no. 3, August 1994, pp. 619-648.*

ADAPTIVE UNIFORMIZATION

Aad P. A. VAN MOORSEL† and William H. SANDERS‡

†University of Twente
Tele-Informatics and Open Systems
P.O. Box 217, 7500 AE Enschede, The Netherlands
moorsel@cs.utwente.nl

‡University of Arizona
Department of Electrical and Computer Engineering
Tucson AZ 85721
whs@ece.arizona.edu

ABSTRACT

Uniformization has been shown to be, in many cases, a good method to compute transient state probabilities of a continuous-time Markov chain. However, two issues limit its use: uniformization can be computationally very intensive, for instance, on stiff models, and uniformization cannot be used for all model classes, e.g., models with not uniformly bounded transition rates. In this paper we introduce *adaptive uniformization*, a variation on standard uniformization, which can overcome these problems for some models. Adaptive uniformization differs from standard uniformization in that it uses a uniformization rate that *adapts* depending on the set of states that the process can be in after a particular number of jumps. Doing this can sometimes significantly reduce the computational cost required to obtain a solution. A formal definition of adaptive uniformization is first given, along with a proof that adaptive uniformization yields correct results. Characteristics of models that can facilitate solution and alternative methods for computing the required "jump probabilities" are then discussed. Finally, the computational cost of adaptive uniformization (relative to standard uniformization) is illustrated, through its application to an extended machine-repairman model.

Keywords: Markov processes, Uniformization, Randomization, Jensen's Method, Transient solution.

1 Introduction

We discuss the problem of determining the transient state probability distribution at some instant of time $t \geq 0$ of a time-homogeneous continuous-time Markov chain (CTMC) $Y = \{Y(s); s \geq 0\}$. Let Y be defined on the countable, possibly infinite state space S, where without loss of generality we assume $S = \{0, 1, 2, ...\}$. Let the infinitesimal generator matrix of Y be $\mathbf{Q} = (q(i,j))$, $i, j \in S$, and define $q_i = \Leftrightarrow q(i,i)$, for $i \in S$. The initial distribution over the state space S is denoted as the row vector $\underline{\pi}(0) = (\pi_0(0), \pi_1(0), ...)$. We are interested in obtaining the transient state probability vector $\underline{\pi}(t)$ with $\pi_i(t) = \Pr\{Y(t) = i\}$, for any $i \in S$.

It is well known from Kolmogorov's differential equations [2] that the transient state probability vector $\underline{\pi}(t)$ obeys

$$\frac{d\underline{\pi}(t)}{dt} = \underline{\pi}(t)\mathbf{Q}, \quad \text{given } \underline{\pi}(0). \tag{1}$$

The solution of this system of differential equations is $\underline{\pi}(t) = \underline{\pi}(0)e^{\mathbf{Q}t}$. Unfortunately, computing the "matrix exponential" $e^{\mathbf{Q}t} = \sum_{i=0}^{\infty}(\mathbf{Q}t)^i/i!$ often gives severe round-off errors because of the negative and possibly large elements in $\mathbf{Q}$ [4]. Other numerical solution procedures exist, for a survey see [18] and [19], but most of them are not fully satisfactory. The technique that is often considered to be the method of choice for solving transient state probability distributions of CTMC's is *uniformization* [5, 18]. Uniformization is based on translating the Taylor series of $e^{\mathbf{Q}t}$ into a Taylor series with a probability matrix (with positive and bounded elements). Uniformization exploits the probabilistic nature of the generator matrix $\mathbf{Q}$, which suggests that it might outperform other, more generic, methods [4, 9]. The applicability of the method for other transient performance and performability measures has been shown extensively [6, 12, 15, 17, 22].

For later use, we now briefly explain uniformization, which, in the context of this paper we call *standard uniformization* (SU). Let $\lambda \geq \sup_{i \in S}\{q_i\}$ be the so-called *uniformization rate*. Then we can define the discrete-time Markov chain (DTMC) $X = \{X_j, j = 0, 1, ...\}$, on the state space S, with transition matrix $\mathbf{P} = (p(i,j))$, $i, j \in S$, defined by

$$\mathbf{P} = \mathbf{I} + \frac{1}{\lambda}\mathbf{Q}. \tag{2}$$

SU then obtains $\underline{\pi}(t)$ via

$$\underline{\pi}(t) = \underline{\pi}(0) \sum_{n=0}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\mathbf{P}^n, \tag{3}$$

which can be rewritten as

$$\underline{\pi}(t) = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \underline{\pi}_n, \tag{4}$$

where $\underline{\pi}_n$, being the state probability distribution vector after n jumps of the DTMC X, is deduced recursively as

$$\underline{\pi}_0 = \underline{\pi}(0) \text{ and } \underline{\pi}_n = \underline{\pi}_{n-1}\mathbf{P}, n = 1, 2, \ldots \tag{5}$$

Of course, in a practical application of SU the infinite sum in Equation (4) must be truncated after some finite number of steps, say $N_s(\epsilon, t)$. We truncate when

$$\sum_{n=0}^{N_s(\epsilon,t)} e^{-\lambda t} \frac{(\lambda t)^n}{n!} = 1 \Leftrightarrow \sum_{n=N_s(\epsilon,t)+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \geq 1 \Leftrightarrow \epsilon. \tag{6}$$

In this way $N_s(\epsilon, t)$ is chosen such that the resulting truncation error is less than a predefined value $\epsilon > 0$, i.e., with $||x||_\infty = \max_i |x_i|$ being the maximum norm:

$$\|\underline{\pi}(t) \Leftrightarrow \sum_{n=0}^{N_s(\epsilon,t)} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \underline{\pi}_n\|_\infty = \| \sum_{n=N_s(\epsilon,t)+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \underline{\pi}_n\|_\infty \leq$$

$$\sum_{n=N_s(\epsilon,t)+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \leq \epsilon, \tag{7}$$

as all elements of $\underline{\pi}_n$ take values in $[0, 1]$. Consequently, by taking $N_s(\epsilon, t)$ large enough, SU can solve for the transient state probabilities *with any desired accuracy* $\epsilon > 0$. Note that, in practice, truncating the summation on the left, as well as the right, may result in a computational advantage. Even when this is done, however, one must carry out the matrix-vector multiplications given in (5), since once the left truncation point is reached, $\underline{\pi}_n$'s are necessary for higher values of n.

SU forces jumps to occur at moments generated according to a Poisson process with rate λ, independent of the current state the DTMC X is in (which explains the term uniformization: independent of the state, there is a single, uniform, transition rate). The conditional probabilities of jumping to state j when in state i are deduced from the DTMC X, and are given by $p(i, j) = q(i, j)/\lambda$, when $i \neq j$, and by $p(i, i) = 1 \Leftrightarrow q_i/\lambda$, $i, j \in S$. Note that self loops are often created in this construction, since $p(i, i)$ can be greater than 0. The value $\lambda \geq q_i$ ensures that the matrix $\mathbf{P}$ is a probability matrix.

SU has been used to solve a variety of interesting models in performance and dependability evaluation, but some important models cannot be solved by SU, either because it is too slow, or because it is not applicable. For

example, stiff models are computationally very intensive to solve using SU. For these systems, the uniformization rate λ is large relative to the time point of interest, and consequently the number of iterations $N_s(\epsilon, t)$ necessary to reach the desired accuracy tends to be very large. Furthermore, models with infinite state spaces and not uniformly bounded transition rates cannot be solved with SU, since a uniformization rate cannot be identified.

In this paper, we introduce "adaptive uniformization" (AU), a variation of standard uniformization. Adaptive uniformization differs from SU in that it uses a uniformization rate that *adapts* depending on the set of states that the process can be in after a particular number of jumps. Doing this can often significantly reduce the computational cost to obtain a solution. When AU is used, computation of the probability of the number of jumps that may occur in some time t requires solution of a general birth process (which we call the "AU-jump process") rather than a Poisson process. Note that the term adaptive uniformization has also been applied (independently) to a related idea, in the context of distributed simulation [16].

AU has the potential to avoid, to some extent, the problems associated with SU. In particular, AU can lower the numerical costs of solving stiff systems, if states with high exit rates cannot be reached until several state transitions have occurred. Furthermore, as shown in [14], it can be used to solve CTMC's with infinite state space and not uniformly bounded transition rates. While AU does not always outperform SU, we believe that in many cases it is superior and, in these cases, should be used.

