Transient Analysis of Rewarded Continuous Time Markov Models by Regenerative Randomization with Laplace Transform Inversion

Juan A. Carrasco
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
Diagonal 647, plta. 9
08028 Barcelona, Spain

Except for formatting details, this version matches exactly the version published with the same title and author in *Computer Journal*, vol. 46, no. 1, 2003, pp. 84–99, including the clarifications published in the same journal, vol. 51, no. 6, p. 745

Abstract

In this paper we develop a variant, regenerative randomization with Laplace transform inversion, of a previously proposed method (the regenerative randomization method) for the transient analysis of rewarded continuous time Markov models. Those models find applications in dependability and performability analysis of computer and telecommunication systems. The variant differs from regenerative randomization in that the truncated transformed model obtained in that method is solved using a Laplace transform inversion algorithm instead of standard randomization. As regenerative randomization, the variant requires the selection of a regenerative state on which the performance of the method depends. For a class of models, class C', including typical failure/repair models, a natural selection for the regenerative state exists and, with that selection, theoretical results are available assessing the performance of the method in terms of "visible" characteristics. Using dependability class C' models of moderate size of a RAID 5 architecture we compare the performance of the variant with those of regenerative randomization and randomization with steady-state detection for irreducible models, and with those of regenerative randomization and standard randomization for models with absorbing states. For irreducible models, the new variant seems to be about as fast as randomization with steady-state detection for models which are not too small when the initial probability distribution is concentrated in the regenerative state, and significantly faster than regenerative randomization when the model is stiff and not very large. For stiff models with absorbing states, the new variant is much faster than standard randomization and significantly faster than regenerative randomization when the model is not very large. In addition, the variant seems to be able to achieve stringent accuracy levels safely.

Keywords: Rewarded continuous time Markov models, dependability/performability analysis, transient analysis, randomization, Laplace transform.

1 Introduction

Rewarded continuous time Markov chain (CTMC) models find applications in dependability and performability analysis of computer and telecommunication systems. In this paper we will restrict our attention to finite rewarded CTMC models. Let $X = \{X(t); t \geq 0\}$ be a CTMC with finite state space Ω. Then, a rewarded CTMC is obtained by imposing over X a reward rate structure r_i, $i \in \Omega$, where the quantity r_i has the meaning of "rate" at which reward is earned while X is in state i. The behavior of the resulting "reward rate" variable $r_{X(t)}$ can be summarized using several measures. In this paper we will consider two such transient measures: the expected transient reward rate measure, $\mathit{ETRR}(t) = E[r_{X(t)}]$, and the expected averaged reward rate measure, $\mathit{EARR}(t) = E[(\int_0^t r_{X(\tau)}\, d\tau)/t]$. Specific instances of those measures can capture important dependability/performability characteristics of computer and telecommunication systems. Consider, for instance, a fault-tolerant system which can be up or down, whose behavior can be modeled by a CTMC. Assume that a reward rate 1 is assigned to the states of the CTMC in which the system is down and a reward rate 0 is assigned to the states in which the system is up. Then, the $\mathit{ETRR}(t)$ measure would be the unavailability of the system at time t, i.e. the probability that the system is down at time t, and the $\mathit{EARR}(t)$ measure would be the expected interval unavailability of the system at time t, i.e. the expected value of the fraction of time that the system is down in the interval $[0, t]$. As another example, consider a fault-tolerant system modeled by a CTMC X with state space $\Omega = S \cup \{f\}$, where S includes the states in which the system is up and entry in the absorbing state f models system failure. Then, if we assign a reward rate 0 to the states in S and a reward rate 1 to the state f, the $\mathit{ETRR}(t)$ measure would be the unreliability of the system at time t, i.e. the probability that the system will have failed by time t. Reward rates may also represent "performance rates" of a degradable fault-tolerant system. In that case, the $\mathit{ETRR}(t)$ measure would be the expected performance rate of the system at time t and the $\mathit{EARR}(t)$ measure would be the expected averaged performance rate of the system in the time interval $[0, t]$. We will assume $r_i \geq 0$. This is not a true limitation, since, if that condition is not satisfied, it is enough to add a positive value β to the reward rates so that the condition is satisfied, compute the $\mathit{ETRR}(t)$ or $\mathit{EARR}(t)$ measure of the model with shifted reward rates, and subtract β from that value to obtain the $\mathit{ETRR}(t)$ or $\mathit{EARR}(t)$ measure of the original model. We also want to point out that impulse rewards $r_{i,j}$ associated with transitions of the CTMC, which are earned each time the CTMC makes a transition (i, j), can be accommodated by adding quantities $\lambda_{i,j} r_{i,j}$ to the reward rates r_i, where $\lambda_{i,j}$ denotes the transition rate from state i to state j. In that case, $\mathit{ETRR}(t)$ would be $\lim_{\Delta t \to 0} E[\text{reward accumulated in } [t, t + \Delta t]]/\Delta t$ and $\mathit{EARR}(t)$ would be $E[\text{reward accumulated in } [0, t]/t]$.

Computation of the measures $\mathit{ETRR}(t)$ and $\mathit{EARR}(t)$ requires the transient analysis of the CTMC X. Such an analysis can be costly, making the development of efficient computational procedures for those measures a research topic of great interest. Commonly used methods are ordinary differential equation (ODE) solvers and randomization. Good recent reviews of these methods with new results can be found in [1, 2, 3]. The randomization method (also called uniformization) is attractive because of its excellent numerical stability and the facts that the computation error is

well-controlled and can be specified in advance[1]. It was first proposed by Grassman [5] and has been further developed by Gross and Miller [6]. The method is also offered by well-known performance, dependability and performability modeling packages [7, 8, 9, 10]. A major drawback of the (standard) randomization method is that it is expensive when the model is stiff. For CTMCs, a practical measure of stiffness is $\max_{i\in\Omega} \lambda_i t$ [3], where λ_i are the output rates of the states of the CTMC. The high computational cost of the standard randomization method for stiff models has motivated the development in the last years of several variants which can outperform the standard randomization method: selective randomization [11, 12], multistepping [3, Section 3.1.2], adaptive uniformization [13], adaptive/standard uniformization [14], uniformization with steady-state detection [1, 15] and regenerative randomization [16, 17]. Randomization-based methods computing bounds which can be much more efficient than "exact" methods have also been proposed [18].

The regenerative randomization method allows one to compute the *ETRR*(t) and *EARR*(t) measures for rewarded CTMC models X with state space $\Omega = S \cup \{f_1, f_2, \ldots, f_A\}$, $|S| \geq 2$, $A \geq 0$, where f_i are absorbing states and, either (a) all states in S are transient, or (b) S has a single trapping component[2] and the chosen regenerative state $r \in S$ belongs to that component, with some non-null transition rate from r to $S' = S - \{r\}$[3]. It is also assumed that all states are reachable from some state with non-null initial probability. Note that irreducible models correspond to the case in which $A = 0$, S has a single trapping component and all states in S belong to that trapping component. The method has the same good properties as standard randomization (numerical stability, well-controlled computation error and ability to specify the computation error in advance) and for large stiff models can be much faster than standard randomization. The method requires the selection of a regenerative state $r \in S$ and its performance depends on that selection. In the method, a truncated transformed model yielding an approximate value with bounded error for the measures *ETRR*(t) and *EARR*(t) is constructed and that approximate value is computed by solving the truncated transformed model by standard randomization. For a class of models, class C', a natural selection for the regenerative state exists and, with that selection, theoretical results are available assessing the performance of the method in terms of "visible" model characteristics.

Intuitively, the regenerative randomization method is based on the characterization of the behavior of the model up to state r and from r till next hit of r. A similar idea has been exploited in [19] to develop a method which reduces dramatically the computation cost of mean-time-to-failure($MTTF$)-like measures for CTMC models with absorbing states.

[1]The computation error has two components: truncation error and round-off error; the truncation error can be made arbitrarily small, the round-off error will have a very small relative value due to the numerical stability of the method if double precision is used. Rigorous bounds for the relative round-off errors have been obtained in [4] under certain conditions concerning the values that transition rates can have and assuming a special method for computing Poisson probabilities.

[2]Two states i, j of a CTMC are strongly connected if there are paths in the state transition diagram of the CTMC from i to j and from j to i; a state is strongly connected with itself; a component is a maximal subset of strongly connected states; a component is trapping if no state of the component has transition rates to states outside the component.

[3]The last condition can be easily circumvented in practice by adding a tiny transition rate $\lambda = 10^{-10}\epsilon/(2r_{\max}t_{\max})$, where ϵ is the absolute error with which the measure has to be computed, $r_{\max} = \max_{i\in\Omega} r_i$ and $t_{\max}$ is the largest time at which the measure has to be computed, from r to some state in S' in case X has no such transition rate, with a negligible impact on both *ETRR*(t) and *EARR*$(t) \leq 10^{-10}\epsilon(t/t_{\max})$, $t \leq t_{\max}$.

The truncated transformed model obtained in regenerative randomization is as stiff as the original model and its solution by standard randomization can be relatively costly when the original model is stiff and not very large. Based on that observation, in this paper we develop a variant of the method, *regenerative randomization with Laplace transform inversion*, which differs from regenerative randomization in that the truncated transformed model is solved using a Laplace transform inversion algorithm. That alternative tends to be significantly more efficient than standard randomization when the model is stiff and results in a method which can be much more efficient than regenerative randomization for stiff models which are not very large. The price paid is to degrade somehow the quality of the numerical solution, since Laplace transform inversion algorithms provide a series converging to the solution instead of a solution with error bounds, as provided by standard randomization. Our experiments seem to indicate, however, that the variant is numerically stable and able to achieve stringent accuracy levels safely. The rest of the paper is organized as follows. Section 2 reviews briefly the standard and regenerative randomization methods for the computation of the measures *ETRR*(t) and *EARR*(t). Section 3 describes the new variant. Section 4 compares the performance of the variant with those of standard randomization, regenerative randomization and randomization with steady-state detection using class C' dependability models of moderate size of a RAID 5 architecture, and analyzes the accuracy of the variant. Section 5 puts in perspective preliminary work related to the regenerative randomization method and the variant proposed here. Section 6 concludes the paper.

2 Review of Standard and Regenerative Randomization

The standard randomization method is based on the *randomization result*. Let $\lambda_{i,j}$, $i,j \in \Omega$, $i \neq j$, $\lambda_i = \sum_{j \in \Omega - \{i\}} \lambda_{i,j}$, $i \in \Omega$, and $\mathbf{A} = (a_{i,j})_{i,j \in \Omega}$, $a_{i,j} = \lambda_{i,j}$, $i \neq j$, $a_{i,i} = -\lambda_i$ be, respectively, the transition rates, output rates and infinitesimal generator of X. Let $\boldsymbol{\alpha} = (\alpha_i)_{i \in \Omega}$, $\alpha_i = P[X(0) = i]$ be the initial probability row vector of X. Let $\Lambda \geq \max_{i \in \Omega} \lambda_i$ and consider the discrete time Markov chain (DTMC) $\widehat{X} = \{\widehat{X}_k;\ k = 0, 1, 2, \dots\}$ with same state space and initial probability distribution as X and transition probability matrix $\mathbf{P} = \mathbf{I} + \mathbf{A}/\Lambda$, where $\mathbf{I}$ denotes an identity matrix of appropriate dimension. $\widehat{X}$ is said to be the randomized DTMC of X with randomization rate Λ. Let $Q = \{Q(t); t \geq 0\}$ be a Poisson process with arrival rate Λ independent of $\widehat{X}$ ($P[Q(t) = k] = e^{-\Lambda t}(\Lambda t)^k/k!$). Then, the randomization result [20, Theorem 4.19] establishes that X is probabilistically identical to $\{\widehat{X}_{Q(t)}; t \geq 0\}$, which means that the probability of any event defined over the values of the random variables $X(t)$ is the same as the probability of the corresponding event defined over the random variables $\widehat{X}_{Q(t)}$.

