This paper is a prepublication version of the paper that appeared in *IEEE Transactions on Reliability*, vol. 46, no. 3, September 1997, pp. 430-440.

TRANSIENT SOLUTION OF MARKOV MODELS BY COMBINING ADAPTIVE AND STANDARD UNIFORMIZATION[*]

Aad P. A. van Moorsel[†] and William H. Sanders[‡]

[†] Bell Laboratories
Lucent Technologies
600 Mountain Ave., Murray Hill, NJ 07974
aad@bell-labs.com

[‡] Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL 61801
whs@crhc.uiuc.edu

Key words: Markov Processes, Transient Solution, Uniformization, Reliability Evaluation, Dependability Evaluation

ABSTRACT

Adaptive uniformization (AU) has recently been proposed as a method to compute transient measures in continuous-time Markov chains and has been shown to be especially attractive for solving large and stiff dependability models. The major advantage of AU is that it requires at most as many iterations as standard uniformization (SU), and often far fewer, thus resulting in substantial computational savings. However, this computational gain can be offset by the need to compute more complex "jump probabilities" in AU, whose computation is more expensive than computing Poisson probabilities in SU. In particular, it can be shown that AU is computationally superior to SU if and only if the considered time instant is less than some threshold time value. To overcome this drawback, we combine AU and SU such that AU is used over the start of the time interval of interest, while SU is applied to the rest of the time interval. We show that combined AU/SU can be implemented in such a way that the combination introduces only minor computational overhead, the number of iterations required is almost as low as AU, and the cost of computing the jump probabilities

[*]This work was carried out when Aad van Moorsel was with the University of Illinois, at Urbana-Champaign.

is as low as SU. Furthermore, combined AU/SU will be shown to yield a strict lower bound of the true result, within any desired pre-specified accuracy. The derived error bounds take into account the error introduced when the Fox/Glynn algorithm is used for computing Poisson probabilities, and we will enhance this algorithm to optimize its error bound characteristics. Implementations of SU and AU that are based on the Fox/Glynn method can benefit from these results, since more accurate error bounds can be determined. To demonstrate the benefits of combined AU/SU, we apply it to a machine-repairman model, using a version of combined AU/SU implemented in *UltraSAN*, a performance and dependability evaluation software package.

I Introduction

When evaluating the dependability of computer systems, one is often interested in transient measures, that is, measures defined over a finite time interval or at a finite point in time. Uniformization (e.g., [11, 18]) is often proposed as a transient solution method for continuous-time Markov chains (CTMCs). The advantages of standard uniformization (SU) include its numerical stability and the fact that the (inevitable) truncation error can be made arbitrarily small.

A potential drawback of SU is that a very large number of iterations (i.e., matrix-vector multiplications) may be needed to achieve a specified accuracy, resulting in a very large computation time. For example, solution of CTMC models of highly reliable, repairable systems will be very time consuming if, as is typical, the repair rates are orders of magnitude higher than failure rates and the time point of interest is on the order of the failure times.

To alleviate this problem, a new variant of uniformization, called "adaptive uniformization," has been proposed [19, 21]. In adaptive uniformization (AU), the number of iterations is reduced by using a uniformization rate that depends on which states can be reached at different uniformization epochs. In dependability models with greatly differing failure and repair rates, this technique will be profitable, since starting from a perfectly operating system, only "slow" failures can take place. Hence, in early epochs, the uniformization rate will be much lower than the highest overall rate, thus requiring less jumps. Unfortunately, this gain is not free, since changing uniformization rates makes the computation of the "jump probabilities" (i.e., the probability of $n, n = 0, 1, \ldots,$ jumps until time t) more difficult in AU compared to SU, where the jump probabilities form a Poisson distribution.

Empirical studies [6] show that AU is beneficial if the time of interest is not too large. In this case, AU often requires only a few iterations, where SU would take many iterations, resulting in several orders of magnitude savings in computation time. However, if the time of interest increases, SU eventually becomes the computationally superior method, since the overhead in computing the AU jump probabilities eventually overtakes the computational gain caused by decreasing the number of iterations.

This paper proposes a method that combines AU and SU, such that the number of iterations is almost as low as in AU, and the computation of the jump probabilities is about as efficient as computing Poisson probabilities in SU. The combined approach is based on the observation that all states of the Markov process eventually become and remain reachable in every epoch, and consequently that the uniformization rate in AU will remain the same as that used in SU from that epoch on. We therefore divide the time interval under consideration into two parts: an interval in which with a large probability not all states can be reached, and an interval in which all states can be reached. AU is then applied to the first part, and SU is applied to the second. The "switching time" between the methods is chosen such that the probability that all states are reachable at that time is less than some specified value. Since we show that the switching time can be determined in a computationally efficient way, combined AU/SU outperforms both AU and SU for most models.

Other methods have been proposed to enhance SU (e.g. [2, 11, 20]), and of particular interest are the methods proposed by Malhotra [14, 15], which also combine transient solvers. In [15] the time interval is split into two parts, where SU is applied to the first part, and an ordinary differential equation solver is applied to the second part of the interval. Depending on the problem at hand Malhotra's method or the here presented method can be most suitable, and a future evaluation with regard to issues such as computational complexity, implementation complexity, error bound properties and integration of the method in [15] with combined AU/SU is of interest.

The remainder of the paper is organized as follows. In Section II, we briefly review AU and SU. We then formally define combined AU/SU and discuss how to determine the switching time. In particular, in Section IV, we show that the switching time can be computed very efficiently and accurately by using an approximation method proposed by Yoon and Shanthikumar [22]. We then develop an error bound for the combined method. In particular, we will show in Section V that the computed transient-state probabilities are lower than the actual state probabilities, with an arbitrary small error. We also discuss the influence of computing Poisson probabilities on the error bound when using the Fox/Glynn algorithm [9]. The Fox/Glynn algorithm is enhanced in the Appendix to improve its error bound properties, and the thus obtained results can be used to more precisely determine a bound on the error in any Fox/Glynn-based implementation of SU and AU. Finally, having presented the theoretical properties of combined AU/SU, we consider in Section VI implementation issues and present an application illustrating the merits of combined AU/SU.

Acronyms

CTMC	Continuous-Time Markov Chain
DTMC	Discrete-Time Markov Chain
SU	Standard Uniformization
AU	Adaptive Uniformization
EMR	Extended Machine-Repairman Model

Notation

$Y = \{Y(s); s \geq 0\}$	CTMC defined on state space S
$Q = (q(i,j)), i, j \in S$	Infinitesimal generator matrix for Y
$q_i = -q(i,i), i \in S$	
$\boldsymbol{\pi}(t)$	State probability vector for Y at time t
λ	Uniformization rate in SU
$X = \{X_n, n = 0, 1, \ldots\}$	DTMC after uniformization of Y
P	Transition matrix of X
$\boldsymbol{\pi}(n)$	State probability vector of X at epoch n
L_s, R_s	Left and right truncation point in SU

ϵ_s	Bound on error in SU
A_n	Set of active states in X at epoch n
Q_n, P_n, λ_n	As Q, P and λ above, for epoch n in AU
$U_n(t)$	Jump probabilities in AU ($Pr\{n$ jumps in time $t\}$)
R_a	Right truncation point in AU
ϵ_a	Bound on error in AU
t_s	Switching time from AU to SU
m	Epoch at which uniformization rate converges
λ_m	Converged uniformization rate
$\epsilon_{AU/SU}$	Bound on error in combined AU/SU
δ	Error bound related parameter to determine switching time
$B = \{B(s), s \geq 0\}$,	Birth process with rates $\lambda_0, \ldots, \lambda_m$
$S_B = \{0, 1, \ldots, m+1\}$	State space of B
$Q_B, \lambda_B, P_B, \boldsymbol{\pi}_B$	Defined as for SU, when SU is applied to CTMC B
$P_B(\lambda_B)$	Stresses the dependence of P_B on the chosen λ_B
t_l, t_r	Last two time points for which $Pr\{\geq m$ jumps in AU$\}$ is evaluated
$\epsilon_s^P(n)$	Error bound of computed Poisson probability at iteration n
ϵ_s^a	Error bound for SU when computing jump probabilities in AU
K	Number of components in EMR
r	Repair starts if number of remaining functional components is down to r
γ	Failure rate of components
p	Probability a failure is a hard failure
μ	Repair rate of hard-failed components
ν	Repair rate of soft-failed components
ϵ	Error bound parameter as input to Fox/Glynn algorithm
ϵ_P	Recomputed error bound for truncation in Fox/Glynn algorithm
superscript c	The *computed* value of associated variable (different from *true* value because of computational error)

