Two New Algorithms to Compute Steady-state Bounds for Markov Models with Slow Forward and Fast Backward Transitions*

Juan A. Carrasco, Angel Calderón and Javier Escribá
Departament d'Enginyería Electrònica, UPC
Diagonal 647, plta. 9
08028 Barcelona, Spain

Abstract

Two new algorithms are proposed for the computation of bounds for the steady-state reward rate of irreducible finite Markov models with slow forward and fast backward transitions. The algorithms use detailed knowledge of the model in a subset of generated states G and partial information about the model in the non-generated portion U of the state space. U is assumed partitioned into subsets U_k, $1 \leq k \leq N$ with a "nearest neighbor" structure. The algorithms involve the solution of, respectively, $|M| + 2$ and 4 linear systems of size $|G|$, where M is the set of values of k corresponding to the subsets U_k through which the model can jump from G to U. Previously proposed algorithms for the same type of models required the solution of $|S|$ linear systems of size $|G|+N$, where S is the subset of G through which the model can enter G from U, to achieve the same bounds as our algorithms, or gave less tighter bounds if state cloning techniques were used to reduce the number of solved linear systems. An availability model with system state dependent repair rates is used to illustrate the application and performance of the algorithms.

1 Introduction

Markov models are widely used for performance, dependability and performability analysis. These models can be solved using numerical techniques or simulation. Both approaches have their advantages and disadvantages. Numerical methods provide a reliable solution of the model but suffer from the well-known state explosion problem. Simulation methods can deal easily with large state spaces but tend to be less reliable when, due to the rarity of the important events of the model, have to be combined with acceleration techniques [2], [6]. Recently, a number of methods have been proposed to compute steady-state availability bounds [3], [9], [10], [11], [12]. In these methods, a finite irreducible continuous-time Markov chain (CTMC) $X = \{X(t); t \geq 0\}$ with state space Ω and reward rate structure $r_i \geq 0$, $i \in \Omega$ is considered, a subset G of states is generated, and bounds to the steady-state reward rate $\lim_{t\to\infty} E[r_{X(t)}]$ are derived using only detailed knowledge of X in G. In the method developed by Muntz, Souza e Silva and Goyal [11] G includes all the states with up to a given number K of failed components. In its simplest version, the method requires the solution of $|S|$ linear systems of size $|G| + N$, where S is the subset of states through which X can jump to G from $U = \Omega - G$ and N is the number of aggregates in which U is decomposed (each aggregate includes the states with a given number of failed components). $|S|$ is typically almost as large as $|G|$ and the method can be very costly. In order to reduce the computational cost, a state cloning technique is proposed in [11], in which duplicates for states in G are added to U so that the new S includes the states with F failed components, where $F < K$. The technique, however, introduces a looseness in the bounds which increases with decreasing F. Lui and Muntz [9] have proposed a refinement of the method for the particular case $F = 0$ including a reuse technique which, at the price of an additional looseness in the bounds, avoids a complete reapplication of the method each time K is incremented in the search for the desired accuracy. The additional looseness has been reduced in a more recent paper from the same authors [10]. Souza e Silva and Ochoa [12] have developed state space exploration techniques in which G is generated incrementally following heuristics which try to obtain the tightest possible bounds for a given number of generated states. More recently, Carrasco [3] has developed a bounding method specifically tailored to availability models which uses the failure distance concept to bound the behavior in U. The method provides significantly tightened bounds than [11] when down states are sparse in U.

In this paper we develop two new algorithms to obtain the same bounds as the algorithm described in [11] without state cloning. The algorithms solve, respectively, $|M| + 2$ and 4 linear systems of size $|G|$, where, when G includes all the states with up to a given number of failed components, M is the set of different cardinalities of the subsets of components of the model which can fail simultaneously. The algorithms include a decomposition technique to accelerate the convergence of iterative methods. As the algorithms described in [9], [10], [11], [12] our algorithms apply to a wide range of models, and its presentation will be intentionally abstracted from the specific peculiarities of availability models. The rest of the paper is organized as follows. Section 2 reviews the bounding algorithm without state cloning described in [11] and develops our algorithms in its basic versions. Section 3 describes the decomposition technique and combines it with the basic algorithms to obtain the final algorithms. Section 4 illustrates the applicability and performance of the algorithms in comparison with the algorithm proposed in [11] with and without state cloning using an availability model with system state dependent repair rates. Section 5 concludes the paper.

*This work was supported by "Comisión Interministerial de Ciencia y Tecnología (CICYT)" under the National Research Grant TIC95-0707-C02-02.

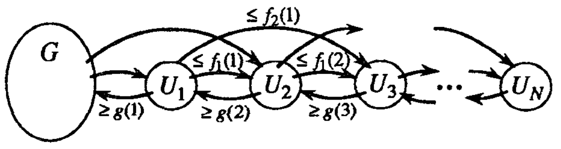

Figure 1: Structure of the X models.