Using the randomization result and the independence of Q from $\widehat{X}$, the transient regime of X can be expressed in terms of the transient regime of $\widehat{X}$ as

$$P[X(t) = i] = \sum_{k=0}^{\infty} P[\widehat{X}_k = i \mid Q(t) = k] P[Q(t) = k] = \sum_{k=0}^{\infty} P[\widehat{X}_k = i] e^{-\Lambda t} \frac{(\Lambda t)^k}{k!}.$$

Then, we have

$$\mathit{ETRR}(t) = \sum_{i\in\Omega} r_i P[X(t)=i] = \sum_{i\in\Omega} r_i \sum_{k=0}^{\infty} P[\widehat{X}_k = i] e^{-\Lambda t}\frac{(\Lambda t)^k}{k!} = \sum_{k=0}^{\infty} d(k) e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,, \quad (1)$$

with

$$d(k) = \sum_{i\in\Omega} r_i P[\widehat{X}_k = i]\,. \quad (2)$$

In the standard randomization method for $\mathit{ETRR}(t)$, the summation in (1) is truncated by the right to obtain an approximate value for $\mathit{ETRR}(t)$,

$$\mathit{ETRR}^a_N(t) = \sum_{k=0}^{N} d(k) e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,.$$

The quantities $d(k)$, $k = 0, 1, \ldots, N$ are obtained from the probability row vectors of $\widehat{X}$ at steps k, $\mathbf{q}(k) = (P[\widehat{X}_k = i])_{i\in\Omega}$, using (2). Those probability row vectors can be obtained using $\mathbf{q}(0) = \boldsymbol{\alpha}$ and

$$\mathbf{q}(k+1) = \mathbf{q}(k)\mathbf{P}\,.$$

Usually, the truncation point is selected by controlling the absolute truncation error, $\sum_{k=N+1}^{\infty} d(k) e^{-\Lambda t}(\Lambda t)^k/k!$. Let $r_{\max} = \max_{i\in\Omega} r_i$. Since $d(k) \le r_{\max}$, the absolute truncation error is upper bounded by $r_{\max}\sum_{k=N+1}^{\infty} e^{-\Lambda t}(\Lambda t)^k/k!$, and N is chosen as

$$N = \min\left\{ m \ge 0 : r_{\max} \sum_{k=m+1}^{\infty} e^{-\Lambda t}\frac{(\Lambda t)^k}{k!} \le \epsilon \right\},$$

where ϵ is the allowed absolute error. The truncation error upper bound increases with t and, if $\mathit{ETRR}(t)$ has to be computed for several values of t, it is enough to control the truncation error for the largest of them.

Using $\mathit{EARR}(t) = (\int_0^t \mathit{ETRR}(\tau)\,d\tau)/t$, it is possible to obtain from (1)

$$\mathit{EARR}(t) = \frac{1}{\Lambda t}\sum_{k=0}^{\infty} d(k) \sum_{l=k+1}^{\infty} e^{-\Lambda t}\frac{(\Lambda t)^l}{l!}\,.$$

An appropriate truncation of the summatories gives the following approximate value, with error upper bounded by $r_{\max}\sum_{k=N+1}^{\infty} e^{-\Lambda t}(\Lambda t)^k/k!$:

$$\mathit{EARR}^a_N(t) = \frac{1}{\Lambda t}\sum_{k=1}^{N+1} e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\sum_{l=0}^{k-1} d(l)\,.$$

This allows an implementation of the standard randomization method for the measure $\mathit{EARR}(t)$ with control of the absolute truncation error analogous to the error control in the previously described implementation of the standard randomization method for the measure $\mathit{ETRR}(t)$. Slightly different implementations of the standard randomization method for $\mathit{EARR}(t)$ with different truncations are also possible [21].

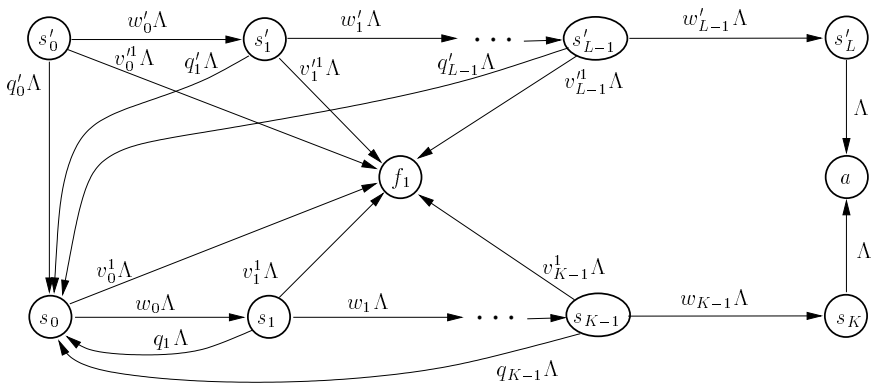

Figure 1: State transition diagram of the CTMC $V_{K,L}$ for the case $A = 1$.

The computational cost of the standard randomization method increases with the truncation point N, which increases with Λ and, for that reason, the best choice for Λ is $\Lambda = \max_{i\in\Omega} \lambda_i$. Using the well-known result [22] that $Q(t)$ has for $\Lambda t \to \infty$ an asymptotic normal distribution with mean and variance Λt, it is easy to realize that, for large Λt, the required N will be $\approx \Lambda t$, and, then, the method will be expensive for stiff large models. Regenerative randomization [16, 17] is a recently developed method for the computation of the measures *ETRR*(t) and *EARR*(t) which has the same good properties as standard randomization and can be much more efficient in those cases. We will briefly review next the method. We will use the notation $\alpha_B = \sum_{i\in B} \alpha_i$. Also, $P_{i,j}$ being the transition probabilities of $\widehat{X}$ (elements of the transition probability matrix $\mathbf{P}$), we will let $P_{i,B} = \sum_{j\in B} P_{i,j}$, $i \in \Omega$, $B \subset \Omega$. In addition, it will be assumed that the randomization rate Λ is slightly larger than $\max_{i\in\Omega} \lambda_i = \max_{i\in S} \lambda_i$, say $\Lambda = (1+\theta) \max_{i\in S} \lambda_i$, $\theta = 10^{-4}$.

The regenerative randomization method includes two phases. In the first phase, a truncated transformed model is built which, with bounded error, yields the same measures *ETRR*(t) and *EARR*(t) as the original model. In the second phase, the truncated transformed model is solved using standard randomization. For the case $\alpha_{S'} > 0$, the truncated transformed CTMC model is called $V_{K,L}$ and has state space $\{s_k, 0 \le k \le K\} \cup \{s'_k, 0 \le k \le L\} \cup \{f_1, f_2, \ldots, f_A, a\}$, initial probability distribution $P[V_{K,L}(0) = s_0] = \alpha_r$, $P[V_{K,L} = s'_0] = \alpha_{S'}$, $P[V_{K,L}(0) = f_i] = \alpha_{f_i}$, $P[V_{K,L}(0) = i] = 0$, $i \notin \{s_0, s'_0, f_1, f_2, \ldots, f_A\}$, the state transition diagram illustrated in Figure 1 for the case $A = 1$ and a reward rate structure $r'_{s_k} = b(k)$, $r'_{s'_k} = b'(k)$, $r'_{f_i} = r_{f_i}$, $r'_a = 0$. For the case $\alpha_{S'} = 0$, the truncated transformed model is called V_K and only differs from $V_{K,L}$ in the absence of the states s'_k.

Let the row vectors $\boldsymbol{\pi}(k) = (\pi_i(k))_{i\in S}$, $\pi_i(k) = P[\widehat{X}'_1 \ne r \wedge \widehat{X}'_2 \ne r \wedge \cdots \wedge \widehat{X}'_k \ne r \wedge \widehat{X}'_k = i]$ and $\boldsymbol{\pi}'(k) = (\pi'_i(k))_{i\in S'}$, $\pi'_i(k) = P[\widehat{X}_0 \ne r \wedge \widehat{X}_1 \ne r \wedge \cdots \wedge \widehat{X}_k \ne r \wedge \widehat{X}_k = i]$, where $\widehat{X}'$ is a version of $\widehat{X}$ with initial state r. Then, the parameters w_k, q_k, v^j_k, w'_k, q'_k, v'^j_k, $b(k)$ and $b'(k)$ on which $V_{K,L}$ and V_K depend can be obtained from $\boldsymbol{\pi}(k)$, $k = 0, 1, \ldots, K$ and $\boldsymbol{\pi}'(k)$, $k = 0, 1, \ldots, L$ using

$$w_k = \frac{\sum_{i\in S} \pi_i(k) P_{i,S'}}{a(k)},$$

$$q_k = \frac{\sum_{i\in S}\pi_i(k)P_{i,r}}{a(k)}\,,$$

$$v_k^j = \frac{\sum_{i\in S}\pi_i(k)P_{i,f_j}}{a(k)}\,,$$

$$b(k) = \frac{\sum_{i\in S}\pi_i(k)r_i}{a(k)}\,,$$

$$w'_k = \frac{\sum_{i\in S'}\pi'_i(k)P_{i,S'}}{a'(k)}\,,$$

$$q'_k = \frac{\sum_{i\in S'}\pi'_i(k)P_{i,r}}{a'(k)}\,,$$

$$v'^j_k = \frac{\sum_{i\in S'}\pi'_i(k)P_{i,f_j}}{a'(k)}\,,$$

$$b'(k) = \frac{\sum_{i\in S'}\pi'_i(k)r_i}{a'(k)}\,,$$

where $a(k) = \sum_{i\in S}\pi_i(k)$ and $a'(k) = \sum_{i\in S'}\pi'_i(k)$.

The row vectors $\boldsymbol{\pi}(k)$ and $\boldsymbol{\pi}'(k)$ required to obtain the truncated transformed model can be obtained as follows. First, we have $\boldsymbol{\pi}(0) = (I_{i=r})_{i\in S}$ and $\boldsymbol{\pi}'(0) = (\alpha_i)_{i\in S'}$, where I_c denotes the indicator function returning the value 1 if condition c is satisfied and the value 0 otherwise. Let $\mathbf{P}_Z$ be the matrix obtained from the restriction of $\mathbf{P}$ to $S \times S$ by setting the elements in the column associated with state r to 0 and let $\mathbf{P}'_Z$ be the restriction of $\mathbf{P}$ to $S' \times S'$[4]. The row vectors $\boldsymbol{\pi}(k)$, $1 \le k \le K$, and $\boldsymbol{\pi}'(k)$, $1 \le k \le L$, can be obtained recursively from $\boldsymbol{\pi}(0)$ and $\boldsymbol{\pi}'(0)$ using $\boldsymbol{\pi}(k+1) = \boldsymbol{\pi}(k)\mathbf{P}_Z$ and $\boldsymbol{\pi}'(k+1) = \boldsymbol{\pi}'(k)\mathbf{P}_{Z'}$.

As we have said, the truncated transformed model has with some bounded error the same expected transient reward rate and expected averaged reward rate as the original model X. More specifically, for the case $\alpha_{S'} > 0$, calling $\mathit{ETRR}^{K,L,a}(t)$ and $\mathit{EARR}^{K,L,a}(t)$, respectively, the expected transient reward rate and expected averaged reward rate measure of $V_{K,L}$, we have:

$$\begin{aligned}\mathit{ETRR}(t) - \mathit{ETRR}^{K,L,a}(t) \le\ & r_{\max}a'(L)\sum_{k=L+1}^{\infty} e^{-\Lambda t}\frac{(\Lambda t)^k}{k!} \\ & +\ r_{\max}\alpha_S a(K)\sum_{k=K+1}^{\infty}(k-K)\,e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,,\end{aligned} \tag{3}$$

$$\begin{aligned}\mathit{EARR}(t) - \mathit{EARR}^{K,L,a}(t) \le\ & \frac{r_{\max}a'(L)}{\Lambda t}\sum_{k=L+2}^{\infty}(k-L-1)\,e^{-\Lambda t}\frac{(\Lambda t)^k}{k!} \\ & +\ \frac{r_{\max}\alpha_S a(K)}{\Lambda t}\sum_{k=K+2}^{\infty}\frac{(k-K)(k-K-1)}{2}\,e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,,\end{aligned} \tag{4}$$

[4]The notation comes from the fact that $\mathbf{P}_Z$ and $\mathbf{P}_{Z'}$ are the restrictions to the subset of transient states of the transition probability matrices of, respectively, the DTMCs Z and Z' considered in [16].

and, for the case $\alpha_{S'} = 0$, calling $\mathit{ETRR}^{K,a}(t)$ and $\mathit{EARR}^{K,a}(t)$, respectively, the expected transient reward rate and expected averaged reward rate measures of V_K:

$$\mathit{ETRR}(t) - \mathit{ETRR}^{K,a}(t) \leq r_{\max}\alpha_S a(K) \sum_{k=K+1}^{\infty} (k-K)\, e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,, \tag{5}$$

$$\mathit{EARR}(t) - \mathit{EARR}^{K,a}(t) \leq \frac{r_{\max}\alpha_S a(K)}{\Lambda t} \sum_{k=K+2}^{\infty} \frac{(k-K)(k-K-1)}{2}\, e^{-\Lambda t}\frac{(\Lambda t)^k}{k!}\,. \tag{6}$$

The model truncation error bounds given by (3)–(6) decrease with the truncation parameters K and L and can be made arbitrarily small by taking K and L large enough.