Other modifications of standard uniformization have been introduced to solve the above mentioned problems. Specifically, Grassmann [5] mentions dynamic uniformization as a method to solve infinite state space models, but as argued in [14], this approach will only work when the transition rates are uniformly bounded. Methods have also been proposed to deal with stiff systems using uniformization, but these are also limited to specific model classes [18, 24]. For example, to solve for stiff models that reach steady-state fast, one can use an approach taken by Muppala and Trivedi [15]. Dunkel *et al.* [1], however, show that for so-called nearly completely decomposable systems steady-state detection does not work well.

The remainder of this paper is organized as follows. Section 2 formally introduces adaptive uniformization and provides a proof of its correctness. Section 3 discusses the nature of the stochastic processes employed when performing AU. It illustrates, via two small examples, the effect different characteristics of the processes can have on the AU procedure. Section 4 discusses issues related to the computation of the state occupancy probabilities of the AU-jump process. Three different methods for computing these probabilities are discussed, differing in their applicability, stability, and computational complexity. Finally, Section 5 illustrates the use of AU on a machine-repairman

model with delayed repair. In the context of this example, we compare the computational costs of AU and SU and suggest when AU is superior to SU.

2 Adaptive Uniformization

2.1 Formal Definition

Adaptive uniformization is based on the idea that it is not necessary to use a single uniformization rate, as is done in standard uniformization, but that one can *adapt* the uniformization rate depending on the states the DTMC $X = \{X_j \mid j = 1, 2, \ldots\}$ can be in at some epoch. To construct the DTMC, we define the set of *active states* at epoch n, $n = 0, 1, 2, ...$, of the DTMC X as the set $A_n \subseteq S$ with

$$A_n = \{i \in S \mid \pi_n(i) > 0\}. \tag{8}$$

Then, for $n = 0, 1, 2, ...$, let $\lambda_n = \sup\{q_i \mid i \in A_n\}$ be the *adapted uniformization rate* and $\mathbf{Q}_n = (q_n(i,j)), i, j \in S$, the *adapted infinitesimal generator* at step n, such that $q_n(i,j) = q(i,j)$ when $i \in A_n$, and $q_n(i,j) = 0$ otherwise. In other words, at epoch n only the rows of the original matrix $\mathbf{Q}$ corresponding to the active states are considered (see Section 3 for examples). Now, the *adapted transition matrices* are defined as

$$\mathbf{P}_n = \mathbf{I} + \frac{1}{\lambda_n}\mathbf{Q}_n, \qquad n = 0, 1, 2, \ldots . \tag{9}$$

Let the stochastic process $T = \{T_n; n = 0, 1, 2, ...\}$ be defined as

$$T_n = Exp(\lambda_0) + Exp(\lambda_1) + ... + Exp(\lambda_{n-1}) \text{ and } T_0 = 0, \tag{10}$$

with $Exp(\lambda_i), i = 0, 1, ..., n \Leftrightarrow 1$, denoting an exponentially distributed random variable with rate λ_i. Furthermore, define $U_n(t)$ as the probability of exactly n jumps in $[0, t]$, i.e.,

$$U_n(t) = \Pr\{T_n \leq t \wedge T_{n+1} > t\}, \ t \geq 0, \ n = 0, 1, 2, \ldots . \tag{11}$$

Now adaptive uniformization computes the transient state probability distribution $\underline{\pi}(t)$ as

$$\underline{\pi}(t) = \underline{\pi}(0) \sum_{n=0}^{\infty} U_n(t) \prod_{i=0}^{n-1} \mathbf{P}_i = \sum_{n=0}^{\infty} U_n(t)\underline{\pi}_n, \tag{12}$$

with

$$\underline{\pi}_0 = \underline{\pi}(0) \text{ and } \underline{\pi}_n = \underline{\pi}_{n-1}\mathbf{P}_{n-1}, n = 1, 2, \ldots . \tag{13}$$

Finally, the infinite sum in (12) is truncated after $N_a(\epsilon_a, t)$ steps, with $N_a(\epsilon_a, t)$ such that

$$\sum_{n=0}^{N_a(\epsilon_a,t)} U_n(t) \geq 1 \Leftrightarrow \epsilon_a. \tag{14}$$

So, as in Equation (7) for SU, AU can produce a solution *with any desired accuracy* $\epsilon_a > 0$.

To aid in the discussion that follows, we introduce the following terminology. In particular, we call the pure birth process with transition rates $\lambda_0, \lambda_1, \lambda_2, ...$ the *AU-jump process*. In this process, the n-th *jump*, $n = 1, 2, ...$, takes an exponentially distributed time with rate λ_{n-1}, and *after* n jumps the AU-jump process is in state n, or at *epoch* n. The DTMC with initial distribution $\underline{\pi}_0$ and transition matrix $\mathbf{P}_n$ at epoch n, subordinated [2] to the AU-jump process, is called the *AU process*.

Note the analogy between Equation (12) and Equation (4). In (4), the stochastic process T defined by (10) corresponds to a Poisson process, and therefore we have $U_n(t) = e^{-\lambda t}(\lambda t)^n/n!$. SU is therefore a special case of AU, and AU allows for a more generic form of randomizing the time between jumps in a DTMC. Depending on the characteristics of the CTMC considered, AU may thus permit larger jumps than SU, thereby allowing truncation after fewer steps, while still obtaining the desired accuracy (i.e., $N_a(\epsilon_a, t) < N_s(\epsilon, t)$ for identical t and $\epsilon = \epsilon_a$).

2.2 Proof of Correctness

To show the correctness of AU, we take a sample path approach. In particular, we show that the AU process is stochastically identical to the original CTMC, i.e., in both processes the holding time in each state i is exponentially distributed with rate q_i, and given a transition out of state i, the process enters state j with probability $q(i,j)/q_i$. We assume that the AU-jump process does *not explode*, i.e., $\sum_{n=0}^{\infty} 1/\lambda_n = \infty$. Note that this is a very natural assumption, which also gives Kolmogorov's forward equation a unique solution.

Lemma 1 *If the AU-jump process does not explode, the holding time in the AU process of an active state $i \in S$ is exponentially distributed with rate q_i.*

Proof. Let $i \in S$ be an active state at epoch $l \geq 0$, i.e., $i \in A_l$, and without loss of generality we may renumber the jumps such that $l = 0$. The holding time distribution $H_i(\cdot)$ of state i in the AU process is defined by:

$$H_i(t) = \Pr\{\text{leave state } i \text{ before time } t\} = \tag{15}$$

$$= \Pr\{\cup_{n=1}^{\infty}\{\text{leave state } i \text{ at epoch } n \wedge \text{jump } n \text{ occurs before time } t\}\} = \tag{16}$$

$$= \sum_{n=1}^{\infty} \Pr\{\text{leave state } i \text{ at epoch } n\} \Pr\{\text{jump } n \text{ occurs before time } t\}. \quad (17)$$

The second multiplicative term, i.e., $\Pr\{\text{jump } n \text{ occurs before time } t\}$, equals $\Pr\{T_n \leq t\}$, defined in (10). To derive an expression for the first multiplicative term in (17) we observe that when state $i, i \in S$, is left for the first time at epoch n, the first $n \Leftrightarrow 1$ jumps must have been from i to i, i.e., via the added self loop in state i of the DTMC. Self loops occur with probability $1 \Leftrightarrow q_i/\lambda_k, k = 0, 1, ..., n \Leftrightarrow 2$. The n-th jump is out of state i, which occurs with probability q_i/λ_{n-1}. In total we have that $p_n = \Pr\{\text{leave state } i \text{ at epoch } n\}$ equals

$$p_n = \frac{q_i}{\lambda_{n-1}} \prod_{k=0}^{n-2} (1 \Leftrightarrow \frac{q_i}{\lambda_k}), \qquad n = 1, 2, ... \quad (18)$$

To assure that the products are well defined for all n we say that $\prod_{k=0}^{-1} g(k) = 1$ for any function g. The holding time distribution $H_i(t)$ is now given by:

$$H_i(t) = \sum_{n=1}^{\infty} p_n \Pr\{T_n \leq t\}. \quad (19)$$

and we will show that $H_i(t) = \Pr\{Exp(q_i) \leq t\}$. To do this, we use that the Laplace transform $\phi_i(s)$ of $H_i(\cdot)$ is [2]:

$$\phi_i(s) = \sum_{n=1}^{\infty} p_n \prod_{k=0}^{n-1} \frac{\lambda_k}{\lambda_k + s} = \sum_{n=1}^{\infty} \frac{q_i}{\lambda_{n-1}} \prod_{k=0}^{n-2} (1 \Leftrightarrow \frac{q_i}{\lambda_k}) \prod_{k=0}^{n-1} \frac{\lambda_k}{\lambda_k + s}. \quad (20)$$