II Background

Assumptions

1. A system is modeled as a CTMC, with generator matrix Q, and initial state distribution vector $\boldsymbol{\pi}(0)$.

2. We provide an algorithm to compute the transient state distribution function $\boldsymbol{\pi}(t)$ at time t.

Let $Y = \{Y(s); s \geq 0\}$ be a CTMC defined on the finite state space S, and let $Q = (q(i,j)), i, j \in S$, be the infinitesimal generator matrix of Y, where, for notational convenience, we define $q_i = -q(i,i)$ for $i \in S$. Furthermore, let $\boldsymbol{\pi}(0) = (\pi_0(0), \pi_1(0), \ldots)$ denote the initial probability distribution over S. The objective is to obtain the transient-state probability vector $\boldsymbol{\pi}(t)$, with $\pi_i(t) = Pr\{Y(t) = i\}$, $i \in S$.

In SU, the CTMC is decomposed into a discrete-time Markov chain (DTMC) and a Poisson process. Let $\lambda \geq \max\{q_i\}, i \in S$, be the so-called *uniformization rate.* Then, the transition matrix $P = I + (1/\lambda)Q$ defines a DTMC $X = \{X_n, n = 0, 1, \ldots\}$, and $\boldsymbol{\pi}(t)$ is then equal to

$$\boldsymbol{\pi}(t) = \sum_{n=0}^{\infty} poim(n, \lambda t)\boldsymbol{\pi}(n), \tag{1}$$

where $\boldsymbol{\pi}(n) = \boldsymbol{\pi}(n-1)P$, for $n = 1, 2, \ldots,$ and $\boldsymbol{\pi}(n = 0) = \boldsymbol{\pi}(t = 0)$.

In an actual implementation of SU, the transient state probability vector is computed as $\boldsymbol{\pi}^c(t)$, where

$$\boldsymbol{\pi}^c(t) = \sum_{n=L_s}^{R_s} poim^c(n, \lambda t)\boldsymbol{\pi}(n). \tag{2}$$

The inherent computational error introduced in (2) is caused by truncation of the infinite sum and by computation of the Poisson probabilities, and is such that for some pre-selected value ϵ_s:

$$\boldsymbol{\pi}_i(t) - \epsilon_s \leq \boldsymbol{\pi}_i^c(t) \leq \boldsymbol{\pi}_i(t),\ \forall i \in S \ \text{ and } \ \sum_{i \in S} \boldsymbol{\pi}_i^c(t) \geq 1 - \epsilon_s. \tag{3}$$

In AU [19, 21], the uniformization rate can change with the number of jumps considered, and depends on the set of states the DTMC X can be in at the epoch. Formally, the set of *active states* at epoch $n, n = 0, 1, \ldots,$ of X is defined as $A_n \subseteq S$, where

$$A_n = \{i \in S \mid \boldsymbol{\pi}_i(n) > 0\}. \tag{4}$$

Then the "adapted" uniformization rates ("AU rates") are defined as $\lambda_n \geq \max\{q_i \mid i \in A_n\}$, for $n = 0, 1, \ldots$. Next, the "adapted" transition matrices are given by

$$P_n = I + \frac{1}{\lambda_n} Q_n, \quad n = 0, 1, \ldots, \tag{5}$$

where $Q_n = (q_n(i, j)), i, j \in S$, is the generator matrix Q restricted to the active states at each epoch, that is,

$$q_n(i,j) = \begin{cases} q(i,j) & \text{if } i \in A_n \\ 0 & \text{otherwise.} \end{cases}$$

If $U_n(t)$ is defined to be the probability of n jumps in time t in the birth process formed by the AU rates $\lambda_0, \lambda_1, \ldots,$ then the transient-state probability vector can be defined as [21]

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0) \sum_{n=0}^{\infty} U_n(t) \prod_{i=0}^{n-1} P_i = \sum_{n=0}^{\infty} U_n(t)\boldsymbol{\pi}(n), \tag{6}$$

where $\boldsymbol{\pi}(n) = \boldsymbol{\pi}(n-1)P_{n-1}, n = 1, 2, \ldots$, with $\boldsymbol{\pi}(n=0) = \boldsymbol{\pi}(t=0)$. As with SU, the sum is truncated as

$$\boldsymbol{\pi}^c(t) = \sum_{n=0}^{R_a} U_n^c(t)\boldsymbol{\pi}(n), \tag{7}$$

such that

$$\boldsymbol{\pi}_i(t) - \epsilon_a \leq \boldsymbol{\pi}_i^c(t) \leq \boldsymbol{\pi}_i(t),\ \forall i \in S, \ \text{ and } \sum_{i \in S} \boldsymbol{\pi}_i^c(t) \geq 1 - \epsilon_a, \tag{8}$$

for some error bound ϵ_a.

By allowing a more general (than Poisson) birth process, it is possible to make "larger" jumps with AU than SU. Thus, for identical $\epsilon_s = \epsilon_a$, truncation of the infinite sums is expected to be such that $R_a \leq R_s$, potentially resulting in a reduction of computation time. On the other hand, we know from [6] that the computation of $U_n^c(t)$ in (7) is more costly than the computation of the Poisson probabilities in SU, resulting in a large computation time for AU if t is large. The next section introduces the combined AU/SU method, designed to alleviate this problem.

III Combined AU/SU

The objective of the combined approach is to find a method that has jumps as long as those in AU and for which the effort of computing the jump probabilities is as low as in SU. To do this, we observe that in many cases the AU rates in AU no longer change after some number of iterations m (i.e., for some m, $\lambda_i = \lambda_m$ for $i \geq m$), since all states will typically eventually become, and remain, active. We speak in this case of a *converged* uniformization rate λ_m. When the rate converges, AU jumps have become of equal length as those in SU, and there is no further benefit in applying AU.

The basic idea behind combining AU and SU is that computation of $\boldsymbol{\pi}(t)$ can be done in two steps—the computation of $\boldsymbol{\pi}^c(t_s)$ given $\boldsymbol{\pi}(0)$ as the initial distribution, and the computation of $\boldsymbol{\pi}^c(t)$ starting from $\boldsymbol{\pi}^c(t_s)$ at time t_s. This decomposition allows us to apply AU for the first part of the computation, where it is profitable to do so, and apply SU for the remaining part, where it is superior to AU. t_s will be called the *switching time,* and we would like to choose t_s such that the probability that more than m jumps are made in time t_s is small. If this is the case, the infinite sum in AU can be truncated when the rate converges (an obvious modification of this is discussed in Section VI). So, we must determine the maximum time point t_s, such that the likelihood that the converged rate is reached is below some probability δ, which has to be chosen small enough to obtain the desired accuracy (in Equation (22) we will see that δ must be chosen smaller than the desired ϵ_a).