To clarify the regenerative randomization method, Figure 2 gives a C-like algorithmic description of the method for the measure $\mathit{ETRR}(t)$. The algorithm has as inputs the CTMC X, the number A of absorbing states f_i, the reward rates r_i, $i \in \Omega$, an initial probability distribution row vector $\boldsymbol{\alpha} = (\alpha_i)_{i\in\Omega}$, the regenerative state r, the allowed error ϵ, the number of time points n at which estimates for the measures have to be computed and the time points, $t_1, t_2, \ldots, t_n$. The algorithm has as outputs the estimates for the measure at the time points t_i. Of the allowed error, ϵ, a portion $\epsilon/2$ is allocated for the error associated with the truncation of the transformed model and a portion $\epsilon/2$ is allocated for the error associated with the solution of the truncated transformed model by standard randomization. Since the upper bounds for the model truncation error given by (3) and (5) increase with t, that error is controlled for $t_{\max} = \max\{t_1, t_2, \ldots, t_n\}$. For the case $\alpha_{S'} > 0$, the allocation $\epsilon/2$ for the model truncation error is divided equally between the two contributions of the model truncation error bound. Solution of the truncated transformed model by standard randomization involves stepping the randomized DTMC $\widehat{V}_{K,L}$ ($\widehat{V}_K$) of $V_{K,L}$ (V_K) with randomization rate Λ. Figure 3 shows the state transition diagram of $\widehat{V}_{K,L}$ for the case $A = 1$. The state transition diagram of $\widehat{V}_K$ is identical to the state transition diagram of $\widehat{V}_{K,L}$ but without its upper part, corresponding to states s'_k, $k \geq 0$. The method has a similar implementation for the measure $\mathit{EARR}(t)$. For that measure, the model truncation error bounds also increase with t and those errors are also controlled for $t = t_{\max}$.

Computing the Poisson probabilities $e^{-\Lambda t}(\Lambda t)^k/k!$ is a delicate issue. A numerically stable and reasonably efficient alternative is the method described in [23, pp. 1028–1029] (see also [24]), and that is the method we use in our implementations. Both the standard and regenerative randomization methods require the computation of $S(m) = \sum_{k=m+1}^{\infty} e^{-\Lambda t}(\Lambda t)^k/k!$ for increasing values of m; in addition, the regenerative randomization method requires the computation of $S'(m) = \sum_{k=m+1}^{\infty}(k-m)e^{-\Lambda t}(\Lambda t)^k/k!$ and $S''(m) = \sum_{k=m+2}^{\infty}((k-m)(k-m-1)/2)e^{-\Lambda t}(\Lambda t)^k/k!$ for increasing values of m. Efficient and numerically stable algorithms for performing those computations are described in [16, 17].

The computational cost of the first phase of regenerative randomization is roughly proportional to $K+L$, if $\alpha_{S'} > 0$, and to K, if $\alpha_{S'} = 0$, with a cost per step (increase in K or L) which tends to be slightly larger than the cost per step of standard randomization. The computational cost of the second phase of the method is roughly proportional to $K + L$ if $\alpha_{S'} > 0$ and to K if $\alpha_{S'} = 0$ and, for large Λt, is roughly proportional to Λt. It is possible to show that the required K is $O(\log(\Lambda t/\epsilon))$ and, if

Inputs: X, $\widetilde{A}$, $\widetilde{r_i}$, $i \in \Omega$, $\widetilde{\boldsymbol{\alpha}}$, r, ε, n, $t_1, \widetilde{t_2, \ldots}, t_n$
Outputs: $\widetilde{ETRR}(t_1), \widetilde{ETRR}(t_2), \ldots, \widetilde{ETRR}(t_n)$
$r_{\max} = \max_{i \in \Omega} r_i$;
$t_{\max} = \max\{t_1, t_2, \ldots, t_n\}$;
$\Lambda = (1 + 10^{-4}) \max_{i \in S} \lambda_i$;
Obtain $\mathbf{P}$;
for $(i \in S)$ $P_{i,S'} = \sum_{j \in S', P_{i,j} \neq 0} P_{i,j}$; $\alpha_{S'} = \sum_{i \in S'} \alpha_i$; $\alpha_S = \alpha_r + \alpha_{S'}$;
if $(\alpha_{S'} > 0)$ $tol\_K = \varepsilon/4$; else $tol\_K = \varepsilon/2$;
$\boldsymbol{\pi} = (I_{i=r})_{i \in S}$; $a = 1$; $K = 0$;
do {
    for $(j = 1; j \leq A; j{+}{+})$ $v_K^j = \sum_{i \in S, P_{i,f_j} \neq 0} \pi_i P_{i,f_j}/a$;
    $q_K = \sum_{i \in S, P_{i,r} \neq 0} \pi_i P_{i,r}/a$; $w_K = \sum_{i \in S} \pi_i P_{i,S'}/a$; $b(K) = \sum_{i \in S} \pi_i r_i/a$;
    $\boldsymbol{n\pi} = \boldsymbol{\pi}\mathbf{P}_Z$; $\boldsymbol{\pi} = \boldsymbol{n\pi}$;
    $K{+}{+}$;
    $a = \sum_{i \in S} \pi_i$;
}
until $(r_{\max} \alpha_S a \sum_{k=K+1}^{\infty} (k-K) e^{-\Lambda t_{\max}} (\Lambda t_{\max})^k/k! \leq tol\_K)$;
$b(K) = \sum_{i \in S} \pi_i r_i/a$;
if $(\alpha_{S'} > 0)$ {
    $\boldsymbol{\pi}' = (\alpha_i)_{i \in S'}$; $a' = \alpha_{S'}$; $L = 0$;
    do {
        for $(j = 1; j \leq A; j{+}{+})$ $v_L'^j = \sum_{i \in S', P_{i,f_j} \neq 0} \pi_i' P_{i,f_j}/a'$;
        $q_L' = \sum_{i \in S', P_{i,r} \neq 0} \pi_i' P_{i,r}/a'$; $w_L' = \sum_{i \in S'} \pi_i' P_{i,S'}/a'$; $b'(L) = \sum_{i \in S'} \pi_i' r_i/a'$;
        $\boldsymbol{n\pi}' = \boldsymbol{\pi}' \mathbf{P}_{Z'}$; $\boldsymbol{\pi}' = \boldsymbol{n\pi}'$;
        $L{+}{+}$;
        $a' = \sum_{i \in S'} \pi_i'$;
    }
    until $(r_{\max} a' \sum_{k=L+1}^{\infty} e^{-\Lambda t_{\max}} (\Lambda t_{\max})^k/k! \leq \varepsilon/4)$;
    $b'(L) = \sum_{i \in S'} \pi_i' r_i/a'$;
}
$N = \min\{m \geq 0 : r_{\max} \sum_{k=m+1}^{\infty} e^{-\Lambda t_{\max}} (\Lambda t_{\max})^k/k! \leq \varepsilon/2\}$;
if $(\alpha_{S'} > 0)$
    Give N steps to $\widehat{V}_{K,L}$ and compute $d(k) = \sum_{l=0}^{K} b(l) P[(\widehat{V}_{K,L})_k = s_l]$
    $+ \sum_{l=0}^{L} b'(l) P[(\widehat{V}_{K,L})_k = s_l'] + \sum_{i=1}^{A} r_{f_i} P[(\widehat{V}_{K,L})_k = f_i]$, $k = 0, 1, \ldots, N$;
else
    Give N steps to $\widehat{V}_K$ and compute $d(k) = \sum_{l=0}^{K} b(l) P[(\widehat{V}_{K,L})_k = s_l]$
    $+ \sum_{i=1}^{A} r_{f_i} P[(\widehat{V}_K)_k = f_i]$, $k = 0, 1, \ldots, N$;
for $(i = 1; i \leq n; i{+}{+})$
    for $(k = 0, \widetilde{ETRR}(t_i) = 0; k \leq N; k{+}{+})$ $\widetilde{ETRR}(t_i)$ $+{=}$ $d(k) e^{-\Lambda t_i} (\Lambda t_i)^k/k!$;

Figure 2: Algorithmic description of the regenerative randomization method for $ETRR(t)$.

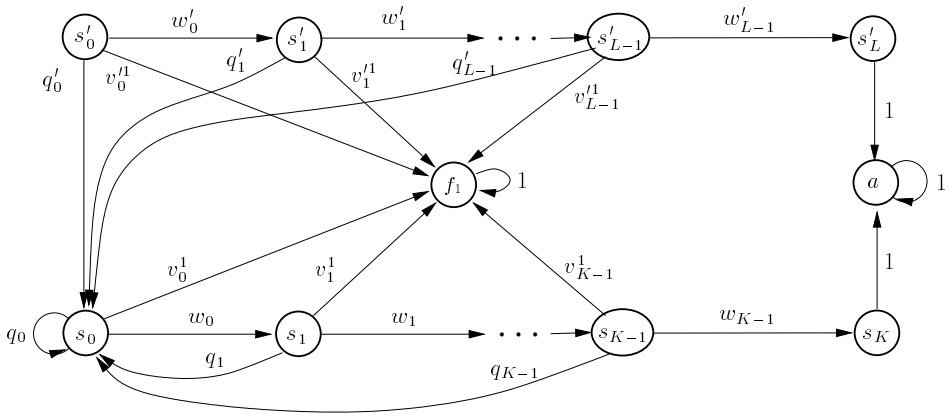

Figure 3: State transition diagram of the CTMC $\widehat{V}_{K,L}$ for the case $A = 1$.

$\alpha_{S'} > 0$, the required L is $O(\log(1/\epsilon))$. This implies that, for large enough and stiff enough models, regenerative randomization will be significantly faster than standard randomization. As previously said, the performance of regenerative randomization depends on the selection of the regenerative state r. That state should be selected so that $a(k)$ and, if $\alpha_{S'} > 0$, $a'(k)$ decrease as fast as possible, and the required K and L are as small as possible. In general, the method relies on user's intuition to select a "good" regenerative state. However, for a class of models, class C', a natural selection for the regenerative state exists and theoretical results are available assessing the performance of the method for that selection in terms of "visible" model characteristics. Using the notation $\lambda_{i,B} = \sum_{j \in B} \lambda_{i,j}$, $B \subset \Omega$, the model class C' includes all CTMCs X covered by regenerative randomization for which a partition $S_0 \cup S_1 \cup \cdots \cup S_{N_C}$ of S exists satisfying the following two properties:

P1. $S_0 = \{o\}$ (i.e. $|S_0| = 1$).

P2. $\max_{0 \le k \le N_C} \max_{i \in S_k} \lambda_{i, S_k - \{i\} \cup S_{k+1} \cup \cdots \cup S_{N_C}}$ is significantly smaller than $\min_{0 < k \le N_C} \min_{i \in S_k} \lambda_{i, S_0 \cup \cdots \cup S_{k-1} \cup \{f_1, \ldots, f_A\}} > 0$.

The class includes failure/repair models with exponential failure and repair time distributions and repair in every state with failed components when failure rates are significantly smaller than repair rates. For those models, a partition of S for which properties P1 and P2 would be satisfied is $S_k = \{\text{states in } S \text{ with } k \text{ failed components}\}$. As an illustration of the model class C', Figure 4 shows a small CTMC reliability model of a repairable fault-tolerant system using the pair-and-spare technique [25] in which active modules have failure rate λ_{M}, the spare module does not fail, the failure of an active module is "soft" with probability S_{M} and "hard" with probability $1 - S_{\mathrm{M}}$ and, whether soft or hard, the failure of an active module is covered with probability C_{M}. Modules in soft failure are independently recovered at rate μ_{S} and modules in hard failure are repaired by a single repairman at rate μ_{H}. A partition for $S = \{1, 2, 3, 4, 5, 6\}$ showing that the model is in class C' is $S = S_0 \cup S_1 \cup S_2$ with $S_0 = \{1\}$, $S_1 = \{2, 3\}$ and $S_2 = \{4, 5, 6\}$.