If the holding time $H_i(\cdot)$ is exponentially distributed with rate q_i, $\phi_i(s)$ must equal the Laplace transform $q_i/(q_i + s)$ of an exponential distribution, i.e., for all $s \geq 0$ it should hold that

$$\sum_{n=1}^{\infty} \frac{q_i}{\lambda_{n-1}} \prod_{k=0}^{n-2} (1 \Leftrightarrow \frac{q_i}{\lambda_k}) \prod_{k=0}^{n-1} \frac{\lambda_k}{\lambda_k + s} = \frac{q_i}{q_i + s} \quad (21)$$

$$\Leftrightarrow \sum_{n=1}^{\infty} \frac{q_i}{\lambda_{n-1}} \frac{\lambda_{n-1}}{(\lambda_{n-1} + s)} \prod_{k=0}^{n-2} (1 \Leftrightarrow \frac{q_i}{\lambda_k}) \frac{\lambda_k}{\lambda_k + s} = \frac{q_i}{q_i + s} \quad (22)$$

$$\Leftrightarrow (q_i + s) \sum_{n=1}^{\infty} \frac{1}{\lambda_{n-1} + s} \prod_{k=0}^{n-2} \frac{\lambda_k \Leftrightarrow q_i}{\lambda_k + s} = 1 \quad (23)$$

$$\Leftrightarrow \sum_{n=0}^{\infty} \frac{q_i + s}{\lambda_n + s} \prod_{k=0}^{n-1} (1 \Leftrightarrow \frac{q_i + s}{\lambda_k + s}) = 1. \quad (24)$$

We now define $f_n = (q_i + s)/(\lambda_n + s), n = 0, 1, 2, ...$, and can look at this as the probability of reaching an absorbing state at epoch $n + 1$, in a Markov chain as in [2, Vol.1, p.400]. That the f_n, $n = 0, 1, 2, ...$, are probabilities follows

directly from $s \geq 0$ and $\lambda_n \geq q_i$, for all n and $i \in A(n)$. From a result from [2, Vol.1, p.400] it then follows that

$$\sum_{n=0}^{\infty} f_n \prod_{k=0}^{n-1} (1 \Leftrightarrow f_k) = 1 \quad \Leftrightarrow \quad \sum_{n=0}^{\infty} f_n = \infty. \tag{25}$$

Now we use the assumption that the AU-jump process does not explode, so $\sum_{n=0}^{\infty} 1/\lambda_n = \infty$. The following relation exists

$$\sum_{n=0}^{\infty} \frac{1}{\lambda_n} = \infty \Rightarrow \sum_{n=0}^{\infty} \frac{1}{(\lambda_n + s)} = \infty \Rightarrow \sum_{n=0}^{\infty} \frac{q_i + s}{\lambda_n + s} = \sum_{n=0}^{\infty} f_n = \infty, \tag{26}$$

and thus the equality (24) holds. So, when the AU-jump process does not explode, the holding time in the AU process is exponentially distributed with rate q_i, for every active state $i \in S$.

Theorem 1 *When the AU-jump process does not explode, the transient state probability distribution of the AU process, defined in Equation (12), is identical to the transient state probability distribution of the CTMC Y.*

Proof. We have shown that the state holding times of the AU process and the CTMC Y are identical. It can be seen that, given a jump out of state i occurs at epoch n in the AU process, it goes to state j with probability $p(i,j)/(1 \Leftrightarrow p(i,i))$. Now, because

$$\frac{p(i,j)}{(1 \Leftrightarrow p(i,i))} = \frac{q(i,j)/\lambda_n}{1 \Leftrightarrow (1 \Leftrightarrow q_i/\lambda_n)} = q(i,j)/q_i, \tag{27}$$

it follows that in the AU process and the original CTMC Y the transition probabilities, conditioned on the fact that a jump takes place, are equal. So, the AU process and the CTMC are stochastically identical and, hence, the transient state probabilities of both processes are identical [7].

3 Nature of the AU Process

Having shown that the AU process indeed mimics the probabilistic behavior of the original CTMC, it remains to be seen if and when AU will be computationally superior to SU. Before addressing that question directly, it helps to look at the nature of the AU process itself (i.e., the DTMC with initial distribution $\underline{\pi}_0$ and transition matrix $\mathbf{P}_n$ at epoch n, subordinated to the AU-jump process). The nature of the AU process determines the states that are active at each epoch, and hence, to a large extent, the cost associated with performing AU. Two issues are important here, and illustrated in this section via two examples.

Figure 1: Machine-repairman models with two and three components, respectively.

The first issue is whether the AU process is periodic or aperiodic. We say that an AU process is periodic if there exists a set of active states that is a proper subset of the state space and repeats itself at regularly-spaced epochs. While periodic AU processes do not appear to be common in practical applications, they often result in a substantial reduction in the necessary number of iterations required in the uniformization algorithm when they occur.

To illustrate this, consider a machine-repairman (MR) model in which there are N components, each having a failure rate ν. The repair rate is μ for each repair facility, and, for the moment, we assume that there is only one such a facility. Typically, the failure rate is much smaller than the repair rate, i.e., $\nu \ll \mu$. The system provides proper service when at least one component is working. The reliability of the model is then the probability that the system provides continuous correct service throughout the interval $[0, t]$ (see, for instance, [8]).

For the state space, we take $S = \{0, 1, 2, ..., N\}$, with state $i \in S$ representing the number of components that have failed. To derive the system reliability, we make state N, representing the whole system down, an absorbing state. Figure 1 shows such state spaces for $N = 2$ and $N = 3$. We can then compute $\pi_N(t)$, and, in turn, the reliability as $R(t) = 1 \Leftrightarrow \pi_N(t)$. When doing this, we assume the system starts in full operation mode, i.e., $\pi_0(0) = 1$.

The two component model ($N = 2$) has the following infinitesimal generator $\mathbf{Q}$:

$$\mathbf{Q} = \begin{pmatrix} \Leftrightarrow 2\nu & 2\nu & 0 \\ \mu & \Leftrightarrow(\nu + \mu) & \nu \\ 0 & 0 & 0 \end{pmatrix}. \tag{28}$$

AU leads to the following scheme:

$$\underline{\pi}_0 = (1,0,0),\ A_0 = \{0\},\ \mathbf{Q}_0 = \begin{pmatrix} \Leftrightarrow 2\nu & 2\nu & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},\ \lambda_0 = 2\nu,\ \mathbf{P}_0 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

$$\underline{\pi}_1 = (0,1,0),\ A_1 = \{1\},\ \mathbf{Q}_1 = \begin{pmatrix} 0 & 0 & 0 \\ \mu & \Leftrightarrow(\nu+\mu) & \nu \\ 0 & 0 & 0 \end{pmatrix},\ \lambda_1 = \nu + \mu,$$

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 0 & 0 \\ \frac{\mu}{\nu+\mu} & 0 & \frac{\nu}{\nu+\mu} \\ 0 & 0 & 1 \end{pmatrix}.$$

We see that for even values of $n \geq 2$:

$$\underline{\pi}_n = (+,0,+), A_n = \{0,2\}, \mathbf{Q}_n = \mathbf{Q}_0, \lambda_n = 2\nu, \mathbf{P}_n = \mathbf{P}_0,$$

with the '+'-sign denoting values greater than 0. For odd $n \geq 3$ we obtain:

$$\underline{\pi}_n = (0,+,+), A_n = \{1,2\}, \mathbf{Q}_n = \mathbf{Q}_1, \lambda_n = \nu + \mu, \mathbf{P}_n = \mathbf{P}_1.$$

Hence, by the definition above, the AU process for the MR model with $N = 2$ is periodic. Applying AU, we obtain a jump process with rates 2ν, $\nu+\mu$, 2ν, $\nu+\mu$, etc., while SU yields a Poisson process with rate $\nu + \mu$. As the repair rate of models like the two component model are typically orders of magnitude larger than the failure rates, i.e. $\mu \gg \nu$, using AU results in a substantial reduction in the necessary number of iterations in the uniformization algorithm.

Periodic AU processes are not typically encountered in realistic applications. A more typical behavior is where the adapted uniformization rate will remain constant after a finite number of jumps, and the AU-jump process therefore has a fixed rate from some epoch on. This behavior occurs in many models due to self loops in the AU process, which cause certain states to remain in the set of active states from some epoch on. We call this type of model *AU with converged rate*, and most applications of AU result in such a process. Rate convergence has two important implications in the development of AU algorithms. First, convergence implies that after some point, the rate of the AU-jump process will be equal to the rate employed in SU, and hence there will be no further savings obtained (in number of iterations required to achieve a particular accuracy) compared with SU. Second, it implies that there is a special structure to the AU-jump process (i.e., $\forall i$ greater than some constant m, $\lambda_i = \lambda_m$). Such a structure can be exploited in algorithms developed to compute $U_n(t)$, as will be seen in Section 4.