More precisely, we have the following algorithm for combined AU/SU:

Algorithm 1 Combined AU/SU

```
Choose $\delta, \epsilon_a$ and $\epsilon_s$
Find $t_s$ such that
    $\sum_{n=0}^{m} U_n(t_s) \approx \delta$
Compute $\boldsymbol{\pi}^c(t_s)$ using AU with $\boldsymbol{\pi}(0)$ as initial state distribution, i.e.,
    $\boldsymbol{\pi}^c(t_s) = \sum_{n=0}^{m} U_n^c(t_s)\boldsymbol{\pi}(n)$ with $\boldsymbol{\pi}(n=0) = \boldsymbol{\pi}(t=0)$, and
    $\delta$ and $U_n^c(t_s), n = 0, \ldots, m$, such that the error bound is $\epsilon_a$, as in (8)
Compute $\boldsymbol{\pi}^c(t)$ using SU starting from $\boldsymbol{\pi}^c(t_s)$, i.e.,
    $\boldsymbol{\pi}^c(t) = \sum_{n=L_s}^{R_s} poim^c(n, \lambda(t-t_s))\boldsymbol{\pi}(n)$ with $\boldsymbol{\pi}(n=0) = \boldsymbol{\pi}^c(t_s)$ and
    $L_s, R_s$ and $poim^c(n, \lambda t), n = L_s, \ldots, R_s$, such that the error bound is $\epsilon_s$,
    as in (3)
```

In Section V, we will make precise the effect the choice of δ, ϵ_a, and ϵ_s has on the total error (which we denote $\epsilon_{AU/SU}$) incurred in the algorithm, and we compute a bound in terms of the choice of these parameters. Before doing so, however, we discuss how to choose the switching time t_s given δ. Efficient computation of t_s is critical for an efficient implementation of the combined AU/SU algorithm.

IV Switching Time

The value of t_s determines the number of iterations combined AU/SU saves compared to SU and should therefore be determined so that a gain close to the optimum can be expected. On the other hand, the computation of the switching time is the main overhead of combined AU/SU compared with the individual methods, and it is therefore equally important to compute t_s with relatively low computational cost.

Before describing an algorithm to compute the switching time, it is helpful to formally state the problem. The AU rates $\{\lambda_0, \lambda_1, ..., \lambda_m\}$, with λ_m the converged rate, form a pure birth process $B(t)$. So, the AU jump probabilities $U_n(t), n = 0, 1, \ldots,$ in (7) are transient state probabilities of the CTMC $B(t)$, defined on the state space $S_B = \{0, 1, ..., m+1\}$, with infinitesimal generator matrix $Q_B = (q_B(i,j)), i, j \in S_B$, where $q_B(i, i+1) = \lambda_i$, for $i = 0, 1, ..., m$. The diagonal elements of Q_B are defined as usual, and note that the state $i = m+1$ is an absorbing state. In this formulation, identifying the switching time t_s corresponds to finding the time point for which $\pi_{B,m+1}(t_s) = \delta$, starting from $\pi_{B,0}(0) = 1$.

To determine the switching time efficiently, we use an approximation method described by Yoon and Shanthikumar ([22], approximation P_4). The basic idea is that, if λ_B is the uniformization rate and $P_B(\lambda_B) = I + (1/\lambda_B)Q_B$, then

$$\boldsymbol{\pi}_B(t) = \boldsymbol{\pi}_B(0) \lim_{\lambda_B \to \infty} P_B^{\lambda_B t}(\lambda_B). \tag{9}$$

This approximation method is particularly suitable for the problem at hand since the state space S_B of the birth process $B(t)$ equals the number of AU rates, and is therefore usually very small, typically no larger than 100 states. In the actual implementation, the user can set a maximum to the number of AU rates considered, so that there is no risk that the computation of the switching time becomes too expensive. Since S_B is small (and P_B even is upper triangular), fill-in resulting from matrix-matrix products is not a problem, and $P_B^{2^k}(\lambda_B)$ can thus be computed by k matrix-matrix multiplications: $P_B^2(\lambda_B) = P_B(\lambda_B)P_B(\lambda_B)$, $P_B^4(\lambda_B) = P_B^2(\lambda_B)P_B^2(\lambda_B)$, $P_B^8(\lambda_B) = P_B^4(\lambda_B)P_B^4(\lambda_B)$, etc. For any l, the probability that more than m jumps takes place is then given by the element $P^l_{B,(0,m+1)}(\lambda_B)$. Using the result in (9), the time point corresponding to l is $t = l/\lambda_B$. To find the switching time, we thus need to find a value of l such that $P^l_{B,(0,m+1)}(\lambda_B) \approx \delta$, and then $t_s = l/\lambda_B$.

The basic algorithm to do this is given as Algorithm 2. The algorithm straightforwardly computes powers $P_B^{2^k}(\lambda_B)$, $k = 1, 2, \ldots$, until $P^{2^k}_{B,(0,m+1)}(\lambda_B) > \delta$. Then, if $P^{2^k}_{B,(0,m+1)}(\lambda_B) > \epsilon_{AU/SU}$, the desired over all accuracy cannot be reached, and so, the algorithm 'back tracks' and multiplies $P^{2^{k-1}}_{B,(0,m+1)}(\lambda_B)$ with powers of $P_B(\lambda_B)$. This procedure is being repeated until a matrix power l has been found with $\delta \le P^l_{B,(0,m+1)}(\lambda_B) \le \epsilon_{AU/SU}$. The algorithm then exits the While-loop and approximates the switching time by an interpolation.

Algorithm 2 Algorithm for determining switching time

```
        Select $\lambda_B$;
        $P_B = I + \frac{Q_B}{\lambda_B}$;
        $A = P_B$;
        $B = P_B$;
        $t_l = 0$;
        $power = 1$;
        $steppower = 1$;
        $prob = 0$;
        While $(prob < \delta)$ {
            $prob = \sum_{j=0}^{m+1} A(0,j)B(j,m+1)$;

            if $(prob > \epsilon_{AU/SU})$ {
(a)             Adjust_$\lambda_B$;
                $B = P_B$;
                $steppower = 1$;
        }
        $A = A \cdot B$;
        $power = power + steppower$;
        if $(prob < \delta)$ {
            $B = B \cdot B$;
            $steppower = 2 \cdot steppower$;
        }
    }
    $t_l = t_l + (1/\lambda_B) \cdot (power - steppower)$;
    $t_r = t_l + (1/\lambda_B) \cdot power$;
    $t_s =$ interpolate$(t_l, t_r)$;
```

Interpolation is required because Algorithm 2 results in two time points t_l and t_r such that the highest of them gives $\delta \leq \pi_{B,m+1}(t_r) \leq \epsilon_{AU/SU}$, while the lower one gives $\pi_{B,m+1}(t_l) \leq \delta$. Using these two points, the interpolation function approximates the time point t_s for which $\pi_{B,m+1}(t_s) \approx \delta$. Experimental results suggest that an accurate interpolation is possible assuming a linear relationship between the logarithm of the error bound $\pi_{B,m+1}(t)$ and the logarithm of the time point t (the theory of distributions of extremes provides an explanation for this phenomenon [1]). We also note that adjusting λ_B, given t_l, in line (a) can be beneficial, since t_s can be estimated based on the results so far. Given this estimate of t_s one adjusts λ_B to make it more likely that the algorithm gets close to t_s using matrix-matrix multiplication of $P_B(\lambda_B)$. It is important to note that combined AU/SU does not break down if the estimate of t_s is not precise. Independent of the precision in computing t_s, AU can be performed over the interval $[0, t_s]$ and the actually introduced error will be computed along the way.

V Error Bound

One of the attractive properties of both AU and SU is that the computed result is a lower bound of the true result and can be assured to be within any predefined error tolerance ϵ. In this section, we show that a similar result holds for combined AU/SU. Specifically, if ϵ_a is the error bound in the AU part, and ϵ_s in the SU part, then the bound $\epsilon_{AU/SU}$ on the total error in combined AU/SU will be shown to be $\epsilon_{AU/SU} = \epsilon_a + \epsilon_s - \epsilon_a\epsilon_s$. We note that we only consider truncation and normalization errors and do not consider rounding errors caused by the limits of machine accuracy. We refer to Grassmann [12], and references therein, for analysis of round-off errors.