Since class C' models move fast to either state o or a state f_i, state o is a "natural" selection for the regenerative state for those models and it can be shown [16, 17] that, with that selection,

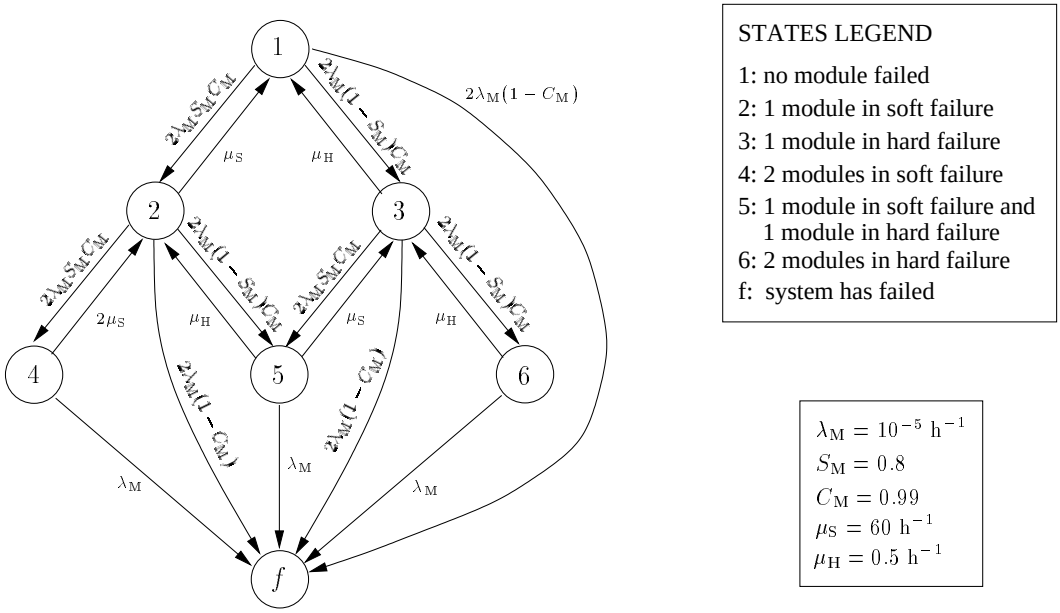

Figure 4: CTMC reliability model of a repairable fault-tolerant system using the pair-and-spare technique.

the required K and L are mainly determined by the parameter $R = \max_{i \in S} \lambda_i / \min_{i \in S'} \lambda_i$: the larger R, the larger the required K and L. In fact, as a rule of thumb, the required K and L can be estimated by $30R$. Since R is a "visible" model characteristic, those rough estimates can be used to anticipate when regenerative randomization can be expected to be significantly faster than standard randomization for class C' models with the selection $r = o$.

3 The New Variant

The truncated transformed model obtained in regenerative randomization has maximum output rate $\Lambda = (1+\theta) \max_{i \in \Omega} \lambda_i$, and, θ being a very small quantity, the truncated transformed model is about as stiff as X. When the original model is stiff, solution of that model by standard randomization will require a very high number of steps on the corresponding randomized DTMC and will be expensive. For large enough X, the truncated transformed model will tend to be much smaller than X and the relative computational cost of the second phase of regenerative randomization will tend to be small. However, when X is not very large, that component can be relatively important and can even dominate the computational cost of the method. This is illustrated by the examples used in Section 4. To improve the computational efficiency of the method in those cases, we propose to use a variant of regenerative randomization, called *regenerative randomization with Laplace transform inversion*, which differs from regenerative randomization only in that the truncated transformed model is solved using a Laplace transform inversion algorithm instead of standard randomization. Many Laplace transform inversion algorithms only require the computation of the Laplace transform of the function to be inverted at several abscissae and the variant uses one of those algorithms. In

Section 3.1 we will derive closed form expressions for the solution of the truncated transformed model in the Laplace transform domain and will describe efficient algorithms for computing those closed form expressions. In Section 3.2 we will describe the Laplace transform inversion algorithm which is used by the variant, paying special attention to error control issues.

3.1 Closed form solution in the Laplace transform domain

The Laplace transform of a function $f(t)$ will be denoted by $\widehat{f}(s)$. We will start by considering the case $\alpha_{S'} > 0$ and deriving closed form expressions for $\widehat{\mathit{ETRR}}^{K,L,a}(s)$ and $\widehat{C}_{K,L}(s)$, where $C_{K,L}(t) = t \times \mathit{EARR}^{K,L,a}(t)$. Then, we will explain how those expressions are simplified for the particular case $\alpha_{S'} = 0$ to obtain closed form expressions for $\widehat{\mathit{ETRR}}^{K,a}(s)$ and $\widehat{C}_K(s)$, where $C_K(t) = t \times \mathit{EARR}^{K,a}(t)$. Finally, based on those closed form expressions, efficient algorithms for computing $\widehat{\mathit{ETRR}}^{K,L,a}(s)$ or $\widehat{\mathit{ETRR}}^{K,a}(s)$ and $\widehat{C}_{K,L}(s)$ or $\widehat{C}_K(s)$ will be given. We will use the notation $p_k(t) = P[V_{K,L}(t) = s_k]$, $p'_k(t) = P[V_{K,L}(t) = s'_k]$, $p_{f_i}(t) = P[V_{K,L}(t) = f_i]$ and $p_a(t) = P[V_{K,L}(t) = a]$.

Taking into account that, because $\sum_{1 \le i \le A} v_k^i + q_k + w_k = 1$, $1 \le k \le K-1$, and $\sum_{1 \le i \le A} v_k'^i + q'_k + w'_k = 1$, $0 \le k \le L-1$, the output rates of the states s_k, $1 \le k \le K$, and s'_k, $0 \le k \le L$, of $V_{K,L}$ are equal to Λ, the transient probabilities of $V_{K,L}$ (see Figure 1) are governed by the following set of equations:

$$\frac{dp_k}{dt} = -\Lambda p_k(t) + w_{k-1}\Lambda p_{k-1}(t), \quad p_k(0) = 0\,, \quad 1 \le k \le K\,,$$

$$\frac{dp'_0}{dt} = -\Lambda p'_0(t)\,, \quad p'_0(0) = \alpha_{S'}\,,$$

$$\frac{dp'_k}{dt} = -\Lambda p'_k(t) + w'_{k-1}\Lambda p'_{k-1}(t), \quad p'_k(0) = 0\,, \quad 1 \le k \le L\,,$$

$$\frac{dp_{f_i}}{dt} = \sum_{k=0}^{K-1} v_k^i \Lambda p_k(t) + \sum_{k=0}^{L-1} v_k'^i \Lambda p'_k(t)\,, \quad p_{f_i}(0) = \alpha_{f_i}\,, \quad 1 \le i \le A\,,$$

$$\frac{dp_a}{dt} = \Lambda p_K(t) + \Lambda p'_L(t)\,, \quad p_a(0) = 0\,,$$

$$1 = \sum_{k=0}^{K} p_k(t) + \sum_{k=0}^{L} p'_k(t) + \sum_{i=1}^{A} p_{f_i}(t) + p_a(t)\,.$$

Taking Laplace transforms we obtain

$$s\widehat{p}_k(s) = -\Lambda\widehat{p}_k(s) + w_{k-1}\Lambda\widehat{p}_{k-1}(s)\,, \quad 1 \le k \le K\,, \tag{7}$$

$$s\widehat{p}'_0(s) - \alpha_{S'} = -\Lambda\widehat{p}'_0(s)\,, \tag{8}$$

$$s\widehat{p}'_k(s) = -\Lambda\widehat{p}'_k(s) + w'_{k-1}\Lambda\widehat{p}'_{k-1}(s)\,, \quad 1 \le k \le L\,, \tag{9}$$

$$s\widehat{p}_{f_i}(s) - \alpha_{f_i} = \sum_{k=0}^{K-1} v_k^i \Lambda\widehat{p}_k(s) + \sum_{k=0}^{L-1} v_k'^i \Lambda\widehat{p}'_k(s)\,, \quad 1 \le i \le A\,, \tag{10}$$

$$s\widehat{p}_a(s) = \Lambda\widehat{p}_K(s) + \Lambda\widehat{p}'_L(s)\,, \tag{11}$$

$$\frac{1}{s} = \sum_{k=0}^{K}\widehat{p}_k(s) + \sum_{k=0}^{L}\widehat{p}'_k(s) + \sum_{i=1}^{A}\widehat{p}_{f_i}(s) + \widehat{p}_a(s)\,. \tag{12}$$

Note that, since, for $k \geq 1$, $\pi_r(k) = 0$, for $k \geq 0$,

$$w_k = \frac{\sum_{i\in S}\pi_i(k)P_{i,S'}}{a(k)} = \frac{\sum_{i\in S'}\pi_i(k+1)}{a(k)} = \frac{\sum_{i\in S}\pi_i(k+1)}{a(k)} = \frac{a(k+1)}{a(k)}\,.$$

Also,

$$w'_k = \frac{\sum_{i\in S'}\pi'_i(k)P_{i,S'}}{a'(k)} = \frac{\sum_{i\in S'}\pi'_i(k+1)}{a'(k)} = \frac{a'(k+1)}{a'(k)}\,.$$

Then, using $a(0) = 1$ and $a'(0) = \alpha_{S'}$, we have

$$\prod_{i=0}^{k-1} w_i = \prod_{i=0}^{k-1}\frac{a(i+1)}{a(i)} = \frac{a(k)}{a(0)} = a(k)\,, \tag{13}$$

$$\prod_{i=0}^{k-1} w'_i = \prod_{i=0}^{k-1}\frac{a'(i+1)}{a'(i)} = \frac{a'(k)}{a'(0)} = \frac{a'(k)}{\alpha_{S'}}\,. \tag{14}$$

Using (13) and (14) we can solve the set of equations (7)–(11) to obtain closed form expressions for $\widehat{p}'_k(s)$, $0 \leq k \leq L$, and express $\widehat{p}_k(s)$, $1 \leq k \leq K$, $\widehat{p}_{f_i}(s)$, $1 \leq i \leq A$, and $\widehat{p}_a(s)$ in terms of $\widehat{p}_0(s)$:

$$\widehat{p}'_k(s) = a'(k)\frac{\Lambda^k}{(s+\Lambda)^{k+1}}\,, \quad 0 \leq k \leq L\,,$$

$$\widehat{p}_k(s) = a(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k\widehat{p}_0(s)\,, \quad 1 \leq k \leq K\,,$$

$$\widehat{p}_{f_i}(s) = \frac{\alpha_{f_i}}{s} + \frac{1}{s}\sum_{k=0}^{K-1} v_k^i a(k)\Lambda\left(\frac{\Lambda}{s+\Lambda}\right)^k\widehat{p}_0(s) + \frac{1}{s}\sum_{k=0}^{L-1} v_k'^i a'(k)\left(\frac{\Lambda}{s+\Lambda}\right)^{k+1}\,, \quad 1 \leq i \leq A\,,$$

$$\widehat{p}_a(s) = \frac{a(K)\Lambda}{s}\left(\frac{\Lambda}{s+\Lambda}\right)^K\widehat{p}_0(s) + \frac{a'(L)}{s}\left(\frac{\Lambda}{s+\Lambda}\right)^{L+1}\,.$$

Plugging those equations in (12) we can solve for $\widehat{p}_0(s)$, obtaining

$$\widehat{p}_0(s) = \frac{A(s)}{B(s)}\,, \tag{15}$$

with

$$\begin{aligned} A(s) \;=\; & 1 - \sum_{i=1}^{A}\alpha_{f_i} - \frac{s}{s+\Lambda}\sum_{k=0}^{L}a'(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k - \frac{\Lambda}{s+\Lambda}\sum_{k=0}^{L-1}\left(\sum_{i=1}^{A}v_k'^i\right)a'(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k \\ & - \; a'(L)\left(\frac{\Lambda}{s+\Lambda}\right)^{L+1} \end{aligned} \tag{16}$$

and

$$B(s) = s\sum_{k=0}^{K} a(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k + \Lambda\sum_{k=0}^{K-1}\left(\sum_{i=1}^{A} v_k^i\right) a(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k + a(K)\Lambda\left(\frac{\Lambda}{s+\Lambda}\right)^K. \quad (17)$$

Let $c(k) = a(k)b(k) = \sum_{i\in S} r_i\pi_i(k)$ and $c'(k) = a'(k)b'(k) = \sum_{i\in S'} r_i\pi'_i(k)$. Then, since $\mathit{ETRR}^{K,L,a}(t) = \sum_{k=0}^{K} b(k)p_k(t) + \sum_{k=0}^{L} b'(k)p'_k(t) + \sum_{i=1}^{A} r_{f_i}p_{f_i}(t)$, $\widehat{\mathit{ETRR}}^{K,L,a}(s)$ can be written as

$$\begin{aligned}\widehat{\mathit{ETRR}}^{K,L,a}(s) &= \sum_{k=0}^{K} b(k)\widehat{p}_k(s) + \sum_{k=0}^{L} b'(k)\widehat{p}'_k(s) + \sum_{i=1}^{A} r_{f_i}\widehat{p}_{f_i}(s)\\ &= \left[\sum_{k=0}^{K} c(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k + \frac{\Lambda}{s}\sum_{k=0}^{K-1}\left(\sum_{i=1}^{A} r_{f_i}v_k^i\right) a(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k\right]\widehat{p}_0(s)\\ &\quad + \frac{\sum_{i=1}^{A} r_{f_i}\alpha_{f_i}}{s} + \sum_{k=0}^{L} c'(k)\frac{\Lambda^k}{(s+\Lambda)^{k+1}} + \sum_{k=0}^{L-1}\left(\sum_{i=1}^{A} r_{f_i}v_k'^i\right) a'(k)\frac{\Lambda^{k+1}}{s(s+\Lambda)^{k+1}}. \quad (18)\end{aligned}$$

Equations (18) and (15)–(17) define a closed form expression for $\widehat{\mathit{ETRR}}^{K,L,a}(s)$. Using $\mathit{EARR}^{K,L,a}(t) = (\int_0^t \mathit{ETRR}^{K,L,a}(\tau)\,d\tau)/t$, we have $C_{K,L}(t) = t \times \mathit{EARR}^{K,L,a}(t) = \int_0^t \mathit{ETRR}^{K,L,a}(\tau)\,d\tau$ and, then,

$$\widehat{C}_{K,L}(s) = \frac{\widehat{\mathit{ETRR}}^{K,L,a}(s)}{s}, \quad (19)$$

which with (18) and (15)–(17) give the closed form expression for $\widehat{C}_{K,L}(s)$.