The three component MR model (i.e., the MR model with $N = 3$, from Figure 1) is an example of a model that results in an AU-jump process with converged rate. In this example, a self loop in state 0 of the AU process is introduced in epoch 2. Note that if the outgoing rate 3ν of state 0 is equal to the outgoing rate $\mu + \nu$ of state 2, there is no self loop in 0, and again we

would have had a periodic AU process. However, these rates will typically not be equal, resulting in the following growth in the number of active states:

$$\begin{array}{lll}
\underline{\pi}_0 = (1,0,0,0) & A_0 = \{0\} & \lambda_0 = 3\nu; \\
\underline{\pi}_1 = (0,1,0,0) & A_1 = \{1\} & \lambda_1 = \mu + 2\nu; \\
\underline{\pi}_2 = (+,0,+,0) & A_2 = \{0,2\} & \lambda_2 = \mu + \nu; \\
\underline{\pi}_3 = (+,+,0,+) & A_3 = \{0,1,3\} & \lambda_3 = \mu + 2\nu; \\
\underline{\pi}_n = (+,+,+,+) & A_n = \{0,1,2,3\} & \lambda_n = \mu + 2\nu \text{ for } n \geq 4.
\end{array}$$

The gain of AU compared to SU is less obvious than for the two component MR model, and strongly depends on specific model parameter values chosen. In Section 5 we will see what the reduction in number of jumps necessary with AU compared to SU can be for systems with converged rate, and what consequences this has for the computational complexity of the solution.

4 Computational Methods for the AU-Jump Process

Efficient and stable computation of the probabilities of differing numbers of jumps in the AU-jump process is necessary if adaptive uniformization is to be a viable solution method for CTMC's. To compute these probabilities, the probability of exactly n jumps ($n = 0, 1, 2, \ldots, N_a(\epsilon_a, t)$) in some birth process must be computed. Thus, any method that can be used to obtain transient solutions of acyclic Markov chains can potentially be employed. However, characteristics of the AU-jump process (e.g., the fact that it is a pure-birth process and may have distinct rates and/or converged rate) can sometimes be used to facilitate the computation. In this section, we discuss three methods that can be used to compute these probabilities, taking into account the special structure of the AU-jump process whenever possible. In particular, we discuss the following approaches:

1. Closed-form solution,
2. Acyclic Markov chain evaluator (ACE), and
3. Uniformization.

Each of the methods will be evaluated with regard to its accuracy and complexity. *Accuracy* denotes the difference between the numerically computed result and the actual (theoretical) result, and is thus influenced by elements such as the numerical stability [21] of the algorithm. *Complexity* is a measure of the computation time of the algorithm. We will use the number of arithmetic expressions which have to be carried out to achieve a certain accuracy as the indicator of the complexity of the method.

4.1 Closed-Form Solution

A simple, closed-form, expression for the jump probabilities exists when all the uniformization rates in the AU-jump process are different. To illustrate this, we express the AU-jump process as a pure birth process $B = \{B(t); t \geq 0\}$, over the state space $S = \{0, 1, 2, \ldots\}$, with intensity λ_{n-1} for the n-th jump, $n = 1, 2, 3, \ldots$. $U_n(t)$ is then expressed as $U_n(t) = \Pr\{B(t) = n\}$. Let $J(n)$ be the number of *different* values in the set $\{\lambda_0, \lambda_1, \ldots, \lambda_n\}$ and γ_j, $j = 1, \ldots, J(n)$, be the different parameter *values*. Furthermore, let $K_j(n)+1$ be the number of parameters in $\{\lambda_0, \lambda_1, \ldots, \lambda_n\}$ which equal γ_j. We use j_n to denote the value of $j, 1 \leq j \leq J(n)$, for which $\lambda_n = \gamma_j$.

The elements $\gamma_j, j = 1, ..., J(n)$, are then the different eigenvalues associated with the birth process B [11, 19]. $U_n(t)$ can thus be expressed as the following sum [11, 19]:

$$U_n(t) = \sum_{j=1}^{J(n)} \sum_{k=0}^{K_j(n)} a_{j,k}^{(n)} e^{-\gamma_j t} t^k \quad \text{for } n \geq 0 \text{ and } t \geq 0. \tag{29}$$

When the parameters λ_n are all different, a closed-form expression for the coefficients $a_{j,k}^{(n)}$ exists. In this case, $K_j(n) = 0$ for all n, and we therefore leave out the subscript k in the coefficients. The coefficients $a_j^{(n)}$ in (29) then equal [13, 23]

$$a_j^{(n)} = \frac{\prod_{i=1}^{n} \lambda_i}{\prod_{i=1|i\neq j}^{n+1} (\lambda_i \Leftrightarrow \lambda_j)} \quad \text{for } n \geq 1, \tag{30}$$

with $a_j^{(0)} = 1$.

To compute expression (30), one can derive rather straightforwardly an iterative scheme in n. Although the numerical accuracy of the method should be investigated, it appears that the expression is simple enough to derive satisfactorily accurate recursive schemes. Notice, however, that numerical difficulties might appear when the differences $\lambda_i \Leftrightarrow \lambda_j$ for pairs $i, j \leq n + 1$ in (30) are small.

The computation of (30) takes a few operations per coefficient when one computes $a_j^{(n)}$ from $a_j^{(n-1)}$, and in the n-th iteration there will be n coefficients to compute. Thus when N_a is the number of iterations in AU, the total number of arithmetic instructions required will be of order $O(\sum_{n=1}^{N_a} n) = O(N_a^2)$. Moreover, the fact that (30) is computed by a recursive scheme suggests that the required computation does not differ significantly from the ACE scheme, as will be seen in Subsection 4.2. Note that one might try to derive a closed-form solution for jump processes where not all rates are different, but that it thus is not clear whether this solution can be computed more efficiently than the generic scheme we discuss in the next subsection.

4.2 Acyclic Markov Chain Evaluator

Another possible way to compute the required probabilities in the AU-jump process is to use a recursive scheme developed by Severo [20] and later called the acyclic Markov chain evaluator (ACE) by Marie *et al.* [11]. Using ACE, the coefficients $a_{j,k}^{(n)}$ in (29) can be computed by an iterative scheme, regardless of the number of occurrences of each rate in the birth process. When doing this, n new coefficients $a_{\ldots}^{(n)}$ must be computed at epoch n. So when AU requires N_a iterations, $O(\sum_{n=1}^{N_a} n) = O(N_a^2)$ computations are required to compute the jump probabilities.

However, as argued in Section 3, the application of AU often leads to an AU-jump process with converged rate. For this case, we have derived a more efficient version of ACE, with order of complexity $O(N_a)$. The outline of this scheme is presented below, and a more complete derivation is presented in the Appendix. In particular, to compute the jump probabilities of an AU process with converged rate, we use the fact that the density of a hypo-exponentially distributed random variable is a sum of the form of (29) [23], and that the part of the AU-jump process with converged rate is a Poisson process, when looked upon separately.

More formally, let the rate in AU be converged from epoch m on, i.e., $\lambda_n = \mu$ for $n \geq m$. The density $f_H^{(m)}(t)$ of the hypo-exponentially distributed random variable with m exponential phases can then be written as follows:

$$f_H^{(m)}(t) = \sum_{j=1}^{J^-(m)} \sum_{k=0}^{K_j(m)} b_{j,k}^{(m)} e^{-\gamma_j t} t^k, \tag{31}$$

where the parameters are as in the previous subsection, except that $J^-(m)$ now denotes the number of different parameter values in the set $\{\lambda_0, \lambda_1, ..., \lambda_{m-1}\}$. Furthermore, let j_μ be the value of $j, 1 \leq j \leq J^-(m)$, for which $\gamma_j = \mu$, if such a j exists. Then define, for $j, k, r, l \in \aleph$, and $\zeta, t \in \Re^+$,

$$A_{j,k,r,t}^{(\zeta)}(l) = b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} \frac{(r+l)!}{l!} e^{-\zeta t} t^{k-r}, \tag{32}$$

$$B_{k,r,t}(l) = b_{j_\mu,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{l!(r+l+1)} e^{-\mu t} t^{k+l+1}, \text{and} \tag{33}$$

$$C_{j,r,t}(l) = \sum_{v=0}^{r+l} \frac{(\mu \Leftrightarrow \gamma_j)^v t^v}{v!}. \tag{34}$$