The SU part in combined AU/SU computes $\boldsymbol{\pi}^c(t)$ starting from $\boldsymbol{\pi}^c(t_s)$ computed in the AU part. To derive error bound properties for combined AU/SU, one has to investigate how the error in $\boldsymbol{\pi}^c(t_s)$ (caused by AU) propagates through the SU part to the final result. We first derive the error bound for the sum of all state probabilities, and from this result, the error bound for individual elements follows directly.

Starting with the error bound ϵ_a for $\boldsymbol{\pi}^c(t_s)$, as in (20), it can immediately be seen that at epoch 1 in the SU part

$$\sum_{i\in S}\pi_i^c(1) = \sum_{i\in S}\sum_{j\in S}\pi_j^c(0)p(j,i) = \sum_{j\in S}\pi_j^c(t_s) \geq 1-\epsilon_a. \tag{10}$$

This holds for all iterations in SU, that is, for $n = 0, 1, \cdots, R_s$,

$$\sum_{i\in S}\pi_i^c(n) \geq 1-\epsilon_a.$$

From (2) and (3) it follows that L_s, R_s and $poim^c(n, \lambda(t-t_s))$ are such that $\sum_{n=L_s}^{R_s} poim^c(n, \lambda(t-t_s)) \geq 1-\epsilon_s$. So, we find in the SU part that

$$\sum_{i\in S}\sum_{n=L_s}^{R_s} poim^c(n, \lambda(t-t_s))\pi_i^c(n) \geq \sum_{n=L_s}^{R_s} poim^c(n, \lambda(t-t_s))(1-\epsilon_a) \geq (1-\epsilon_s)(1-\epsilon_a). \tag{11}$$

So, the total error bound $\epsilon_{AU/SU}$ is given by

$$\epsilon_{AU/SU} = 1-(1-\epsilon_s)(1-\epsilon_a) = \epsilon_a + \epsilon_s - \epsilon_a\epsilon_s, \tag{12}$$

and we obtain

$$\pi_i^c(t) \leq \pi_i(t) \leq \pi_i^c(t) + \epsilon_{AU/SU}, \quad \forall i \in S, \quad \text{and} \tag{13}$$

$$\sum_{i\in S}\pi_i^c(t) \geq 1-\epsilon_{AU/SU}. \tag{14}$$

We note that the result in (13) for the individual elements can be derived in a straightforward manner from (14). Using a similar reasoning as in the derivation of (10), it follows from $\pi_i^c(t_s) \leq \pi_i(t_s), \forall i \in S$, that $\pi_i^c(t) \leq \pi_i(t), \forall i \in S$. In words this means that all errors are positive, and thus the individual errors in all states are less than or equal to the summed error $\epsilon_{AU/SU}$ in (14). So, it follows directly from (14) that $\pi_i(t) \leq \pi_i^c(t) + \epsilon_{AU/SU}$ in (13).

The above results for the error bound can be extended to the case that the original time interval is divided into an arbitrary number of N intervals $[0, t_1], [t_1, t_2], ..., [t_{N-1}, t]$, using any uniformization method for the individual subintervals. This is suggested in [13] and has an application in phased-mission reliability models [7]. Note that our derivation for the accumulated error is more precise than the related one given in [4, 13].

Given the error $\epsilon_{AU/SU}$, we need bounds for ϵ_a in AU and ϵ_s in SU, respectively. Since bounds in the literature are some times imprecise in that the computation of the jump/Poisson probabilities is not considered, we derive accurate expressions for ϵ_a and ϵ_s in the next two subsections.

A Expression for ϵ_s

Most recent implementations of SU use an algorithm developed by Fox and Glynn [9] to compute the required Poisson probabilities (see [3, 10] for variations that have been used). In the Fox/Glynn algorithm, the relative values of the needed Poisson probabilities (those between L_s and R_s) are computed exactly, but absolute values of the Poisson probabilities require normalization, thus introducing a "normalization error." In this section we show how to carry out the normalization such that SU gives a lower bound for the true result, and give an expression of the error bound in SU in terms of both the truncation and normalization error.

Let L_s and R_s in (2) be such that

$$\sum_{n=L_s}^{R_s} poim(n, \lambda t) \geq 1 - 0.5\epsilon_s. \tag{15}$$

To guarantee that the computed Poisson probabilities are smaller than the true probabilities, we normalize the computed Poisson probabilities to $1 - 0.5\epsilon_s$, i.e., $\sum_{n=L_s}^{R_s} poim^c(n, \lambda t) = 1 - 0.5\epsilon_s$. To determine the resulting error in SU, we use that the following relation exists between the computed and true Poisson probabilities:

$$poim^c(n, \lambda t) = \frac{poim(n, \lambda t)}{\sum_{k=L_s}^{R_s} poim(k, \lambda t)}(1 - 0.5\epsilon_s). \tag{16}$$

Now, for all iterations $n, n = L_s, \ldots, R_s$, introduce

$$\epsilon_s^P(n) = 0.5\epsilon_s \frac{poim(n, \lambda t)}{\sum_{k=L_s}^{R_s} poim(k, \lambda t)}, \tag{17}$$

and, hence, $\sum_{n=L_s}^{R_s} \epsilon_s^P(n) = 0.5\epsilon_s$. Then it follows from (15), (16) and (17) that at the n-th iteration

$$poim^c(n, \lambda t) \leq poim(n, \lambda t) \leq \frac{poim(n, \lambda t)}{\sum_{k=L_s}^{R_s} poim(k, \lambda t)} = poim^c(n, \lambda t) + \epsilon_s^P(n).$$

So, for all $i \in S$, it follows that

$$\pi_i^c(t) = \sum_{n=L_s}^{R_s} poim^c(n, \lambda t)\pi_i(n) \geq \sum_{n=L_s}^{R_s} (poim(n, \lambda t) - \epsilon_s^P(n))\pi_i(n) \geq \tag{18}$$

$$\sum_{n=L_s}^{R_s} poim(n, \lambda t)\pi_i(n) - \sum_{n=L_s}^{R_s} \epsilon_s^P(n) = \sum_{n=L_s}^{R_s} poim(n, \lambda t)\pi_i(n) - 0.5\epsilon_s \geq \pi_i(t) - \epsilon_s,$$

where the last inequality, which reflects the truncation error, follows from (1) and (15). So, truncation and normalization together result in a error bound ϵ_s, such that

$$\pi_i^c(t) \leq \pi_i(t) \leq \pi_i^c(t) + \epsilon_s. \tag{19}$$

By summing over i in the first equality in (18), we also obtain directly that

$$\sum_{i \in S} \pi_i^c(t) \geq 1 - \epsilon_s. \tag{20}$$

The result computed in this way is a lower bound on the true probabilities and has a true error bound ϵ_s, as in (3). In the Appendix, we show that the normalization error can be made smaller by normalizing the Poisson probabilities to a larger value than $1 - 0.5\epsilon_s$, while still assuring that the final result of SU is a lower bound of the true state probabilities.

B Expression for ϵ_a

In AU, the jump probabilities are best computed using SU [19, 21]. In that case, let ϵ_s^a be the bound on the error in the computation of $U_n(t)$ by means of SU. Then we see from (19) that, for $n = 0, 1, \ldots, m$:

$$U_n^c(t) \leq U_n(t) \leq U_n^c(t) + \epsilon_s^a.$$

Moreover, for the sum of all jump probabilities we have, from (20), that

$$\sum_{n=0}^{m} U_n^c(t) \leq \sum_{n=0}^{m} U_n(t) \leq \sum_{n=0}^{m} U_n^c(t) + \epsilon_s^a.$$

In addition to the error in the jump probabilities, we have to consider the error caused by right truncation at R_a iterations. With $R_a = m$, and with truncation error δ (that is, $\sum_{n=0}^{m} U_n(t) \geq 1 - \delta$), we obtain that

$$\sum_{i \in S} \pi_i^c(t) = \sum_{n=0}^{m} U_n^c(t) \sum_{i \in S} \pi_i(n) = \sum_{n=0}^{m} U_n^c(t) \geq \sum_{n=0}^{m} U_n(t) - \epsilon_s^a \geq 1 - \delta - \epsilon_s^a.$$

Hence, since $\pi_i^c(t) \leq \pi_i(t)$ for all $i \in S$, we have that

$$\pi_i^c(t) \leq \pi_i(t) \leq \pi_i^c(t) + \delta + \epsilon_s^a, \tag{21}$$

and, hence, the error bound ϵ_a in AU is

$$\epsilon_a = \delta + \epsilon_s^a. \tag{22}$$

Note that the δ that has to be chosen in Algorithm 2 is identical to the truncation error δ in this expression. In Algorithm 2 we thus must have that $\delta < \epsilon_a$, for some $\epsilon_a < \epsilon_{AU/SU}$.