The previous developments can be easily modified to derive closed form expressions for $\widehat{\mathit{ETRR}}^{K,a}(s)$ and $\widehat{C}_K(s)$. The results are:

$$\widehat{\mathit{ETRR}}^{K,a}(s) = \left[\sum_{k=0}^{K} c(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k + \frac{\Lambda}{s}\sum_{k=0}^{K-1}\left(\sum_{i=1}^{A} r_{f_i}v_k^i\right) a(k)\left(\frac{\Lambda}{s+\Lambda}\right)^k\right]\widehat{p}_0(s) + \frac{\sum_{i=1}^{A} r_{f_i}\alpha_{f_i}}{s}$$

with

$$\widehat{p}_0(s) = \frac{1 - \sum_{i=1}^{A}\alpha_{f_i}}{B(s)},$$

where $B(s)$ is given by (17) and

$$\widehat{C}_K(s) = \frac{\widehat{\mathit{ETRR}}^{K,a}(s)}{s}. \quad (20)$$

Based on the obtained closed form expressions, Figure 5 describes an efficient algorithm to compute $\widehat{\mathit{ETRR}}^{K,L,a}(s)$ or $\widehat{\mathit{ETRR}}^{K,a}(s)$, depending on whether $\alpha_{S'} > 0$ or $\alpha_{S'} = 0$. The algorithm deals efficiently with the case in which $r_i = 0$, $i \in S$, which occurs quite often (for instance, when X is a reliability model), by exploiting the fact that in that case $c(k) = c'(k) = 0$. The algorithm has as inputs $\alpha_{S'}$, a boolean variable *null* which indicates when $r_i = 0$, $i \in S$, K, L, A, $a(k)$, $c(k)$,

$0 \leq k \leq K$, $a'(k)$, $c'(k)$, $0 \leq k \leq L$, v_k^i, $0 \leq k \leq K-1$, $1 \leq i \leq A$, $v_k'^i$, $0 \leq k \leq L-1$, $1 \leq i \leq A$, α_{f_i}, $1 \leq i \leq A$, r_{f_i}, $1 \leq i \leq A$, Λ, and the abscissa s, and leaves the Laplace transform $\widehat{ETRR}^{K,L,a}(s)$ or $\widehat{ETRR}^{K,a}(s)$ in $\widehat{f}(s)$. Based on (19) and (20), $\widehat{C}_{K,L}(s)$ and $\widehat{C}_K(s)$ can be computed efficiently by invoking the algorithm and dividing the value returned by that algorithm by s.

3.2 Laplace transform inversion algorithm

There are many numerical Laplace transform inversion algorithms. From those requiring only the computation of the Laplace transform of the function to be inverted, we have experimented with the methods proposed in [26] and [27]. Both are based on Durbin's approximation for $f(t)$ [28]:

$$f^a(t) = \frac{1}{T}\, e^{at} \left[\frac{\widehat{f}(a)}{2} + \sum_{k=1}^{\infty} \Re \left\{ \widehat{f}\left(a + \frac{ik\pi}{T} \right) e^{\frac{ik\pi t}{T}} \right\} \right] , \tag{21}$$

where $i = \sqrt{-1}$ and $\Re\{x\}$ denotes the real part of a complex x. The approximation error, $f(t) - f^a(t)$, is:

$$f^\epsilon(t) = \sum_{k=1}^{\infty} f(2kT + t) e^{-2akT} . \tag{22}$$

The methods only differ in the value taken for T. T is taken equal to $\mathrm{TMUL} \times t$, with $\mathrm{TMUL} = 1$ in [26] and $\mathrm{TMUL} = 16$ in [27]. The series in (21) tends to exhibit a poor convergence rate and, typically, an acceleration algorithm is used to improve it. The stability but also the cost (number of abscissae at which Laplace transforms have to be computed to reach convergence) increases with TMUL. Thus, we decided to experiment with several choices for TMUL from $\mathrm{TMUL} = 1$ to $\mathrm{TMUL} = 16$. As acceleration algorithm we used the epsilon algorithm described in [29] over the series which results when the terms of the summation of (21) are taken in groups of TMUL terms, i.e.

$$S_m = \frac{1}{T}\, e^{at} \left[\frac{\widehat{f}(a)}{2} + \sum_{k=1}^{m \times \mathrm{TMUL}} \Re \left\{ \widehat{f}\left(a + \frac{ik\pi}{T} \right) e^{\frac{ik\pi t}{T}} \right\} \right] .$$

The accelerated series has an element for every element S_m with odd m. To limit the cost of the epsilon algorithm, which is quadratic with the length of the series S_m, we limited the number of elements of that series to a maximum value of 101 and started to shift the series S_m to the left when m would become greater than 101 so that the maximum value is not overpassed. In addition, to improve the numerical stability of the epsilon algorithm, we used the more robust Brezinski's recursion described in [29] when important cancellations were found in the standard recursion of the epsilon algorithm. The choice $\mathrm{TMUL} = 8$ seemed to be the fastest one giving enough stability, and we decided to use that selection with the epsilon algorithm implemented as previously described. Suumarizing, our Laplace transform inversion algorithm differs from the one implemented in the DLAINV FORTRAN subroutine described in [27] only in that: (1) the discretization error is strictly controlled using special knowledge about $f(t)$ (see later); (2) $\mathrm{TMUL} = 8$ is used instead of $\mathrm{TMUL} = 16$; (3) the series accelerated by the epsilon algorithm is the sifted series S_m previously

Inputs: $\alpha_{S'}$, $null$, K, L, A, $a(k)$, $c(k)$, $a'(k)$, $c'(k)$, v_k^i, $v_k'^i$, α_{f_i}, r_{f_i}, Λ, s
Outputs: $\widehat{f}(s)$
$spL = s + \Lambda$; $sfL = s/spL$; $LfL = \Lambda/spL$; $pLfL = 1$;
if $(\alpha_{S'} > 0)\{$ $np1 = 0$; $np2 = 0$; $it1 = 0$; $it2 = 0$;$\}$
$dp1 = 0$; $dp2 = 0$; $ptf1 = 0$; $ptf2 = 0$;
if $(\alpha_{S'} > 0)$ $M = \min\{K, L\}$; else $M = K$;
for $(k = 0; k \leq M - 1; k{+}{+})\{$
  if $(\alpha_{S'} > 0)\{$
    $aux = a'(k) * pLfL$; $np1 \mathrel{+}= aux$; $np2 \mathrel{+}= (\sum_{i=1}^{A} v_k'^i) * aux$;
    if $(!null)$ $it1 \mathrel{+}= c'(k) * pLfL$; $it2 \mathrel{+}= (\sum_{i=1}^{A} r_{f_i} v_k'^i) * aux$;
  $\}$
  $aux = a(k) * pLfL$; $dp1 \mathrel{+}= aux$; $dp2 \mathrel{+}= (\sum_{i=1}^{A} v_k^i) * aux$;
  if $(!null)$ $ptf1 \mathrel{+}= c(k) * pLfL$; $ptf2 \mathrel{+}= (\sum_{i=1}^{A} r_{f_i} v_k^i) * aux$;
  $pLfL \mathrel{*}= LfL$;
$\}$
if $(\alpha_{S'} == 0)\{$
  $dp1 \mathrel{+}= a(K) * pLfL$; $dp = s * dp1 + \Lambda * dp2 + a(K)\Lambda * pLfL$;
  $p = (1 - \sum_{i=1}^{A} \alpha_{f_i})/dp$; if $(!null)$ $ptf1 \mathrel{+}= c(K) * pLfL$; $ptf = ptf1 + \Lambda/s * ptf2$;
  $it = (\sum_{i=1}^{A} r_{f_i} \alpha_{f_i})/s$;
  $\widehat{f}(s) = it + ptf * p$;
$\}$
else$\{$
  if $(K < L)\{$
    for $(k = K; k \leq L - 1; k{+}{+})\{$
      $aux = a'(k) * pLfL$; $np1 \mathrel{+}= aux$; $np2 \mathrel{+}= (\sum_{i=1}^{A} v_k'^i) * aux$;
      if $(!null)$ $it1 \mathrel{+}= c'(k) * pLfL$; $it2 \mathrel{+}= (\sum_{i=1}^{A} r_{f_i} v_k'^i) * aux$;
      if $(k == K)$ $pLfLK = pLfL$;
      $pLfL \mathrel{*}= LfL$;
    $\}$
    $pLfLL = pLfL$;
  $\}$
  else if $(L < K)\{$
    for $(k = L; k \leq K - 1; k{+}{+})\{$
      $aux = a(k) * pLfL$; $dp1 \mathrel{+}= aux$; $dp2 \mathrel{+}= (\sum_{i=1}^{A} v_k^i) * aux$;
      if $(!null)$ $ptf1 \mathrel{+}= c(k) * pLfL$; $ptf2 \mathrel{+}= (\sum_{i=1}^{A} r_{f_i} v_k^i) * aux$;
      if $(k == L)$ $pLfLL = pLfL$;
      $pLfL \mathrel{*}= LfL$;
    $\}$
    $pLfLK = pLfL$;
  $\}$
  else$\{$
    $pLfLK = pLfL$; $pLfLL = pLfL$;
  $\}$
  $aux = a'(L) * pLfLL$; $np1 \mathrel{+}= aux$; $np = 1 - \sum_{i=1}^{A} \alpha_{f_i} - LfL * aux - sfL * np1 - LfL * np2$;
  $aux = a(K) * pLfLK$; $dp1 \mathrel{+}= aux$; $dp = s * dp1 + \Lambda * dp2 + \Lambda * aux$;
  $p = np/dp$;
  if $(!null)$ $it1 \mathrel{+}= c'(L) * pLfLL$; $it = (\sum_{i=1}^{A} r_{f_i} \alpha_{f_i})/s + it1/spL + LfL/s * it2$;
  if $(!null)$ $ptf1 \mathrel{+}= c(K) * pLfLK$; $ptf = ptf1 + \Lambda/s * ptf2$;
  $\widehat{f}(s) = it + ptf * p$;
$\}$

Figure 5: Efficient algorithm for the computation of $\widehat{ETRR}^{K,L,a}(s)$ and $\widehat{ETRR}^{K,a}(s)$.

described; and (4) Brizinski's recursion is used when important cancellations are detected in the standard implementation of the epsilon algorithm.

The error of the Laplace inversion algorithm is controlled as follows. In the case of inverting $\widehat{ETRR}^{K,L,a}(s)$ or $\widehat{ETRR}^{K,a}(s)$, the error must be $\leq \epsilon/2$. The error has two components: the approximation error and the truncation error (resulting from the truncation of the accelerated series) and we allocate $\epsilon/4$ to each of them. Since $b(k), b'(k), r_{f_i} \leq r_{\max}$, we have $f(t) \leq r_{\max}$ and, consequently (22), assuming $a > 0$, which implies $e^{-2aT} < 1$,

$$f^{\epsilon}(t) \leq \sum_{k=1}^{\infty} r_{\max}\, e^{-2akT} = r_{\max}\, \frac{e^{-2aT}}{1 - e^{-2aT}}\,,$$

and to upper bound the approximation error by $\epsilon/4$ we take the $a > 0$ satisfying

$$r_{\max}\, \frac{e^{-2aT}}{1 - e^{-2aT}} = \frac{\epsilon}{4}\,,$$

i.e.

$$a = \frac{1}{2T} \log\left(1 + \frac{4r_{\max}}{\epsilon}\right)\,.$$

The truncation error is estimated from the absolute difference between the last two elements of the accelerated series. When that series converges slowly, the actual truncation error can be larger than the absolute difference between the last two elements. Furthermore, in our experiments we observed that, from time to time, that absolute difference does not decrease monotonically. Taking all this into account, we found it advisable to introduce a "large" factor between the requested absolute difference between the last two elements of the accelerated series and the desired truncation error, $\epsilon/4$, and to require that the absolute difference between the last two elements of the accelerated series be smaller than the requested one two consecutive times. We decided to take a factor of 25, making the requested absolute difference between the last two elements of the accelerated series equal to $\epsilon/100$.