Then $U_{m+l}(t)$ can be expressed as (see the Appendix)

$$U_{m+l}(t) = \sum_{j=1|j\neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} [A_{j,k,r,t}^{(\gamma_j)}(l) \Leftrightarrow A_{j,k,r,t}^{(\mu)}(l) C_{j,r,t}(l)] + \sum_{k=0}^{K_{j_\mu}(m)} \sum_{r=0}^{k} B_{k,r,t}(l). \tag{35}$$

The terms, $A(l)$, $B(l)$, and $C(l)$ can be computed recursively, after defining $D(l)$, with $j, r, l \in \aleph, t \in \Re^+$, as follows:

$$D_{j,r,t}(l) = \frac{(\mu \Leftrightarrow \gamma_j)^{r+l}}{(r+l)!} t^{r+l}. \tag{36}$$

Then,

$$A^{(\zeta)}_{j,k,r,t}(l) = A^{(\zeta)}_{j,k,r,t}(l \Leftrightarrow 1) \frac{\mu}{(\mu \Leftrightarrow \gamma_j)} \frac{(r+l)}{l}; \tag{37}$$

$$B_{k,r,t}(l) = B_{k,r,t}(l \Leftrightarrow 1) \frac{\mu}{l} \frac{(r+l)}{(r+l+1)} t; \tag{38}$$

$$C_{j,r,t}(l) = C_{j,r,t}(l \Leftrightarrow 1) + D_{j,r,t}(l), \tag{39}$$

with

$$D_{j,r,t}(l) = D_{j,r,t}(l \Leftrightarrow 1) \frac{(\mu \Leftrightarrow \gamma_j)}{r+l} t. \tag{40}$$

For specific cases, even less complex schemes can be derived. For example, when all the non-converged rates are different from the converged rate (i.e., $\gamma_j \neq \mu$ for all $j = 1, 2, ..., J^-(m)$), the B-terms in (35) disappear, because j_μ is not defined. See [13] for detailed derivations of other possible schemes.

The computational complexity of the modified ACE scheme (for AU processes with converged rate) is very good. In particular, computing the terms $A(l), B(l), C(l)$ and $D(l)$ (with appropriate indices) from the recursive relations (37), (38), (39) and (40) requires only a constant number of operations per step, independent of the value of l. As a consequence, if N_a is the number of iterations in AU, the complexity of the modified ACE scheme is $O(N_a)$, which implies that computing the jump probabilities does not have a higher order complexity than carrying out a matrix-vector multiplication every of the N_a iterations.

However, as pointed out by other authors (e.g., [10]), the accuracy of the computational schemes which are based on ACE cannot be guaranteed, independent of the rates of the birth processes. This is a serious drawback of our modified ACE method as well, and needs careful study before any implementation of AU that uses ACE is made. Lindemann *et al.* [10] suggest that further modifications can be made to ACE that improve its stability, but it is not clear how much these modifications improve the method.

4.3 Uniformization

A third possible method to compute the probabilities of differing numbers of jumps in the AU-jump process is by standard uniformization. While using SU to compute these probabilities has a high complexity, it is numerically stable, and in some cases will have a lower overall complexity than using SU

alone. We call the use of SU to compute the jump probabilities in AU *layered uniformization* (LU), since uniformization is applied twice: first AU on the original CTMC, and then SU on the AU-jump process.

We consider two variants of the method, the first for general AU processes, and the second for AU processes with converged rate. As with the ACE approach, if the AU process has converged rate, we can use this fact to speed the computation time. In this section, we describe each approach, and show the error that results from truncation in both cases.

We first consider the use of uniformization to compute the jump probabilities for general AU processes. In particular, let the AU-jump process have transition rates $\lambda_0, \lambda_1, \ldots, \lambda_{N_a}$, leading to the following generator matrix $\mathbf{Q}_B$:

$$\mathbf{Q}_B = \begin{pmatrix} \Leftrightarrow\lambda_0 & \lambda_0 & 0 & 0 & \cdots & 0 \\ 0 & \Leftrightarrow\lambda_1 & \lambda_1 & 0 & \cdots & 0 \\ 0 & 0 & \Leftrightarrow\lambda_2 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \Leftrightarrow\lambda_{N_a} & \lambda_{N_a} \end{pmatrix}. \tag{41}$$

Then let $\beta \geq \max\{\lambda_i \mid i = 0, 1, \ldots, N_a\}$. We can then apply uniformization and obtain the stochastic matrix $\mathbf{P}_B = (p_B(i,j))$, defined as:

$$\mathbf{P}_B = \mathbf{I} + \frac{\mathbf{Q}_B}{\beta} = \begin{pmatrix} 1 \Leftrightarrow \lambda_0/\beta & \lambda_0/\beta & 0 & 0 & \cdots & 0 \\ 0 & 1 \Leftrightarrow \lambda_1/\beta & \lambda_1/\beta & 0 & \cdots & 0 \\ 0 & 0 & 1 \Leftrightarrow \lambda_2/\beta & \lambda_2/\beta & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & 1 \Leftrightarrow \lambda_{N_a}/\beta & \lambda_{N_a}/\beta \end{pmatrix} \tag{42}$$

with initial distribution $\pi_0^B(0) = 1$. Uniformization then gives the following expression for $U_n(t)$:

$$U_n(t) = \sum_{k=0}^{N_B} e^{-\beta t} \frac{(\beta t)^k}{k!} \pi_k^B(n), \tag{43}$$

with N_B denoting the number of iterations necessary to derive the jump probabilities with pre-specified error tolerance ϵ_B and with $\pi_k^B(n)$ denoting the probability that the DTMC (defined by $\mathbf{P}_B$ and $\underline{\pi}_0^B$) is at the k-th epoch in state n, which can be computed from

$$\pi_0^B(0) = 1 \text{ and } \underline{\pi}_k^B = \underline{\pi}_{k-1}^B \mathbf{P}_B \text{ for } k = 1, 2, \ldots. \tag{44}$$

Now, combining the definition of AU (equations (12), (14)) with (43), leads to the following formula defining LU:

$$\underline{\pi}(t) = \sum_{n=0}^{N_a} \Big(\sum_{k=0}^{N_B} e^{-\beta t} \frac{(\beta t)^k}{k!} \pi_k^B(n) \Big) \underline{\pi}_n. \tag{45}$$

The error incurred by using LU can be bounded in the following manner. In particular, let the error tolerance (14) for AU be ϵ_a, then the total error ϵ for LU is bounded by

$$\epsilon = \|\underline{\pi}(t) \Leftrightarrow \sum_{n=0}^{N_a}(\sum_{k=0}^{N_B} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n))\underline{\pi}_n\|_\infty \leq \epsilon_B + \epsilon_a. \tag{46}$$

To show this, first note that

$$\sum_{n=0}^{N_a}\sum_{k=N_B+1}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n) = \sum_{k=N_B+1}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!}\sum_{n=0}^{N_a}\pi_k^B(n) =$$

$$\sum_{k=N_B+1}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!} \leq \epsilon_B. \tag{47}$$

Now, for any $i \in S$ we have that

$$\sum_{n=0}^{N_a}(\sum_{k=0}^{N_B} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n))\pi_n(i) \leq \pi_i(t) = \sum_{n=0}^{\infty}(\sum_{k=0}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n))\pi_n(i) \tag{48}$$

$$\leq \sum_{n=0}^{N_a}\sum_{k=0}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n)\pi_n(i) + \epsilon_a \tag{49}$$

$$= \sum_{n=0}^{N_a}(\sum_{k=0}^{N_B} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n) + \sum_{k=N_B+1}^{\infty} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n))\pi_n(i) + \epsilon_a \tag{50}$$

$$\leq \sum_{n=0}^{N_a}(\sum_{k=0}^{N_B} e^{-\beta t}\frac{(\beta t)^k}{k!}\pi_k^B(n))\pi_n(i) + \epsilon_B + \epsilon_a \tag{51}$$

and (46) follows.

In the special case that the AU process has converged rate μ from some epoch m on, and uniformization of the AU-jump process is carried out with the converged rate $\beta = \mu$, the computational scheme can be simplified considerably. In this case, it is evident that $p_B(i, i+1) = 1$ for $i \geq m$ ($p_B(i, i+1)$ as defined in (42)). Consequently, (44) reduces to

$$\pi_k^B(n) = \pi_{k-1}^B(n \Leftrightarrow 1), \text{ for } n > m. \tag{52}$$

Now, if the elements $\pi_k^B(m)$ are stored in memory, $\pi_k^B(n)$, for $n \geq m$ and $k \geq n \Leftrightarrow m$ is just

$$\pi_k^B(n) = \pi_{k+m-n}^B(m). \tag{53}$$

So, the probabilities $\pi_k^B(n)$ for all $n \geq m$ are known at epoch m, which substantially reduces the amount of computation necessary to perform LU.