In conclusion, the obtained error bound for AU in (22) and for SU in (19) can be plugged into (12) to obtain the bound on the error inherent in combined AU/SU. In the implementation, the different error bound parameters can be tuned such that maximal computational gain is obtained, an issue further discussed in Section VI.B.

VI Practical Considerations and Example

To illustrate the practical considerations and benefits of applying combined AU/SU, we apply the method to an extended machine-repairman model (EMR model), also used in [6, 8] as an example of a computationally intensive Markov model. The results we present have been obtained from the implementation of combined AU/SU in the performance and dependability evaluation software package *UltraSAN* [17]. Using this package, one can specify large Markov models in the form of relatively compact stochastic activity networks [16], for which *UltraSAN* automatically generates the underlying CTMC. We first describe the EMR model and then discuss computational results.

A Extended Machine-Repairman Model

We consider an EMR model with K components, all with the same failure rate, γ. There are two classes of failure, hard (i.e., hard to repair) and soft (i.e., easy to repair), where the probability of a failure being soft is p and of its being hard is $1-p$. Repair is delayed until only r components remain functional. The failure and repair times are all negative exponentially distributed random variables with rates γ (for failures), μ (for repairs of hard-failed components), and ν (for repairs of soft-failed components). Each component has independent repair capability, and repair continues until all units are operational. For highly reliable systems, repair time is small relative to inter-failure time, that is, $\mu \gg \gamma$ and $\nu \gg \gamma$.

The basic model we consider has $K = 200$ components, and repair is initiated when only $r = 100$ components are still functioning. The other parameters for the basic model are $\gamma = 1$, $\nu = 100$, $\mu = 80$, and $p = 0.5$. (When other parameter values are chosen, we will mention these individually.) We are interested in the reliability of the system, with the system considered to have failed if all of the components have failed. The state with all components down is thus taken to be an absorbing state. This reliability model has 25,150 states, and the AU rates when the initial state corresponds to all components functioning are given in Table 1.

It follows from Table 1 that, for the considered model, the AU part (before the converged rate is reached) would take 201 jumps, with rates λ_0 to λ_{200}, and that the uniformization rate in the SU part would be $\lambda = 19{,}901$. However, we do not expect much gain from jumps in the AU part with a rate close to the SU uniformization rate, and we therefore do not consider AU rates in the AU part that are larger than a certain fraction of the SU uniformization rate. This will save computation when computing the switching time, and for the example, we choose only to consider the rates λ_0 to λ_{99}. In the *UltraSAN* implementation, a parameter is present that specifies the fraction of the maximum rate we consider to be converged and hence not perform AU on.

Table 1: Adaptive uniformization rates for the EMR model with $K = 200$ and $r = 100$.

Rate	*Expression*	*Value*
λ_0	$K\gamma$	*200*
λ_1	$(K-1)\gamma$	*199*
λ_2	$(K-2)\gamma$	*198*
$\cdots$	$\cdots$	$\cdots$
λ_{99}	$(r+1)\gamma$	*101*
λ_{100}	$(K-r)\nu + r\gamma$	*10,100*
λ_{101}	$(K-r+1)\nu + (r-1)\gamma$	*10,199*
$\cdots$	$\cdots$	$\cdots$
λ_{199}	$(K-1)\nu + \gamma$	*19,901*
λ_{200}	$(K-2)\nu + \mu + \gamma$	*19,081*
λ_{201}	$(K-1)\nu + \gamma$	*19,901*
λ_{202}	$(K-1)\nu + \gamma$	*19,901*
$\cdots$	$\cdots$	$\cdots$

B Dividing Error Tolerance Between AU and SU

The switching time depends on the value chosen for the error bound parameter δ in Algorithm 2. The value of δ influences the over all error $\epsilon_{AU/SU}$ via $\epsilon_a = \delta + \epsilon_s^a$ (see (22)) and $\epsilon_{AU/SU} = \epsilon_a + \epsilon_s - \epsilon_a\epsilon_s$ (see (12)). Assuming that ϵ_s^a is small relative to δ (the computation of the jump probabilities in AU can be done very accurately without much cost), and neglecting the term $\epsilon_a\epsilon_s$, we see that $\epsilon_a \approx \delta$ and $\epsilon_s \approx \epsilon_{AU/SU} - \delta$. In other words, for given $\epsilon_{AU/SU}$, the choice of δ determines the accuracy needed in the AU part and the SU part, respectively, and it is of interest to see what division of error tolerance if computationally optimal.

The curve in Figure 1 shows the number of iterations in the SU part for different values of the error parameter δ, given that $\epsilon_{AU/SU} = 10^{-6}$. In other words, the horizontal axis corresponds to the error bound ϵ_a in the AU part, and the accuracy in the SU part equals $10^{-6} - \epsilon_a$. The combination of two oppositely operating effects explains the shape of the curve. On the one hand, allowing ϵ_a to increase makes that the switching time increases, which makes that less SU iterations will be necessary. On the other hand, increasing ϵ_a implies a tighter error bound ϵ_s in SU, which makes that more iterations are required in SU. Based on Figure 1, we conclude that $\epsilon_a \approx 0.9 \times \epsilon_{AU/SU}$ is close to optimal. We therefore use the 'allocation factor' of 0.9 to derive the results that follow, meaning that we allocate 90% of the allowable error to the AU part and 10% to the SU part.

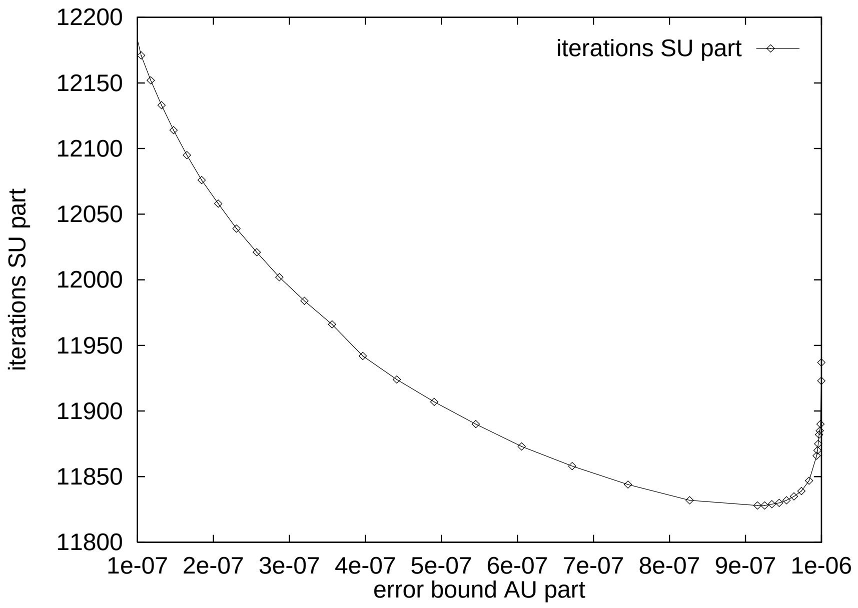

Figure 1: Error bound AU part versus number of iterations in SU part.