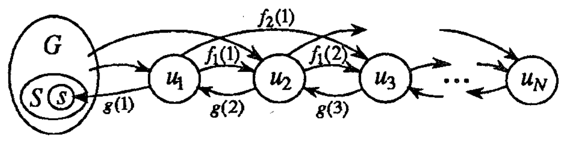

Figure 2: Models X'_s, $s \in S$ used in the algorithm without state cloning described in [11].

2 Basic algorithms

For the model X, we will denote by λ_{ij} the transition rate from state i to state j, by λ_i the output rate of state i, and by $\lambda_{i,C}$ the transition rate from the state i to the subset of states C ($\lambda_{i,C} = \sum_{j \in C} \lambda_{ij}$). The bounding algorithm without state cloning described in [11] assumes that U can be partitioned into subsets U_k, $1 \le k \le N$ such that: 1) the subsets have a "nearest neighbor" structure, i.e., states in U_k, $k > 1$ have no transitions to U_i, $i < k-1$ or G, and 2) upper bounds $f_i(k) \ge 0$ for $\lambda_{j,U_{k+i}}$, $j \in U_k$, $0 < i \le N-k$ and lower bounds $g(k) > 0$, $k > 1$ for $\lambda_{j,U_{k-1}}$, $j \in U_k$ and $g(1) > 0$ for $\lambda_{j,G}$, $j \in U_1$ are known (see Figure 1). A lower bound $[r]_{lb}$ and an upper bound $[r]_{ub}$ for all the reward rates of the model are also assumed known. Failure/repair models such as those solved in the SAVE package [5] are examples of models satisfying those requirements. For those models G can include the states with up to K failed components and a partition for U with $U_k = \{$states with $K+k$ failed components$\}$ can be considered. With that partition, $f_i(k)$ can be taken equal to the sum of the maximum rates of the failure events of the model (sets of components which can fail simultaneously) involving i components and $g(k)$ can be taken equal to the slowest repair rate. If the measure of interest is the steady-state unavailability ($r_i = 0$ if i is an operational state and $r_i = 1$ if i is a down state), we can take $[r]_{lb} = 0$ and $[r]_{ub} = 1$.

The bounding algorithm without state cloning described in [11] considers the CTMC's X'_s, $s \in S$ depicted in Figure 2. S includes the "return" states of G, i.e., the states through which X can jump from U to G and the states u_k collect the transitions from G to the subsets U_k. For availability models, S would include all states with K failed components. The algorithm can be described as follows:

1. For each $s \in S$ solve the steady-state regime of X'_s. Let $[R_s]_{lb}$ be the steady-state reward rate of X'_s with reward rate structure $r'_i = r_i$, $i \in G$, $r'_i = [r]_{lb}$, $i \in \{u_1, \ldots, u_N\}$. Let $[R_s]_{ub}$ be the steady-state reward rate of X'_s with reward rate structure $r'_i = r_i$, $i \in G$, $r'_i = [r]_{ub}$, $i \in \{u_1, \ldots, u_N\}$.
2. $[R]_{lb} = \min_{s \in S}\{[R_s]_{lb}\}$, $[R]_{ub} = \max_{s \in S}\{[R_s]_{ub}\}$

The algorithm involves the solution of $|S|$ linear systems of size $|G| + N$ and can be very expensive when $|S|$ is large. This is usually the case for availability models since $|S|$ is typically almost as large as $|G|$.

To describe our algorithms it is useful to split the models X'_s, $s \in S$ into two transient CTMC's. The first one, Y_G, is obtained from X by making absorbing the exits of G; the second one, Y'_U, has the state transition diagram of Figure 3. Let $C_{G,s}$ be the expected accumulated reward to absorption of Y_G with initial state s and same reward rate structure as X in G, and let $T_{G,s}$ be the mean time to absorption of Y_G with initial state s. Let $\alpha_s(k)$, $1 \le k \le N$ be the probability that X with initial state s exits G through U_k (identical to the probability that X'_s exits G through u_k), let $T(k)$, $1 \le k \le N$, be the mean time to absorption of Y'_U with initial state u_k, and let M be the set of values of k for which X has transitions from G to U_k. Then, considering the regenerative structure of X'_s defined by the times at which X'_s enters s from $\{u_1, \ldots, u_N\}$ and the relationships between X'_s and Y_G, Y'_U, we have:

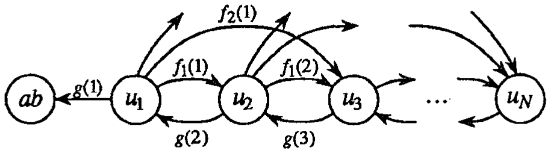

Figure 3: State transition diagram of Y'_U.