Method	Stability	Order Complexity
Closed-form	Not Stable	N_a^2
ACE	Not Stable	N_a^2
Modified ACE	Not Stable	N_a
Uniformization	Stable	$N_B N_a$
Uniformization with converged rate	Stable	$\sqrt{N_B} N_a$
Poisson probabilities	Stable	$\sqrt{N_s}$

Table 1: Stability and order complexity when computing the jump probabilities in AU and SU.

The accuracy of LU is good, but the complexity is high. In particular, LU leads to a stable numerical algorithm since uniformization itself is stable, and it thus can be expected to give accurate results. The computational complexity of computing the jump probabilities is that of uniformization with uniformization rate β for a CTMC with $N_a + 1$ states. Thus the general case (non-converged rate) demands $O(N_B)$ (with N_B as defined in (43)) matrix vector multiplications for squared matrices of size N_a, giving a total complexity of $O(N_B N_a)$.

When the AU process has a converged rate, left truncation can be applied, as in the case of computing Poisson probabilities in SU, i.e., one only computes (52) for $k \geq L$, with L the left truncation point, chosen such that the Poisson probabilities $e^{-\beta t}(\beta t)^l / l!$ can be neglected for $l < L$ (see for more details [14]). This reduces the complexity to $O(\sqrt{N_B})$ operations at every iteration in the AU process (as shown in [3]), and consequently the total complexity becomes $O(\sqrt{N_B} N_a)$.

4.4 Comparison of Methods

Table 1 lists the characteristics of the discussed methods for computing the jump probabilities in AU, as well as for computing the Poisson probabilities in SU by the scheme of Fox and Glynn [3]. N_a, N_s and N_B are defined in (14), (7) and (43), i.e., they denote the necessary number of iterations in AU, SU and in the uniformization applied to the AU-jump process in LU, respectively. So, N_a is always less than or equal to N_s, but their absolute difference is problem dependent. Furthermore, uniformization of the AU-jump process can be carried out with the same uniformization rate as standard uniformization of the original CTMC Y if it has converged rate (see [14] for more details). In that case $N_B = N_s$.

We see from Table 1 that none of the methods outperforms the computation of Poisson probabilities in SU. Furthermore, the only numerically stable method for computing the jump probabilities for AU in Table 1 is uniformiza-

tion, which has a high time complexity. On the other hand, the modified ACE algorithm is the least time consuming method, but is not numerically stable. This suggests that further study is necessary to determine a "best" method for computing the jump probabilities in AU. In spite of the higher order of LU than SU, and the numerical instability of ACE, there are cases where AU will outperform SU. The situations when this will occur are investigated in the next section, in the context of a machine-repairman problem with delayed repair.

5 Extended Machine-Repairman Example

The previous section gave order-of-magnitude estimates of the complexity of computing the jump probabilities of the AU-jump process. While these estimates are important in determining the overall complexity of adaptive uniformization, the complexity also depends on other aspects of the computation, and is dependent on the specific characteristics of the system under study. To investigate these issues, we consider a machine-repairman example with delayed repair. The intent is not to give a precise accounting of the costs of a specific implementation of AU and SU, but to evaluate qualitatively, when AU is computationally attractive. This will be done by counting the number of operations that need be performed using SU, AU using ACE with converged rate, and LU (AU with SU to compute the jump probabilities).

5.1 System Description

The system considered is an extended machine-repairman (EMR) model with delayed repairs. There are K components, each with failure rate ρ, and repair starts when r components have failed, and continues until all components have been repaired. At this point, repair is once again disabled, and does not begin again until r components have once again failed. A failure is a *hard* failure with probability $1 \Leftrightarrow c$ and a *soft* failure with probability c, c being the so-called *coverage factor*. Repair takes an exponentially distributed time with parameter μ for a hard failure, and with parameter ν for a soft failure. Every failed component has its own repair facility. Typically $\nu > \mu$ and $\nu, \mu \gg \rho$. System failure occurs when all components have failed. We are interested in the reliability of the system, starting from a fully operational system.

To construct a CTMC for the EMR model, we define the state space as the set $S = \{(i, j, b) \mid 0 \leq i, j \leq K; i + j \leq K; b \in \{0, 1\}\}$, with i denoting the number of hard failed components, j the number of soft failed components, and b whether repair is enabled ($b = 1$) or not ($b = 0$). When $i = j = 0$ we always have $b = 0$. To compute the reliability, all states in which the system is no longer providing proper service have been joined to one absorbing state

Figure 2: Extended machine repairman model with four components ($K = 4$) and delayed repair starting after two failures ($r = 2$).

d. See Figure 2 for an example of the CTMC with $K = 4$ and $r = 2$. The reliability $R(t)$ is then defined as: $R(t) = 1 \Leftrightarrow \pi_d(t)$. We start with a fully operational system, so $\pi_{(0,0,0)}(0) = 1$. For highly reliable applications, such as those encountered in avionics, the mission time t will typically be considerably less than the mean time until the first failure $1/K\rho$. In other applications, such as distributed computer systems, a mission time higher than $1/K\rho$ might be more common. We will thus study the computational cost for a wide range of t's, to see the effect of the choice of method in each case.

The first step in the solution process, when using AU, is to construct the AU-jump process for the CTMC. For this example, we obtain:

$$\lambda_0 = K\rho, \lambda_1 = (K \Leftrightarrow 1)\rho, \ldots, \lambda_{r-1} = (K \Leftrightarrow r+1)\rho, \tag{54}$$

$$\lambda_r = r\nu + (K \Leftrightarrow r)\rho, \lambda_{r+1} = (r+1)\nu + (K \Leftrightarrow r \Leftrightarrow 1)\rho, \ldots, \lambda_{K-1} = (K \Leftrightarrow 1)\nu + \rho,$$

$$\lambda_K = (K \Leftrightarrow 2)\nu + 2\rho, \lambda_{K+1} = (K \Leftrightarrow 1)\nu + \rho, \lambda_{K+2} = (K \Leftrightarrow 1)\nu + \rho, \ldots.$$

In the following, we have used the above adapted uniformization rates, except for λ_K, which was taken to be $\lambda_K = (K \Leftrightarrow 1)\nu + \rho$. This was done so the AU process would have a converged rate after $K \Leftrightarrow 1$ jumps.

5.2 Operation Counts

To determine the computational cost associated with SU, AU with modified ACE, and LU for this example, we need to count the number of operations

Figure 3: The number of iterations needed with AU and SU for the EMR model, as function of t.

necessary to achieve the desired accuracy for the given model parameter values. Looking at equations (6) (SU) and (14) (AU), we see that we must first calculate the number of iterations required to reach the specified accuracy. This was done for SU by direct solution of (6), and for AU by solving (using standard uniformization) the AU-jump process and determining the number of states in the process that need to be considered to achieve the desired accuracy. For this computation, ϵ and ϵ_a were taken to be 10^{-4}, $K = 20$, $\rho = 1$ and $\nu = 1000$. r and t were varied to see their influence on the number of jumps required.

Figure 3 gives (in logarithmic scale), the number of iterations $N_a(t)$ necessary with AU for different repair strategies ($r = 5, 10, 15$ and 20), and compares them with the number of iterations $N_s(t)$ needed with SU to achieve the same accuracy. In the figure, the upper curve (marked SU) gives the number of iterations necessary with standard uniformization. The remaining curves give the number of iterations needed with AU, for the values of r as marked. Note that as expected, independent of r, $N_s(t) > N_a(t)$. Furthermore, note the curves all have the same values for t with $N_a(t) < r$, as the adaptive uniformization rates in (54) are in that case independent of the value of r.

Given the required numbers of iterations for each method, we can now count the total number of operations necessary to obtain the desired solution.

In this calculation, we consider additions, subtractions, multiplications, and divisions equal in cost, and to have a cost of one operation. Implementation details that may require additional operations (such as work necessary to prevent arithmetic underflow and overflow) are not considered, since the goal is to compare the algorithms themselves, not a particular implementation. The costs associated with each method were broken into three categories:

1. The operations necessary to generate the $\mathbf{P}_n$ (i.e., computation of (9), denoted *GEN*, in the following),

2. The operations necessary to perform the required matrix-vector multiplications $\underline{\pi}_n = \underline{\pi}_{n-1}\mathbf{P}_{n-1}$ (denoted *MVM*), and

3. The operations necessary to compute the jump probabilities $U_n(t)$ (denoted *JPR(ACE)* if by ACE, and *JPR(LU)* if by SU).