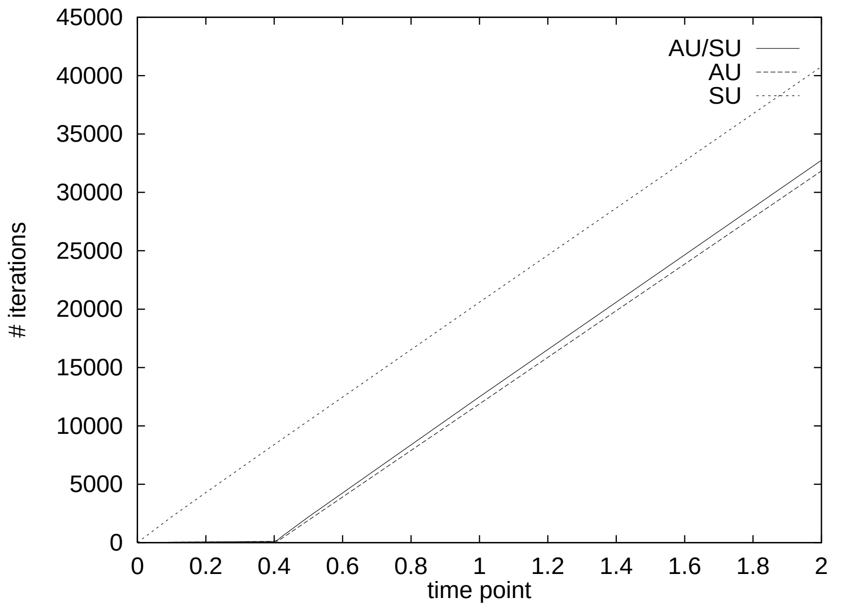

Figure 2: Number of iterations in SU, AU and combined AU/SU.

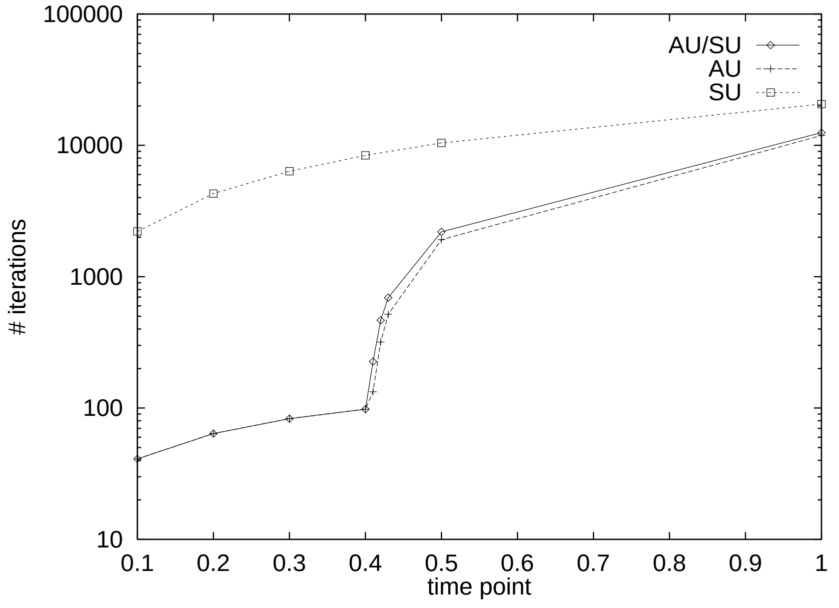

Figure 3: Number of iterations in SU, AU and combined AU/SU for small time values.

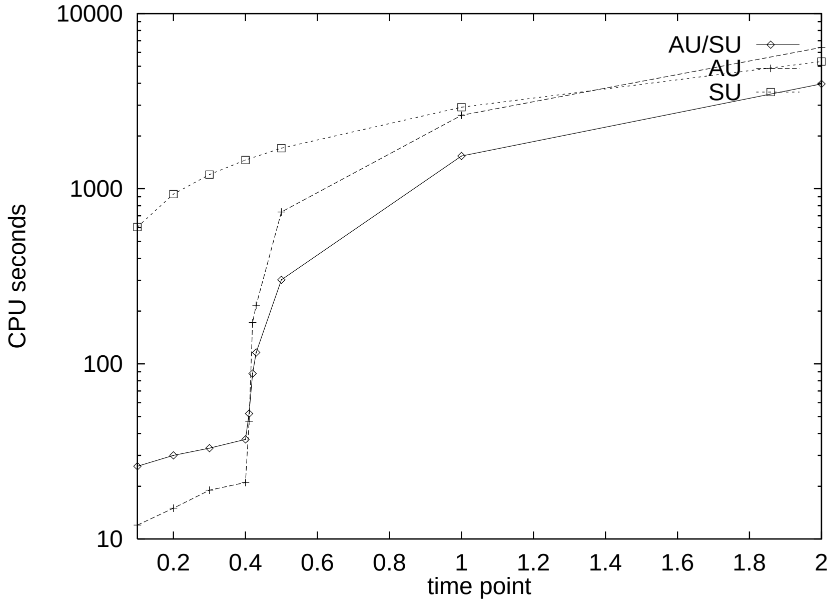

Figure 4: Computation time in SU, AU and combined AU/SU.

C Computational Complexity

For the EMR model with $K = 200$ and $r = 100$, and a total error bounded by $\epsilon_{AU/SU} = 10^{-6}$, the required number of iterations for SU, AU and combined AU/SU is given in Figures 2 and 3, and the computation time is given in Figure 4.

Figure 2 is the only figure in this section with results on a linear scale. It shows that the number of iterations in combined AU/SU is close to the number in AU, while the number in SU is considerably higher. The switching time for the model is $t_s = 0.406$. For values of t less than the switching time, only AU is used, and the number of iterations is at most the number of AU rates (i.e., 100, in the current example). Since the number of iterations in SU is known to be an almost linear function of the product of time t and the uniformization rate λ = 19,901, SU requires about 0.406 $\times$ 19,901 - 101 = 7,980 iterations more than combined AU/SU. This difference between SU and combined AU/SU roughly equals the almost constant amount of saved iterations for increasing t visible in Figure 2.

The number of iterations in AU is slightly less than in combined AU/SU. This is due to two facts. First, the two separate truncation requirements in the AU and SU parts of combined AU/SU are stricter than a single requirement over the whole interval, as in AU. Second, AU will not reach the highest rate until the probability of more than 100 jumps is greater than $\epsilon_{AU/SU}$, while combined AU/SU switches to SU when this probability exceeds $0.9 \times \epsilon_{AU/SU}$. To show this effect, Figure 3 focuses on the smaller time values and shows the number of iterations on a logarithmic scale. The curves show that the number of iterations of combined AU/SU starts increasing for $t > t_s$, while in AU the number of iterations starts increasing from a slightly higher point in time. We conclude however that, compared to AU, the extra number of iterations required in combined AU/SU is minimal.

Figure 4 plots the used CPU time for the example. This and all runs were made on an HP 715/64 workstation, using the implementation of AU, SU, and combined AU/SU made in *UltraSAN*. The times reported are iteration times and do not include the time to read in the matrix from disk, since that is independent of the method used. The computational overhead in combined AU/SU consists of determining the switching time according to Algorithm 2, as well as the computation of the jump probabilities in the AU part of combined AU/SU.

For this example, the overhead of computing the jump probabilities takes only about 2% away from the gain of computing fewer matrix-vector multiplications (MVMs), and we see therefore in Figure 4 for large t a savings of between 15 and 20 CPU minutes. Note that the savings of combined AU/SU compared to SU will be about the same for any $t > t_s$. The next example shown has much greater savings, but the important conclusion to be drawn from Figure 4 is that the overhead of computing the switching time is minor when the AU part consists of about 100 jumps or less.