In the case of inverting $\widehat{C}_{K,L}(s)$ or $\widehat{C}_K(s)$, the error must be $\leq t\epsilon/2$ (to have an error in $EARR^{K,L}(t)$ or $EARR^{K}(t) \leq \epsilon/2$), and we allocate $t\epsilon/4$ for the approximation error and $t\epsilon/4$ for the truncation error. From $f(t) \leq r_{\max}\, t$, assuming $a > 0$, which implies $e^{-2aT} < 1$, we obtain

$$\begin{aligned} f^{\epsilon}(t) &\leq \sum_{k=1}^{\infty} r_{\max}(2kT + t)e^{-2akT} = r_{\max}\, t \sum_{k=1}^{\infty} e^{-2akT} + 2r_{\max}\, T \sum_{k=1}^{\infty} k e^{-2akT} \\ &= r_{\max}\, t \frac{e^{-2aT}}{1 - e^{-2aT}} + 2r_{\max} T\, \frac{e^{-2aT}}{(1 - e^{-2aT})^2} = r_{\max} \frac{(t + 2T)e^{-2aT} - te^{-4aT}}{(1 - e^{-2aT})^2}\,, \end{aligned}$$

and to make the approximation error $\leq t\epsilon/4$ we take the $a > 0$ satisfying

$$r_{\max} \frac{(t + 2T)e^{-2aT} - te^{-4aT}}{(1 - e^{-2aT})^2} = t\, \frac{\epsilon}{4}\,,$$

i.e.

$$a = \frac{1}{2T} \log\left(\frac{1}{x}\right)\,,$$

with

$$x = \frac{r_{\max}(t+2T) + t\epsilon/2 - \sqrt{(r_{\max}(t+2T) + t\epsilon/2)^2 - (r_{\max} + \epsilon/4)t^2\epsilon}}{(2r_{\max} + \epsilon/2)t} \,. \tag{23}$$

To control the truncation error we decide that convergence has been achieved when the absolute difference between the last two elements of the accelerated series is $\leq t\epsilon/100$ two consecutive times, i.e. we leave a safety factor of 25, as when inverting $\widehat{ETRR}^{K,L,a}(s)$ and $\widehat{ETRR}^{K,a}(s)$. Expression (23) has severe cancellation errors when $y = \sqrt{(r_{\max} + \epsilon/4)t^2\epsilon}/(r_{\max}(t+2T) + t\epsilon/2) \ll 1$. The problem can be solved by using the shortest non-null Taylor series of x on y. This gives

$$x \approx \frac{t\epsilon}{4(r_{\max}(t+2T) + t\epsilon/2)} \,,$$

which is accurate for $y < 10^{-3}$.

4 Analysis

We will start by analyzing the performance of the regenerative randomization with Laplace transform inversion method (RRL). For irreducible models, we will compare RRL with regenerative randomization (RR) and randomization with steady-state detection (RSD) [15]. For models with absorbing states, we will compare RRL with RR and standard randomization (SR). The comparison will be made using dependability models of moderate size of a RAID 5 architecture [30]. The RAID 5 architecture provides high throughput and dependability at a moderate cost and is widely used in storage servers. The models belong to class C'.

Figure 6 shows the architecture of the considered RAID 5 system. The system includes $G \times N_{\rm C}$ disks and $N_{\rm C}$ controllers. The disks are organized into G parity groups, each with $N_{\rm C}$ disks. Each controller controls a string of G disks. The system also includes $C_{\rm H}$ hot spare controllers and $D_{\rm H}$ hot spare disks. The information stored in each parity group is organized into groups of $N_{\rm C}$ blocks, of which $N_{\rm C} - 1$ blocks contain data bits and one block contains parity bits. Each block of a group is stored in a different disk of the parity group. Since parity blocks are accessed more often than data blocks, parity blocks are distributed evenly among the disks to achieve load balancing and maximum performance. The system is up if there are at least $N_{\rm C} - 1$ available disks in each parity group. A disk is available if it is unfailed, the controller of its string is unfailed and has updated data. When a failed disk is replaced by a good one, the replaced disk is obviously outdated. The other reason why an unfailed disk may be outdated is because the controller of its string has previously failed. Assuming the disk and the controller of its string unfailed, an outdated disk is updated by a reconstruction process in which all remaining disks of the string participate. The reconstruction process requires that those disks be available and, thus, can only occur when the system is up. All disks of the parity group involved in a reconstruction are "overloaded" and have a higher failure rate. Non-overloaded disks fail with rate $\lambda_{\rm D}$. Overloaded disks fail with rate $\lambda_{\rm O}$. Controllers fail with rate $\lambda_{\rm C}$. The reconstruction process has an exponential duration with rate $\mu_{\rm DRC}$. Failed disks and controllers are replaced, if hot spares are available, by a repairman with rates $\mu_{\rm DRP}$ and $\mu_{\rm CRP}$, respectively, with priority given to controllers. Lacking spares and failed disks and controllers for

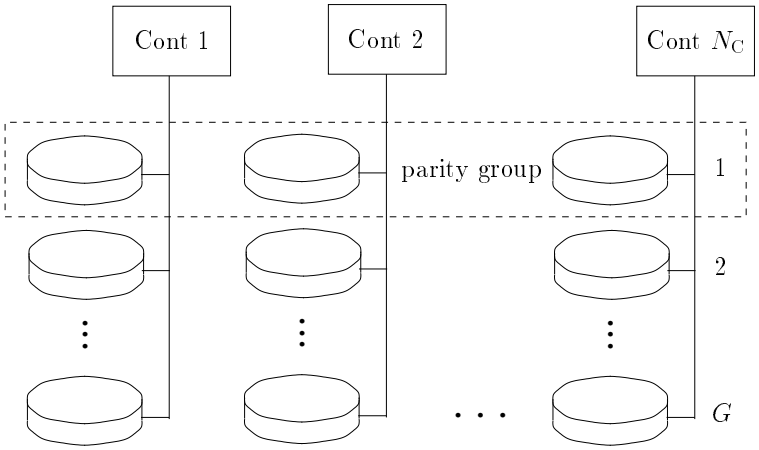

Figure 6: Architecture of the considered RAID 5 system.

which there are not spares are replaced with rate $\mu_{\rm SR}$ by an unlimited number of repairmen. A reconstruction process is successful with probability $P_{\rm R}$. Failure of a reconstruction process causes the failure of the system. Finally, when the system is down, it is brought to a fully operational state, with all disks in the parity groups available and all hot spares available, by a global repair action which has rate $\mu_{\rm G}$, and it is assumed that this is the only repair action undertaken when the system is down.

An exact modeling of the previously described RAID 5 system results in very large CTMCs for even moderate values of G and $N_{\rm C}$. An approximation will be used to reduce the size of the CTMCs. Unavailable disks are said to be aligned if they belong to the same string. The approximation consists in assuming that, if unavailable disks are not aligned, when one of them becomes available the remaining disks would still be unaligned whenever their number is ≥ 2. Using that approximation, it is possible to describe the state of the system using the following state variables: NFD (number of failed disks), NDR (number of disks under reconstruction), NDW (number of disks waiting for reconstruction), NSD (number of hot spare disks), AL (a boolean variable which has value yes when unavailable disks —disks which are failed, under reconstruction, or waiting for reconstruction— are aligned and value NO otherwise), NFC (number of failed controllers), NSC (number of hot spare controllers), and DOWN (a boolean value which has value YES when the system is down and value NO when it is up). Since all down states of the system are dealt with identically, the approximated model only includes one down state. This is achieved by setting the other state variables to predefined values whenever the state variable DOWN is set to yes.

We will consider for the model parameter G the values $G = 20$ and $G = 40$ and will fix the other parameters of the model to the values: $N_{\rm C} = 5$, $C_{\rm H} = 1$, $D_{\rm H} = 3$, $\lambda_{\rm D} = 10^{-5}\,{\rm h}^{-1}$, $\lambda_{\rm O} = 2 \times 10^{-5}\,{\rm h}^{-1}$, $\lambda_{\rm C} = 5 \times 10^{-5}\,{\rm h}^{-1}$, $\mu_{\rm DRC} = 1\,{\rm h}^{-1}$, $\mu_{\rm DRP} = 4\,{\rm h}^{-1}$, $\mu_{\rm CRP} = 4\,{\rm h}^{-1}$, $\mu_{\rm SR} = 0.25\,{\rm h}^{-1}$, $\mu_{\rm G} = 0.25\ {\rm h}^{-1}$ and $P_{\rm R} = 0.999$. We will consider three measures: the unavailability at time t, *UA*(t), the expected interval unavailability at time t, *EIUA*(t), and the unreliability at time t, *UR*(t). The measures *UA*(t) and *EIUA*(t) can be obtained as the generic measures *ETRR*(t) and *EARR*(t) considered in the paper for the irreducible CTMC model previously described if a reward rate 1 is

assigned to the single down state and reward rates 0 are assigned to the remaining, up, states. The *UR*(t) measure can be obtained as the generic measure *ETRR*(t) considered in the paper for the CTMC obtained from the irreducible CTMC by making the down state absorbing if a reward rate 1 is assigned to that state and a reward rate 0 is assigned to the remaining, up, states. The irreducible CTMC model has 3,841 states and 24,785 transitions for $G = 20$ and 14,081 states and 94,405 transitions for $G = 40$. The unreliability model has the same number of states as the irreducible model and one transition less. Both models belong to class C', the state o being the state in which the system is in its fully operational state; that state will be taken as the regenerative state in both the RRL and RR methods. It will be assumed that the system is initially in state o, implying $\alpha_{S'} = 0$ for the RR and RRL methods. All methods were run with a single target time t and $\epsilon = 10^{-12}$. The CPU times were measured on a 167 MHz UltraSPARC 1 workstation.

We start by comparing RRL, RR, and RSD for the irreducible model and the *UA*(t) measure. Table 1 gives the value of the truncation parameter K required by RRL and RR and the number of steps, N, on $\widehat{X}$ required by RSD for several values of t. Figure 7 plots the corresponding CPU times. The parameters K and N have similar values. The parameter K is smaller than N up to a certain value of t and greater beyond that value. Regarding CPU times, RRL is about as fast as RSD and significantly faster than RR for large t. The numerical Laplace transform inversion is fast and consumes a small percentage of the CPU time consumed by the RRL method. That percentage had a maximum value of $7.5\,\%$ for the example with $G = 20$ and a maximum value of $1.7\,\%$ for the example with $G = 40$. The accelerated series passed the convergence test when the original series was computed up to a k ranging from 121 to 281. Since the computational cost of RRL for models which are not too small is dominated by the generation of the truncated transformed model and that cost divided by K is similar to the cost per step on $\widehat{X}$, we should expect the computational cost of RRL to be similar to the computational cost of RSD for models which are not too small when the initial probability distribution of the model is concentrated in the regenerative state. Otherwise, RRL will tend to be somewhat more expensive due to the computational cost associated with the truncation parameter L. The RR method is significantly more expensive than RRL for large t due to the relative high computational cost of solving the truncated transformed model by standard randomization. This will happen when the original model is stiff and not very large and, thus, we should expect RRL to be significantly faster than RR for stiff, not very large irreducible models. Table 2 and Figure 8 give the results obtained when computing the measure *EIUA*(t). The performance of the methods compare as for the measure *UA*(t).

We now compare RRL, RR and SR for the unreliability model. Table 3 and Figure 9 give the results. The value of the truncation parameter K in RR and RRL is always smaller than the number of steps on $\widehat{X}$ required by SR. For small t, SR is slightly faster than both RR and RRL. This is because the cost of the generation of the truncated transformed model divided by K is somewhat larger than the cost of a step on $\widehat{X}$. However, for large t, RR becomes significantly faster than SR and RRL becomes significantly faster than RR. As for the irreducible model, the latter is due to the relative high computational cost of the solution of the truncated transformed model by standard randomization in RR for large t. The slightly superior performance of SR over both RR and RRL for small t will be accentuated when the the initial probability distribution of the model is not

Table 1: Required K in RR and RRL and required number of steps, N, in RSD when computing $UA(t)$ for several values of t.