The number of operations necessary to generate the required P_n (*GEN*) depends on the structure of the CTMC, and the number of different matrices which have to be generated. In particular, to compute 1), the structure of the EMR-model was analyzed and the number of operations required to generate each matrix was computed. These costs were then summed to obtain the total cost for this component. For small time points, only non-converged rates need be considered, and the number of generations depends on N_a. For larger time points, states in which the jump process has converged are reached, and further generations are not necessary. Thus for large t the number of operations necessary for *GEN* has an upper bound, independent of N_a.

Computation of the number of operations necessary to perform the matrix-vector multiplications (*MVM*) in AU was done in a similar manner, except that a new multiplication is necessary at each iteration. To obtain the total cost, the cost for each iteration was calculated, and these costs were summed together. Note that only active states need to be considered in the multiplication, and hence not all multiplications have the same cost.

Computation of *JPR* in AU depends on the method used for the computation. For ACE with converged rate, we simply counted the required number of operations, using equations (35-40). For LU, we computed the number of operations needed to perform SU on the AU-jump process, using the method presented in Subsection 4.3. For this computation, N_B was taken to be equal to N_s, and the converged nature of the AU-jump process was taken into account. Note that the *MVM* required in performing this calculation is much less costly than required if SU was performed on the original CTMC, since the AU-jump process is a pure birth process.

Finally, computation of the number of operations necessary to perform SU was done in a similar manner to that for computing the jump probabilities via

Figure 4: The contribution of *JPR*, *GEN* and *MVM* in the complexity of AU for the EMR model with $r = 10$, as function of t.

LU except that the *MVM* now was done on the transition matrix constructed as in (2).

5.3 Results

Figure 4 presents the count of operations for each of the categories just discussed when $r = 10$. As can be seen in the figure, both the computational cost of AU with modified ACE (marked *JPR(ACE)*) and with uniformization (marked *JPR(LU)*) are given. The graph gives a good indication of the relative complexities of the various parts of the AU computation. In particular, note that when the uniformization rate has converged no new matrices $\mathbf{P}_n$ need be computed and the cost of *GEN* remains constant. As a consequence, the contribution of *GEN* becomes negligible for large t. Furthermore, it can be seen that *MVM* requires the most work, even though the model considered is relatively small.

With regard to the computation of the jump probabilities (*JPR*), we see that the modified ACE scheme *JPR(ACE)* is computationally less expensive than layered uniformization (*JPR(LU)*), as was expected from the discussion in Section 4. Note further that because *JPR(LU)* is of higher order ($O(\sqrt{N_B}N_a)$) than *MVM* ($O(N_a)$), *JPR(LU)* increases faster than *MVM*. For

Figure 5: The difference between complexity of AU and SU for the EMR with $r = 10$, as function of t, for small values of t.

large t *JPR(LU)* will thus become the computationally most intensive factor. *JPR(ACE)*, however, remains less than *MVM*, as it has the same order but a lower "constant." Finally, note that for small t *JPR(LU)* profits least from the fact that only a few iterations are necessary with AU. This can be explained by the fact that uniformization of the AU-jump process requires a considerable number of iterations, even for small t.

Having examined the components that make up the AU computation, the total computation cost of SU, AU with ACE and LU is now examined. Figure 5 shows the total operation cost for the EMR model when $r = 10$, for small values of t. As before, both AU with modified ACE (marked $AU \Leftrightarrow ACE$) and LU (marked LU) are considered. For small t, AU outperforms SU since it demands considerably fewer iterations. When t increases, however, LU becomes less attractive than SU, because it has higher complexity (note that LU does not result in a straight line). In particular, there exists a turning point t^*_{LU} such that LU demands less computational effort than SU when $t < t^*_{LU}$, and more effort when $t > t^*_{LU}$.

AU with modified ACE also exhibits a turning point, with respect to SU, but at a higher value of t, as can be seen in Figure 6. Although both AU with ACE and SU have order complexity linear in the number of iterations necessary, the "constant" in AU with ACE is higher. As a consequence, there exists

Figure 6: The difference between complexity of AU and SU for the EMR model with $r = 10$, as function of t.

a turning point t^*_{ACE}, such that AU with ACE demands less computational effort when $t < t^*_{ACE}$, and more effort when $t > t^*_{ACE}$.

As illustrated by the EMR example, there are situations where AU (with ACE or SU to compute the jump probabilities) will perform better than SU, and cases when it will perform worse. The conditions that lead to such differing performance are model specific, and depend on the particular point in time considered. For example, when the state-space size of the CTMC increases, the cost of *MVM* will become an even more prominent factor. The size of the model considered in this section was very small, only a few hundred states. In larger state spaces, as would be expected in realistic system models, the turning points would probably be orders-of-magnitude larger. Furthermore, the "stiffness," by which we mean the difference in magnitude of values of parameters in the model, influences the speed-up achievable with AU. If the model is very stiff, SU will perform poorly. AU will be much better if there are several jumps until states with high exit rates become active. Other characteristics that can influence the relative speeds of AU and SU include: the desired accuracy ϵ, the number of time points t of interest, and the chosen adapted uniformization rates.

The relative difference between AU and SU is thus very problem dependent, and it is hard to tell, from a single example, how useful it will be in practice. It

is clear, however, that it is better than SU in some cases, and that experimental work should be done to determine how common these cases are.

6 Summary and Conclusions

This paper introduces adaptive uniformization, a generalization of standard uniformization. In adaptive uniformization, a discrete time Markov chain is subordinated to a jump process that, contrary to SU, is not necessarily a Poisson process. Using a more general jump process can, in most cases, reduce the number of iterations necessary to compute the state-occupancy probabilities of the process at some time t with a specified accuracy. Although there will normally be more computational overhead per iteration, the reduction in number of iterations can sometimes outweigh the increased cost per iteration, and reduce overall the cost of solution.

In the paper, we first formally defined adaptive uniformization and proved the correctness of the method. We then discussed characteristics of the AU process that can facilitate solution, followed by alternative methods for computing the jump probabilities. This computation is a key, and difficult, part of the AU algorithm. The computation is key for two reasons: it may have much higher cost than the corresponding Poisson probability calculation in SU, and it may not be computationally stable, depending on the method chosen for computing the jump probabilities. We proposed three methods by which these probabilities could be calculated, and discussed the relative complexity and stability of each. Furthermore, we showed that more efficient algorithms can be developed for AU processes with converged rate, which we believe often arise in realistic examples. For example, for AU processes with converged rate, we derived a tailored ACE scheme with complexity linear in the necessary number of jumps. Similarly, we derived a layered uniformization method which takes advantage of the converged-rate nature of many problems.

Finally, we illustrated the computational cost of AU (relative to SU) through its application to an extended machine-repairman model. The example considered illustrated the different computational costs associated with SU, AU with ACE, and LU. In particular, we identified when AU-ACE and LU can be expected to be computationally more attractive than SU for this example. Specifically, AU-ACE and LU both outperform SU for small values of t, and exhibit a turning point t^* such that AU becomes computationally more intensive than SU for $t > t^*$. While we have only shown the existence of these turning points for a single example, we believe that they occur in most applications.

While there are still many practical issues that need to be resolved, we believe that adaptive uniformization has advantages over standard uniformization in many cases. Work is ongoing to implement the variations of AU pre-

sented herein, and compare, precisely, the relative merits of AU and SU for a wide range of examples. It would also be interesting to investigate the extension of AU to solve for other measures of interest, such as mean cumulative rewards [19] and cumulative reward distributions [22]. AU might also be profitably used to cope with the state-space explosion problem, by combining it with techniques such as those in [15, 17].

7 Acknowledgments

We like to acknowledge Boudewijn R. Haverkort, Tele-Informatics, University of Twente, and Erik A. van Doorn, Department of Mathematics, University of Twente, for suggestions that helped to improve the paper. We would also like to acknowledge John D. Diener, Department of Electrical and Computer Engineering, University of Arizona, for writing computer programs to calculate the number of iterations necessary for SU and AU to achieve a desired error bound, and for suggestions that helped improve the paper. We also would like to thank the three anonymous reviewers, who made suggestions that substantially improved the content of the paper.