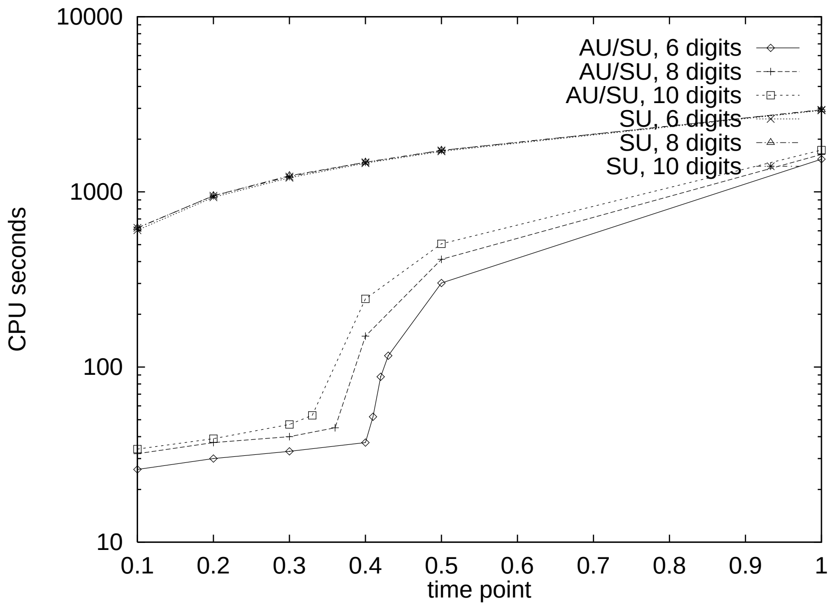

Figure 5: Influence of specified error bound $\epsilon_{AU/SU}$ on computation time.

D Factors Influencing Computation Time

If one applies combined AU/SU to some model, several factors will influence the possible savings obtained with combined AU/SU: the value of the switching time, the cost of computing the switching time, and the cost of the MVM every uniformization iteration. The value of the switching time depends on the AU rates and on the specified error parameters, the cost of computing the switching time largely depends on the size of the matrices on which Algorithm 2 operates, and the cost of the MVMs is mainly determined by the size of the state space. To investigate these factors, we consider several variations of the experiments described in the previous subsection. Figure 5 shows the influence of the specified error bound $\epsilon_{AU/SU}$ on the CPU time consumed. We take values for $\epsilon_{AU/SU}$ of $10^{-6}, 10^{-8}$, and 10^{-10}, while $\epsilon_a = 0.9 \times \epsilon_{AU/SU}$. A higher accuracy causes the switching time to shift to the left, thus reducing the computational gain of combined AU/SU. For $\epsilon_{AU/SU} = 10^{-6}$, the switching time computed with Algorithm 2 is $t_s = 0.406$, for $\epsilon_{AU/SU} = 10^{-8}$, $t_s = 0.367$, and for $\epsilon_{AU/SU} = 10^{-10}$, $t_s = 0.335$. The cost of computing the switching time, which depends on the number of iterations in Algorithm 2, are 20, 23, and 24 CPU seconds, respectively, for these three error bounds. In all cases, the cost of computing the switching time is insignificant compared to the obtained savings in MVMs. We can conclude from Figure 5 that although t_s shifts to the left, combined AU/SU remains worthwhile when high accuracies are needed.

Figure 6 shows the influence of the state-space size on the computational gain, as well as the influence of the number of rates in the AU part on the overhead of computing

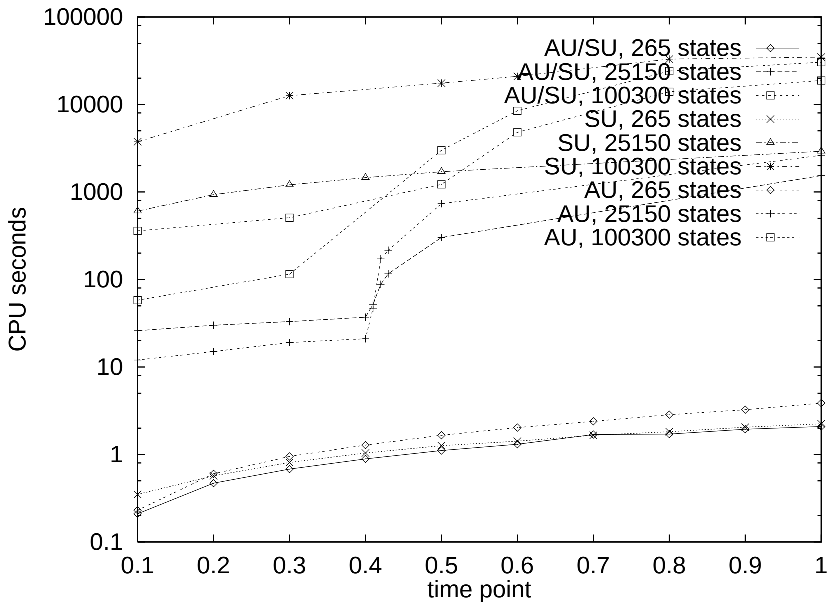

Figure 6: Influence of state-space size on computation time.

the switching time. For three different EMR models, with $K = 20$ and $r = 10$, with $K = 200$ and $r = 100$, and with $K = 400$ and $r = 200$, the CPU time is given, for all three uniformization methods, with the error bounded by 10^{-6}. The size of these models is 265 states, 25,150 states, and 100,300 states, respectively, and the maximum outgoing rate (which equals the SU uniformization rate) of the models is 1,901, 19,901, and 39,901, respectively. The number of considered AU rates are 10, 100, and 200, and the switching times turn out to be equal to $t_s = 0.091$, 0.406, and 0.480, respectively. The overhead of computing the switching time varies from about 0.1 seconds for the smallest model to 20 seconds for the mid-size model and 285 CPU seconds for the largest model.

The main conclusion to be drawn from Figure 6 is that, irrespective of the amount of overhead in computing the switching time, combined AU/SU is superior to SU for all investigated state space sizes and time values. Note that for the largest model the savings are about 5 hours of CPU time. For time points smaller than t_s, AU is superior, but for larger time points, AU becomes far too expensive. Since one does not know the switching time beforehand, combined AU/SU is the more attractive method, especially since it is not much more expensive than AU for small time values. It is also important to note that one usually is interested in a set of time points, in which case the computational complexity is dominated by the largest time point considered. Hence, if the largest considered time point is larger than the switching time, combined AU/SU is the preferred method.

Acknowledgment

In the implementation of the combined AU/SU method, as well as in the discussed example, we have made use of the earlier *UltraSAN* implementation of AU and of the stochastic activity networks created by John Diener [5, 6]. We are also grateful for the careful reviews of the anonymous referees.

Appendix: Accurate Normalization of Poisson Probabilities

Notation In this appendix we propose an enhancement of the Fox/Glynn method for computing Poisson probabilities [9]. The enhancement aims at accurately approximating the truncation error, thus establishing a stricter bound for the error in SU, AU and combined AU/SU.

The algorithm to compute Poisson probabilities as proposed by Fox and Glynn computes left and right truncation points L_s and R_s depending on the specified accuracy ϵ, such that

$$\sum_{n=L_s}^{R_s} poim(n, \lambda t) \geq 1 - \epsilon, \tag{23}$$

and computes "scaled" Poisson probabilities between L_s and R_s. Given the scaled values, a normalization is carried out to approximate the absolute values of the Poisson probabilities. In Fox and Glynn's method normalization to 1 is carried out, but this does not give a lower bound for the transient-state probabilities. In Section V.A, we showed that normalization to $1 - \epsilon$ guarantees that the computed Poisson probabilities are smaller than the actual Poisson probabilities and, when used in SU, give a guaranteed lower bound of 2ϵ on the computed measure. We now show that, given the values of L_s and R_s for some ϵ, it is possible to provide a better estimate of the actual probability mass present between the two truncation points, normalize accordingly, and still obtain a guaranteed lower bound. An earlier version of this normalization approach was developed by Diener [5].