	$G = 20$		$G = 40$	
t (h)	RR/RRL	RSD	RR/RRL	RSD
1	56	66	86	99
2	91	104	145	162
5	183	204	306	332
10	323	355	554	594
20	583	637	1,024	1,092
50	1,299	1,439	2,346	2,526
100	2,234	2,612	4,187	4,823
200	2,496	2,612	4,725	4,823
500	2,630	2,612	4,978	4,823
1,000	2,708	2,612	5,123	4,823
2,000	2,780	2,612	5,257	4,823
5,000	2,871	2,612	5,425	4,823
10,000	2,938	2,612	5,549	4,823
20,000	3,004	2,612	5,672	4,823
50,000	3,091	2,612	5,834	4,823
100,000	3,157	2,612	5,957	4,823

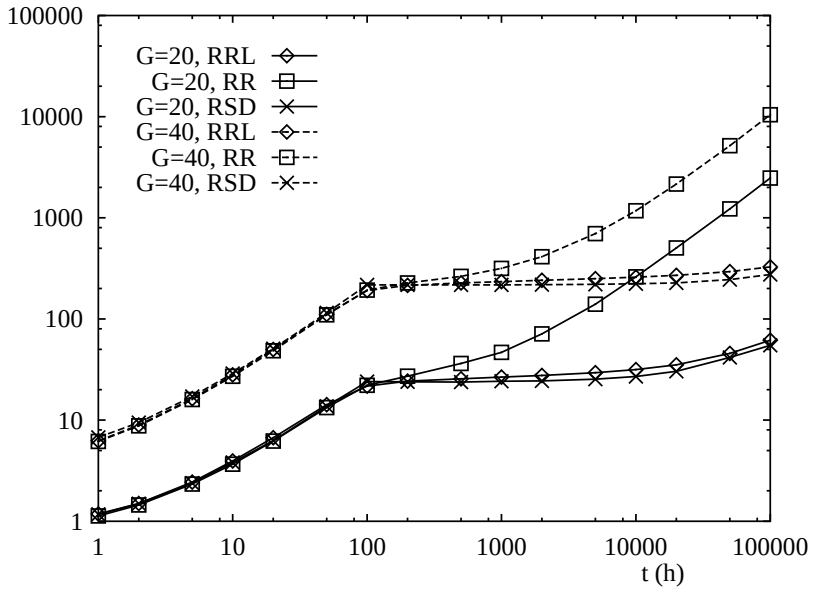

Figure 7: CPU times in seconds required by RRL, RR and RSD when computing $UA(t)$ as a function of t.

Table 2: Required K in RR and RRL and required number of steps, N, in RSD when computing *EIUA*(t) for several values of t.

	$G = 20$		$G = 40$	
t (h)	RR/RRL	RSD	RR/RRL	RSD
1	52	66	81	99
2	85	104	138	162
5	174	204	293	332
10	309	355	534	594
20	562	637	993	1,092
50	1,252	1,439	2,278	2,526
100	2,056	2,612	3,864	4,823
200	2,370	2,612	4,486	4,823
500	2,542	2,612	4,814	4,823
1,000	2,632	2,612	4,981	4,823
2,000	2,709	2,612	5,124	4,823
5,000	2,802	2,612	5,298	4,823
10,000	2,871	2,612	5,425	4,823
20,000	2,938	2,612	5,549	4,823
50,000	3,025	2,612	5,712	4,823
100,000	3,091	2,612	5,834	4,823

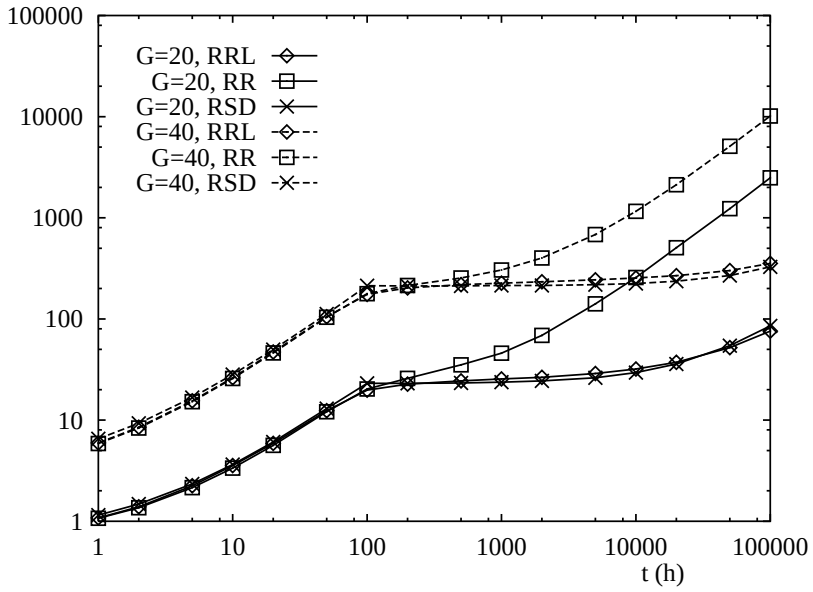

Figure 8: CPU times in seconds required by RRL, RR and RSD when computing *EIUA*(t) as a function of t.

Table 3: Required K in RR and RRL and required number of steps, N, in SR when computing $UR(t)$ for several values of t.

	$G = 20$		$G = 40$	
t (h)	RR/RRL	SR	RR/RRL	SR
1	56	65	86	98
2	91	103	145	161
5	183	203	306	331
10	323	354	554	593
20	583	636	1,024	1,091
50	1,299	1,438	2,346	2,525
100	2,233	2,726	4,186	4,849
200	2,496	5,243	4,724	9,417
500	2,630	12,651	4,976	22,926
1,000	2,708	24,844	5,122	45,234
2,000	2,779	49,046	5,255	89,597
5,000	2,870	121,193	5,423	222,069
10,000	2,937	240,958	5,547	442,203
20,000	3,003	479,900	5,671	881,672
50,000	3,091	1,195,284	5,833	2,198,122
100,000	3,157	2,386,068	5,955	4,390,141

concentrated in the regenerative state due to the cost associated with stepping Z'. For stiff models with absorbing states, RRL will be much faster than SR and significantly faster than RR when the model is not very large.

We now analyze the accuracy of RRL. To that end we will use the dependability model of the C.mmp system described in [2]. The model corresponds to a multiprocessor system including 16 processors, 16 memories and one switch. The system is up if at least four processors, four memories, and the switch are unfailed. Processors fail with rate λ_{P}, memories fail with rate λ_{M} and the switch fails with rate λ_{S}. All failed components are independently repaired. The repair rate is μ_{P} for processors, μ_{M} for memories and μ_{S} for the switch. The corresponding CTMC has 578 states and 2,754 transitions. The unavailability of the system at time t, $UA(t)$ has the following closed form expression:

$$UA(t) = 1 - A_{\mathrm{S}}(t) \left(\sum_{i=4}^{16} \binom{16}{i} A_{\mathrm{P}}(t)^i (1 - A_{\mathrm{P}}(t))^{16-i} \right) \left(\sum_{i=4}^{16} \binom{16}{i} A_{\mathrm{M}}(t)^i (1 - A_{\mathrm{M}}(t))^{16-i} \right) ,$$

where $A_{\mathrm{P}}(t)$, $A_{\mathrm{M}}(t)$ and $A_{\mathrm{S}}(t)$ are, respectively, the availability at time t of a processor, a memory and the switch. Each availability is given by

$$A(t) = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} e^{-(\lambda+\mu)t} ,$$

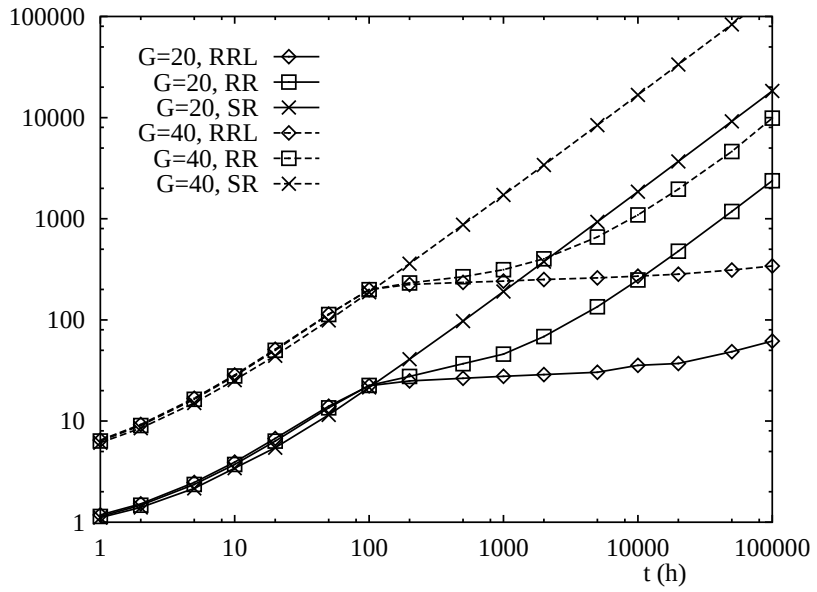

Figure 9: CPU times in seconds required by RRL, RR and SR when computing $UR(t)$ as a function of t.

where λ and μ are the failure and repair rates of the component. Based on that closed form expression we computed "exact" values of $UA(t)$ using a mathematical software package with enough number of digits. "Exact" values for the expected interval unavailability at time t, $EIUA(t)$, were computed using the relationship $EIUA(t) = (\int_0^t UA(\tau)\, d\tau)/t$. We then compared the values obtained by RRL when run with a single target time t for several values of ϵ with the "exact" values to obtain the computation errors in RRL. We used $\lambda_{\rm P} = \lambda_{\rm M} = 10^{-4}\ {\rm h}^{-1}$, $\lambda_{\rm S} = 10^{-6}\ {\rm h}^{-1}$, and $\mu_{\rm P} = \mu_{\rm M} = \mu_{\rm S} = 1\ {\rm h}^{-1}$. Figures 10 and 11 give the results. In all cases, the actual computation error is much smaller than the requested one, ϵ. This is because the error upper bounds used to control the model truncation error and the approximation error of the Laplace transform inversion algorithm are very coarse, and because the accelerated series obtained with the epsilon algorithm converges relatively fast, making the actual truncation error of the Laplace transform inversion algorithm typically much smaller than the controlled absolute tolerance in the accelerated series, $\epsilon/100$. The coarseness of the model truncation error upper bounds comes from the fact that those bounds are obtained by assigning a reward rate equal to $r_{\max} = 1$ to the truncated behavior, captured by the absorbing state a, when in fact the averaged reward rate in that behavior is of the order of the $UA(t)$, which varies from 9.51626×10^{-8} for $t = 0.1$ h to 9.99999×10^{-7} for $t =$ 10,000 h. The coarseness of the upper bound of the approximation error of the Laplace transform inversion algorithm comes from the fact that the upper bound is obtained by upper bounding $UA(t)$ and $EIUA(t)$ by $r_{\max} = 1$, while $UA(t)$ and $EIUA(t)$ have much smaller values. The Laplace transform inversion algorithm seems to be numerically very stable. Thus, for $t =$ 10,000 h, $UA(t) = 9.99999 \times 10^{-7}$ and $EIUA(t) = 9.99899 \times 10^{-7}$, and, for $\epsilon = 10^{-18}$, the accelerated series passed the convergence test with an absolute tolerance of $\epsilon/100 = 10^{-20}$, which is 14 orders of magnitude smaller than the inverted value. In summary, RRL seems to be able to achieve stringent accuracy levels safely.