Much of this work was done while A. P. A. van Moorsel was visiting the University of Arizona, under the support of the Netherlands Organization for Scientific Research (NWO), grants R62-385 and SIR13-1260. W. H. Sanders has been supported, in part, by the Digital Faculty Program: Incentives for Excellence.

REFERENCES

[1] Dunkel, J. and Stahl, H. On the transient analysis of stiff Markov chains, *Proc. Third Working Conference on Dependable Computing for Critical Applications*, Mondello, Italy, Sept. 1992.

[2] Feller, W. *An Introduction to Probability Theory and Its Applications, Volume I and II*, John Wiley, New York, 1966.

[3] Fox, B. L. and Glynn, P. W. Computing Poisson probabilities, *Comm. ACM* **31**, pp. 440-445, 1988.

[4] Grassmann, W. K. Transient solutions in Markovian queueing systems, *Comput. & Ops Res.* **4**, pp. 47-53, 1977.

[5] Grassmann, W. K. Finding transient solutions in Markovian event systems through randomization, *Numerical Solution of Markov Chains*, W.J. Stewart (Ed.), Marcel Dekker, New York, 1991.

[6] Gross, D. and Miller, D. R. The randomization technique as a modeling tool and solution procedure for transient Markov processes, *Opns Res.* **32**, pp. 343-361, 1984.

[7] Heyman, D. P. and Sobel, M. J. *Stochastic Models in Operations Research*, McGraw-Hill, New York, 1982.

[8] Laprie, J.-C. ed., *Dependability: Basic Concepts and Terminology*, Springer Verlag, Wien, 1992.

[9] Lindemann, C. An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models, *Perf. Eval.* **18**, pp. 79-95, 1993.

[10] Lindemann, C., Malhotra, M. and Trivedi, K. S. Numerical methods for reliability evaluation of closed fault-tolerant systems, *Technical Report*, Duke University, 1992.

[11] Marie, M. R., Reibman, A. L. and Trivedi, K. S. Transient analysis of acyclic Markov chains, *Perf. Eval.* **7**, pp. 175-194, 1987.

[12] Melamed, B. and Yadin, M. Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes, *Opns Res.* **32**, pp. 927-944, 1984.

[13] van Moorsel, A. P. A. and Sanders, W. H. Adaptive uniformization: Technical details, *Memoranda Informatica* 93-13, University of Twente, The Netherlands, 1993.

[14] van Moorsel, A. P. A. *Performability Evaluation Concepts and Techniques*, Ph.D.-thesis, University of Twente, The Netherlands, 1993.

[15] Muppala, J. K. and Trivedi, K. S. Numerical transient solution of finite Markovian queueing systems, in *Queueing and Related Models*, U.Bhat (Ed.), Oxford University Press, 1992.

[16] Nicol, D. M. and Heidelberger, P. Parallel Simulation of Markovian Queueing Networks Using Adaptive Uniformization, *Proc. ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Santa Clara, CA, pp. 135-145, May 1993.

[17] Qureshi, M. A. and Sanders, W. H. Reward model solution methods with impulse and rate rewards: An algorithm and numerical results, to appear in *Performance Evaluation.*

[18] Reibman, A. and Trivedi, K. S. Numerical transient analysis of Markov models, *Comput. Opns Res.* **15**, pp. 19-36, 1988.

[19] Reibman, A., Smith, R. and Trivedi, K. S. Markov and Markov reward model transient analysis: An overview of numerical approaches, *Eur. J. Op. Res.* **40**, pp. 257-267, 1989.

[20] Severo, N. C. A recursion theorem on solving differential-difference equations and applications to some stochastic processes, *J. Appl. Prob.* **6**, pp. 673-681, 1969.

[21] Stoer, J. and Bulirsch, R. *Introduction to Numerical Analysis*, Springer Verlag, New York, 1980.

[22] de Souza e Silva, E. and Gail, H. R. Calculating cumulative operational time distributions of repairable computer systems, *IEEE Tr. Comp.* **35**, pp. 322-332, 1986.

[23] Trivedi, K. S. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, 1982.

[24] Yoon, B. S. and Shanthikumar, J. G. Bounds and approximations for the transient behavior of continuous-time Markov chains, *Prob. Eng. Inf. Sc.* **3**, pp. 175-198, 1989.

Appendix

In this appendix, we give the derivation of the modified ACE scheme for AU with converged rate. In particular, we will show that for AU with converged rate μ after m jumps, $U_{m+l}(t)$ is given, for $l = 0, 1, 2, \ldots$, by

$$U_{m+l}(t) = \sum_{k=0}^{K_{j_\mu}(m)} \sum_{r=0}^{k} b_{j_\mu,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{l!(r+l+1)} e^{-\mu t} t^{k+l+1} \tag{55}$$

$$+ \sum_{j=1|j\neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} \frac{(r+l)!}{l!} e^{-\gamma_j t} t^{k-r} \tag{56}$$

$$\Leftrightarrow \sum_{j=1|j\neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} \frac{(r+l)!}{l!}$$

$$e^{-\mu t} t^{k-r} \sum_{v=0}^{r+l} \frac{(\mu \Leftrightarrow \gamma_j)^v t^v}{v!}, \tag{57}$$

with j_μ denoting the value of j for which $\gamma_j = \mu$.

To do this, we use the fact that the birth process is constructed out of two parts; the first m jumps are such that the probability of having m jumps before time t is hypo-exponentially distributed, and the last l jumps behave

as a Poisson process, when looked upon separately. The hypo-exponential density with m phases is given by the expression (31). We thus obtain

$$U_{m+l}(t) = \int_0^t f_H^{(m)}(t \Leftrightarrow s) \frac{(\mu s)^l}{l!} e^{-\mu s} ds \tag{58}$$

$$= \int_0^t \sum_{j=1}^{J^-(m)} \sum_{k=0}^{K_j(m)} b_{j,k}^{(m)} e^{-\gamma_j (t-s)} (t \Leftrightarrow s)^k \frac{(\mu s)^l}{l!} e^{-\mu s} ds. \tag{59}$$

Using $(t \Leftrightarrow s)^k = \sum_{r=0}^k \frac{k!}{r!(k-r)!} t^{k-r} (\Leftrightarrow s)^r$ we obtain

$$U_{m+l}(t) = \sum_{j=1}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{\mu^l}{l!} \frac{k!}{(k \Leftrightarrow r)! r!} e^{-\gamma_j t} t^{k-r} \int_0^t s^{r+l} e^{-(\mu-\gamma_j)s} ds. \tag{60}$$

We separate the summation in (60) in terms of j, such that $\mu = \gamma_j$ (i.e., $j = j_\mu$) and $\mu \neq \gamma_j$. This is necessary since it is possible that μ is also one of the rates in the first m non-converged jumps. When $\mu = \gamma_j$ the integral expression in (60) reduces to $\int_0^t s^{r+l} ds = t^{r+l+1}/r + l + 1$. When $\mu \neq \gamma_j$ we use that, e.g., [2],

$$\int_0^t s^{r+l} e^{-(\mu-\gamma_j)s} ds = \frac{(r+l)!}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} (1 \Leftrightarrow e^{-(\mu-\gamma_j)t} \sum_{v=0}^{r+l} \frac{(\mu \Leftrightarrow \gamma_j)^v t^v}{v!}) \tag{61}$$

So, we obtain:

$$U_{m+l}(t) = \sum_{k=0}^{K_{j_\mu}(m)} \sum_{r=0}^{k} b_{j_\mu,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{l!} e^{-\mu t} t^{k-r} \frac{t^{r+l+1}}{(r+l+1)} \tag{62}$$

$$+ \sum_{j=1|j\neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{l!} \frac{(r+l)!}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} e^{-\gamma_j t} t^{k-r} \tag{63}$$

$$\Leftrightarrow \sum_{j=1|j\neq j_\mu}^{J^-(m)} \sum_{k=0}^{K_j(m)} \sum_{r=0}^{k} b_{j,k}^{(m)} (\Leftrightarrow 1)^r \frac{k!}{(k \Leftrightarrow r)! r!} \frac{\mu^l}{(\mu \Leftrightarrow \gamma_j)^{r+l+1}} \frac{(r+l)!}{l!}$$

$$e^{-\mu t} t^{k-r} \sum_{v=0}^{r+l} \frac{(\mu \Leftrightarrow \gamma_j)^v t^v}{v!}. \tag{64}$$

Equations (62), (63) and (64) are close to final form, but can be simplifed, resulting in the desired result, as given in equations (55), (56) and (57). Finally, an inspection of (55), (56) and (57) shows that the required terms in the summation can be written recursively, as was given in equations (37), (38), (39) and (40). We thus obtain the results that are given in Section 4.2.