To estimate the sum of the Poisson probabilities between the two truncation points L_s and R_s, we use the following results from Fox and Glynn [9]:

$$poif(L_s - 1, \lambda t) \leq gauf\left(\frac{\lfloor \lambda \rfloor - L_s - 1/2}{\sqrt{\lambda}}\right), \tag{24}$$

and

$$poifc(R_s + 1, \lambda t) \leq gauf\left(\frac{R_s - \lfloor \lambda \rfloor - 1/2}{\sqrt{2\lambda}}\right). \tag{25}$$

To simplify the computation of these terms, the above integrals can themselves be bounded by using that

$$gauf(x) \leq \frac{gaud(x)}{x}, \tag{26}$$

Table 2: Specified error bound $0.5\epsilon_s$, resulting L_s and R_s, and the improved normalization error bound ϵ_P.

$0.5\epsilon_s$	ϵ_P	L_s	R_s
10^{-4}	1.9×10^{-5}	20366	21972
10^{-5}	4.0×10^{-7}	20221	21972
10^{-6}	4.0×10^{-7}	20221	21972
10^{-7}	1.3×10^{-9}	20076	22177
10^{-8}	1.3×10^{-9}	20076	22177
10^{-9}	1.8×10^{-12}	19932	22381
10^{-10}	1.8×10^{-12}	19932	22381
10^{-11}	1.8×10^{-12}	19932	22381
10^{-12}	8.4×10^{-16}	19787	22586

with error less than $gaud(x)/x^3$ for $x > 0$.

We define ϵ_P as the sum of the two bounds given in the right hand side of (24) and (25), and normalize the Poisson probabilities such that

$$\sum_{n=L_s}^{R_s} poim^c(n, \lambda t) = 1 - \epsilon_P. \tag{27}$$

Typically, $\epsilon_P < \epsilon$, but in any case, the minimum of the two can be taken for normalization. From Section V.A we know that the total error in SU constitutes of a truncation error and a normalization error. The normalization error using (27) equals ϵ_P, and we also know that the truncation error will not exceed ϵ_P. Hence, we obtain as total error bound ϵ_s in SU:

$$\epsilon_s = \epsilon_P + \epsilon_P = 2\epsilon_P \leq 2\epsilon. \tag{28}$$

Note that in Section V.A, ϵ was taken to be $\epsilon = 0.5\epsilon_s$. Table 2 illustrates that orders of magnitude improvement on the error bound can be achieved when normalizing to $1 - \epsilon_P$ instead of $1 - 0.5\epsilon_s$. Table 2 should be read as follows. For given $\epsilon = 0.5\epsilon_s$ in column 1, the belonging L_s and R_s from (15) or (23) are given in column 3 and 4. Normalizing to $1 - 0.5\epsilon_s$ would give a total error in SU of ϵ_s, while normalizing to $1 - \epsilon_P$ would give a total error of $2\epsilon_P$, which is typically far smaller.

A final remark is in place about the error introduced if the Poisson probabilities are normalized to 1, as is done in most implementations. In that case, error bound results for SU can be derived in a way similar to the derivation in Section V.A:

$$(1 - \epsilon)\ \boldsymbol{\pi}_i^c(t) \leq \boldsymbol{\pi}_i(t) \leq \boldsymbol{\pi}_i^c(t) + \epsilon, \quad \forall i \in S, \tag{29}$$

if L_s and R_s are such that (23) holds. Similar as in (28), ϵ in (29) can be substituted by ϵ_P, if one is willing to make the effort of computing ϵ_P by the method given above. It is important to note that when normalizing to 1, the computed Poisson probabilities are larger than the actual Poisson probabilities, and the bounds are no longer strict lower bounds, as can be seen directly from (29). This is sometimes an undesirable property, for example, in reliability computation; in that case, the normalization proposed in this Appendix is preferred.

REFERENCES

[1] A. H.-S. Ang and W. H. Tang, *Probability Concepts in Engineering Planning and Design*, volume 2, UC Irvine, UI Urbana-Champaign, 1990.

[2] J. A. Carrasco and A. Calderon, "Regenerative randomization: Theory and application examples," in *Sigmetrics'95/Performance'95*, pp. 241–252, Ottawa, Canada, May 1995.

[3] E. de Souza e Silva and H. R. Gail, "Performability analysis of computer systems: From model specification to solution," *Performance Evaluation*, vol. 14, pp. 157–196, 1992.

[4] E. de Souza e Silva and H. R. Gail, "The uniformization method in performability analysis," in *Second International Workshop on Performability Modelling of Computer and Communication Systems*, Mont Saint-Michel, France, 1993.

[5] J. D. Diener, *Empirical Comparison of Uniformization Methods for Continuous-Time Markov Chains*, Master's thesis, University of Arizona, Tucson, Arizona, 1994.

[6] J. D. Diener and W. H. Sanders, "Empirical comparison of uniformization methods for continuous-time Markov chains," in *Computations with Markov Chains*, W. J. Stewart, editor, chapter 29, pp. 547–570, Kluwer Academic Publishers, Boston, 1995.

[7] J. B. Dugan, "Automated analysis of phased-mission reliability," *IEEE Transactions on Reliability*, vol. 40, no. 1, pp. 45–52, April 1991.

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

[9] B. L. Fox and P. W. Glynn, "Computing Poisson probabilities," *Communications of the ACM*, vol. 31, pp. 440–445, 1988.

[10] W. K. Grassmann, "Means and variances of time averages in Markovian environments," *European Journal of Operational Research*, vol. 31, pp. 132–139, 1987.

[11] W. K. Grassmann, "Finding transient solutions in Markovian event systems through randomization," in *Numerical Solution of Markov Chains*, W. Stewart, editor, pp. 357–371, Marcel Dekker, New York, 1991.

[12] W. K. Grassmann, "Rounding errors in certain algorithms involving Markov chains," *ACM Transactions on Mathematical Software*, vol. 19, no. 4, pp. 496–508, December 1993.

[13] D. Gross and D. R. Miller, "The randomization technique as a modeling tool and solution procedure for transient Markov processes," *Operations Research*, vol. 32, pp. 343–361, 1984.

[14] M. Malhotra, "A computationally efficient technique for transient analysis of repairable Markovian systems," *Performance Evaluation*, vol. 24, no. 4, pp. 311–331, February 1996.

[15] M. Malhotra, "An efficient stiffness-insensitive method for transient analysis of Markovian availability models," *IEEE Transactions on Reliability*, vol. 45, no. 3, pp. 426–428, September 1996.

[16] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Real-Time Systems Symposium*, Austin, TX, December 1984.

[17] W. H. Sanders, W. D. Obal, M. A. Qureshi, and F. K. Widjanarko, "The *UltraSAN* modeling environment," *Performance Evaluation*, vol. 24, no. 1, pp. 89–115, 1995.

[18] H. Tijms, "Computational probability: Old ideas never die," in *Queueing, Performance and Control in ATM (ITC-13)*, J. W. Cohen and C. D. Pack, editors, pp. 255–257, Elsevier Science Publishers B. V. (North-Holland), The Netherlands, 1991.

[19] A. P. A. van Moorsel, *Performability Evaluation Concepts and Techniques*, PhD thesis, University of Twente, The Netherlands, 1993.

[20] A. P. A. van Moorsel and B. R. Haverkort, "Probabilistic evaluation for the analytical solution of large Markov models: Algorithms and tool support," *Microelectronics and Reliability*, vol. 36, no. 6, pp. 733–753, 1996.

[21] A. P. A. van Moorsel and W. H. Sanders, "Adaptive uniformization," *Communications in Statistics - Stochastic Models*, vol. 10, no. 3, pp. 619–647, 1994.

[22] B. S. Yoon and J. G. Shanthikumar, "Bounds and approximations for the transient behavior of continuous-time Markov chains," *Probability in Engineering and Informational Sciences*, vol. 3, pp. 175–198, 1989.