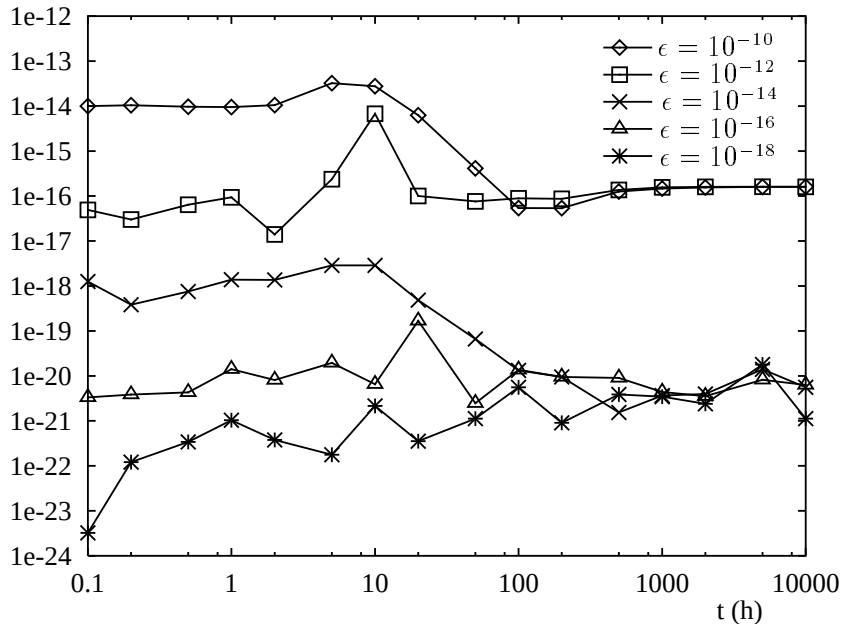

Figure 10: Computation errors (in absolute value) in $UA(t)$ under RRL for several values of ϵ.

5 Related work

Calderna and Carrasco [31, 32] describe preliminary related work. With respect to the regenerative randomization method as described in [16, 17], those references deal with the particular cases in which, respectively, $A = 0$ and the CTMC is irreducible and $A = 1$ and all states in S are transient. More importantly, in [31, 32] the model truncation error is controlled by solving the truncated transformed model exactly using a simpler Laplace transform inversion algorithm than the one we use here each time the parameter K or the parameter L is increased. When the required K and L are not small, that approach introduces a considerable overhead over the approach taken in [16, 17] and here, based on easy-to-compute upper bounds for the part of the model truncation error which depends on K. We should note that using an exact solution of the truncated transformed model as in [31, 32] typically reduces the required K very little [16]. Another important difference is the absence of theoretical results in [31, 32] assessing the efficiency of the method for class C' models. This is relevant since, although particular, class C' is an important model class and it is important to be able to anticipate when the regenerative randomization method will be more efficient than other alternatives (the standard randomization method).

6 Conclusions and discussion

Rewarded continuous time Markov models find applications in dependability and performability analysis of computer and telecommunication systems. The transient analysis of those models is costly when the model is stiff, making the development of efficient transient analysis techniques for those models a research topic of great interest. In this paper we have considered two transient

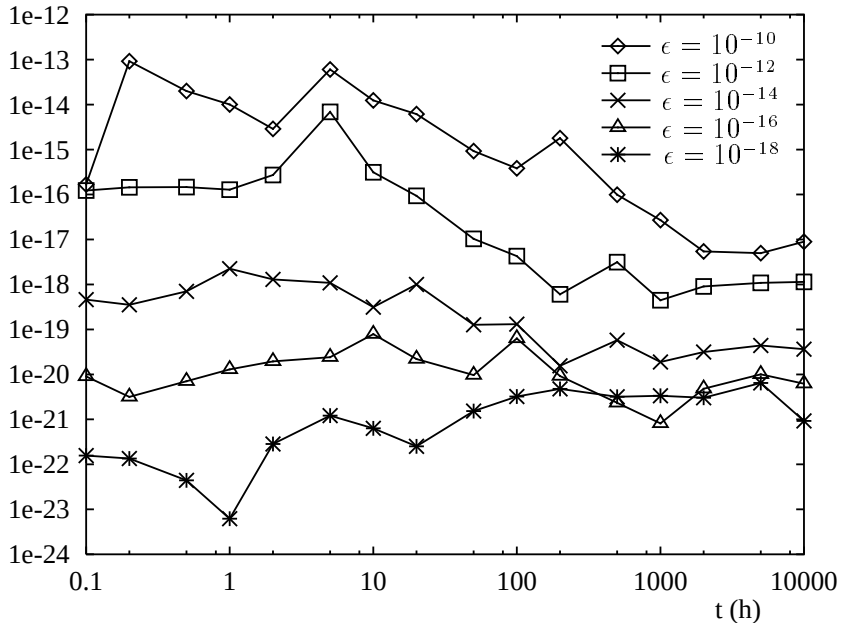

Figure 11: Computation errors (in absolute value) in *EIUA*(t) under RRL for several values of ϵ.

measures, the expected transient reward rate and the expected averaged reward rate, and have developed a variant, regenerative randomization with Laplace transform inversion, of the regenerative randomization method for the computation of those measures. As regenerative randomization, the performance of the variant can be approximately predicted in advance for a class of models, class C', including failure/repair models with exponential failure and repair time distributions and repair in every state with failed components when failure rates are significantly smaller than repair rates. For irreducible models of that class, the new variant seems to be about as fast as randomization with steady-state detection for models which are not too small when the initial probability distribution is concentrated in the regenerative state and significantly faster than regenerative randomization when the model is stiff and not very large. For stiff class C' models with absorbing states, the variant is much faster than standard randomization and significantly faster than regenerative randomization when the model is not very large. Since the cost of the second phase of regenerative randomization is approximately proportional to the stiffness of the model ($\max_{i\in\Omega} \lambda_i t$) and the cost of the first phase is approximately proportional to the size of the model (number of transitions), as the stiffness increases the second phase will be relatively costly for larger models and, therefore, the variant proposed in this paper will be attractive for larger models. The quality of the numerical solution obtained by the variant is not as high as the quality of the numerical solution obtained by regenerative randomization, due to the impossibility of upper bounding the truncation error of the Laplace transform inversion algorithm. However, the variant seems to be able to achieve stringent accuracy levels safely.

Another possibility to improve the efficiency of regenerative randomization would be to solve the truncated transformed model using a stiff ODE solver. When the original model is stiff, this would be certainly more efficient than solving the truncated transformed model by standard randomization. However, the quality of the numerical solution achieved by a stiff ODE solver is inferior to

the quality of the solution achieved by the Laplace transform inversion algorithm we use. Three error sources exist in a stiff ODE solver: (1) local truncation error inherent to the approximation formula used in the steps of the method; (2) error in the solution of the linear systems involved in the steps by an iterative method (direct methods would be much more costly); and (3) propagation of the above local errors. In practice, a stiff ODE solver requests an error tolerance and that error tolerance is used to control (without upper bounding) the local errors, and the only thing which is known about the actual error is that it is roughly proportional to the error tolerance. This is certainly inferior to the quality of error control in the Laplace transform inversion algorithm we use. Secondly, a stiff ODE solver would probably be less efficient, from a computational point of view, when the number of time points at which the measure has to be computed is not large, since the Laplace transform inversion algorithm typically requires about 150 Laplace transform evaluations while a stiff ODE solver requires of the order of thousands of vector-matrix multiplications [1, 2, 3] and the cost of a Laplace transform evaluation is not much larger than the cost of a vector-matrix multiplication. Finally, even sophisticated stiff ODE solvers [2] achieve with difficulty and high computational cost the accuracy levels that the Laplace transform inversion algorithm we use has shown to achieve.

Acknowledgments

This research work was supported by the "Comisión Interministerial de Ciencia y Tecnología (CICYT)" of the Ministry of Science and Technology of Spain under the research grant TAP99-0443-C05-05.

References

[1] Malhotra, M., Muppala, J. K. and Trivedi, K. S. (1994) Stiffness-Tolerant Methods for Transient Analysis of Stiff Markov Chains. *Microelectronics and Reliability*, 34, 1825–1841.

[2] Malhotra, M. (1995) A computationally efficient technique for transient analysis of repairable Markovian systems. *Performance Evaluation*, 24, 311–331.

[3] Reibman, A. and Trivedi, K. S. (1988) Numerical Transient Analysis of Markov Models. *Computers and Operations Research*, 15, 19–36.

[4] Grassman, W. K. (1993) Rounding Errors in Certain Algorithms Involving Markov Chains. *ACM Transactions on Mathematical Software*, 19, 496–508.

[5] Grassmann, W. K. (1977) Transient solutions in Markovian queuing systems. *Computers and Operations Research*, 4, 47–53.

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

[7] Béounes, C. et al. (1993) SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems. In *Proc. 23rd IEEE Int. Symp. on Fault-Tolerant Computing*, Toulouse, France, August, 668–673.

[8] Ciardo, G., Muppala, J. and Trivedi, K. (1989) SPNP: Stochastic Petri Net Package. In *Proc. 3rd IEEE Int. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December, pp. 142–151.

[9] Couvillon, J., Freire, R., Johnson, R., Obal II, W., Qureshi, A., Rai, M., Sanders, W. and Tvedt J. (1991) Performability modelling with UltraSAN. *IEEE Software*, 8, 69–80.

[10] Goyal, A., Carter, W. C., de Souza e Silva, E. and Lavenberg, S. S. (1986) The System Availability Estimator. In *Proc. 16th IEEE Int. Symp. on Fault-Tolerant Computing*, July, 84–89.

[11] Melamed, B. and Yadin, M. (1984) Randomization Procedures in the Computation of Cumulative-Time Distributions over Discrete State Markov Processes. *Operations Research*, 32, 926–944.

[12] Miller, D. R. (1983) Reliability Calculation using Randomization for Markovian Fault-Tolerant Computing Systems. In *Proc. 13th IEEE Int. Symp. on Fault-Tolerant Computing*, Vienna, Austria, June, pp. 284–289.

[13] Moorsel, A. P. and Sanders, W. H. (1994) Adaptive Uniformization. *Communications in Statistics—Stochastic Models*, 10, 619–647.

[14] Moorsel A. P. and Sanders, W. H. (1997) Transient Solution of Markov Models by Combining Adaptive and Standard Uniformization. *IEEE Trans. on Reliability*, 46, 430–440.

[15] Sericola, B. (1999) Availability Analysis of Repairable Computer Systems and Stationarity Detection. *IEEE Trans. on Computers*, 48, 1166–1172.

[16] Carrasco, J. A. (1999) Transient Analysis of Large Markov Models with Absorbing States using Regenerative Randomization. Technical Report DMSD_99_2, Universitat Politècnica de Catalunya, available at `ftp://ftp-eel.upc.es/techreports`.

[17] Carrasco, J. A. (1999) Computation of Bounds for Transient Measures of Large Rewarded Markov Models using Regenerative Randomization. Technical Report DMSD_99_4, Universitat Politècnica de Catalunya, available at `ftp://ftp-eel.upc.es/techreports`.

[18] Carrasco, J. A. (2002) Computationally Efficient and Numerically Stable Reliability Bounds for Repairable Fault-Tolerant Systems. *IEEE Trans. on Computers*, 51, 254–268.

[19] Heidelberger, P., Muppala, K. K. and Trivedi, K. S. (1996) Accelerating mean time to failure computations. *Performance Evaluation*, 27–28, 627–645.

[20] Kijima, M. (1997) *Markov Processes for Stochastic Modeling*. Chapman & Hall, London.

[21] Reibman, A. and Trivedi, K. S. (1989) Transient Analysis of Cumulative Measures of Markov Model Behavior. *Communications in Statistics—Stochastic Models*, 5, 683–710.

[22] Ross, S. M. (1983) *Stochastic Processes*. John Wiley & Sons, New York.

[23] Knüsel, L. (1986) Computation of the Chi-square and Poisson Distribution. *SIAM Journal on Scientific and Statistical Computing*, 7, 1022–1036.

[24] Abramowitz, M. and Stegun, I. A. (1965) *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York.

[25] Johnson, B. W. (1989) *Design and Analysis of Fault Tolerant Digital Systems*. Addison-Wesley, Reading, MA, USA.

[26] Crump, K. S. (1976) Numerical Inversion of Laplace Transforms Using a Fourier Series Approximation. *Journal of the ACM*, 23, 89–96.

[27] Piessens, R. and Huysmans, R (1984) Algorithm 619: Automatic Numerical Inversion of the Laplace Transform [D5]. *ACM Transactions on Mathematical Software*, 10, 348–353.

[28] Durbin, F. (1974) Numerical inversion of Laplace transforms: an efficient improvement to Dubner and Abate's method. *Computer Journal*, 17, 371–376.

[29] Brezinski, C. (1977) *Accélération de la Convergence en Analyse Numérique*, Lecture Notes in Mathematics, 584, Springer-Verlag, Berlin.

[30] Friedman, M. B. (1996) RAID keeps going and going and... *IEEE Spectrum*, 4, 73–79.

[31] Calderón, A. and Carrasco, J. A. (1995) Computation of Absorption Probability Distributions of continuous-time Markov chains using Regenerative Randomization. In *Proc. IEEE Int. Computer Performance and Dependability Symposium*, Erlangen, Germany, April, pp. 92–101.

[32] Carrasco, J. A. and Calderón, A. (1995) Regenerative Randomization: Theory and application examples. *Performance Evaluation Review*, 23, 241–252.