$$[R_s]_{lb} = \frac{C_{G,s} + [r]_{lb}[T_{U,s}]_{ub}}{T_{G,s} + [T_{U,s}]_{ub}}, \tag{1}$$

$$[R_s]_{ub} = \frac{C_{G,s} + [r]_{ub}[T_{U,s}]_{ub}}{T_{G,s} + [T_{U,s}]_{ub}}, \tag{2}$$

with

$$[T_{U,s}]_{ub} = \sum_{k \in M} \alpha_s(k) T(k). \tag{3}$$

The notation $[T_{U,s}]_{ub}$ is justified by the fact that, by the mean holding time lemma of [11], $[T_{U,s}]_{ub}$ upper bounds the mean time spent by X with return state s in each visit to U. Using (1), (2) the lower and upper bounds for R can be written as;

$$[R]_{lb} = \min_{s \in S}\left\{\frac{C_{G,s} + [r]_{lb}[T_{U,s}]_{ub}}{T_{G,s} + [T_{U,s}]_{ub}}\right\}, \tag{4}$$

$$[R]_{ub} = \max_{s \in S}\left\{\frac{C_{G,s} + [r]_{ub}[T_{U,s}]_{ub}}{T_{G,s} + [T_{U,s}]_{ub}}\right\}. \tag{5}$$

Our first basic algorithm involves the computation of $T(k)$, $k \in M$; $\alpha_s(k)$, $s \in S$, $k \in M$; and $C_{G,s}$, $T_{G,s}$, $s \in S$. The second one computes $T(k)$, $k \in M$; and $T'_s = T_{G,s} + [T_{U,s}]_{ub}$, $C'_s = C_{G,s} + [r]_{lb}[T_{U,s}]_{ub}$, $C''_s = C_{G,s} + [r]_{ub}[T_{U,s}]_{ub}$, $s \in S$. The bounds for R can be expressed in terms of T'_s, C'_s, and C'_s, $s \in S$ as:

$$[R]_{lb} = \min_{s \in S}\left\{\frac{C'_s}{T'_s}\right\}, \tag{6}$$

$$[R]_{ub} = \min_{s \in S} \left\{ \frac{C''_s}{T'_s} \right\}. \tag{7}$$

A direct computation of $T(k)$, $k \in M$, based on the solution of Y'_U with each u_k as initial state would involve the solution of $|M|$ linear systems. A more efficient procedure, specially for large $|M|$, can be developed exploiting the following equations, where $\lambda(k) = g(k) + \sum_i f_i(k)$ denotes the output rate of u_k in Y'_U:

$$T(k) = \frac{1}{\lambda(k)} + \frac{g(k)}{\lambda(k)} T(k-1) + \sum_i \frac{f_i(k)}{\lambda(k)} T(k+i), \quad 1 < k < N, \tag{8}$$

$$T(N) = \frac{1}{g(N)} + T(N-1). \tag{9}$$

These equations are obtained as follows. First, consider (8). $T(k)$, mean time to absorption of Y'_U with initial state u_k, is equal to the mean sojourn time in u_k, $1/\lambda(k)$, plus the mean time to absorption from the next visited state. The next visited state is u_{k-1} with probability $g(k)/\lambda(k)$ and u_{k+i} with probability $f_i(k)/\lambda(k)$. Equation (9) is obtained similarly; in this case, $\lambda(N) = g(N)$ and state u_{N-1} is the next visited state with probability 1. Equations (8), (9) can be solved recursively in $T(k)$, $1 \le k < N$ in terms of $T(N)$, yielding:

$$T(N-1) = T(N) - \frac{1}{g(N)}, \tag{10}$$

$$T(k) = \frac{1}{g(k+1)} \Big[\lambda(k+1) T(k+1) - 1 - \sum_i f_i(k+1) T(k+1+i) \Big], 1 \le k < N-1. \tag{11}$$

It remains to discuss the computation of $T(N)$. Let τ_i denote the mean time to absorption in state u_i of Y'_U with initial state u_N. Then,

$$T(N) = \sum_{i=1}^{N} \tau_i. \tag{12}$$

The vector $\tau^T = (\tau_1 \ldots \tau_N)$ is the solution of the linear system:

$$\tau^T \mathbf{A} = -(0 \ldots 01), \tag{13}$$

where $\mathbf{A}$ is the restriction of the transition rate matrix of Y'_U to the transient states. A direct solution of (13) is possible exploiting the upper Hessenberg structure of $\mathbf{A}$ and the fact that the $N-1$ first components of the right-hand vector of (13) are null. Defining $\nu_i = \tau_i/\tau_1$ ($\nu_1 = 1$), the first $N-1$ equations of (13) give a triangular linear system on ν_i, $2 \le i \le N$ which can be easily solved. Substituting then τ_i by $\nu_i \tau_1$, $2 \le i \le N$, in the last equation of (13) and using the solution for ν_i, $2 \le i \le N$ found in the previous step gives an equation on τ_1. Solving that equation and using $\tau_i = \nu_i \tau_1$, $2 \le i \le N$ we obtain τ_i, $2 \le i \le N$. The solution procedure can be described as follows:

$$\begin{aligned} \nu_1 &= 1, \\ \nu_i &= \frac{1}{g(i)} \Big[\lambda(i-1)\nu_{i-1} - \sum_{j=1}^{i-2} f_{i-j-1}(j) \nu_j \Big], \quad i = 2, \ldots, N, \end{aligned} \tag{14}$$

$$\begin{aligned} \tau_1 &= \frac{1}{\lambda(N)\nu_N - \sum_{i=1}^{N-1} f_{N-i}(i)\nu_i}, \\ \tau_i &= \nu_i \tau_1, \quad i = 2, \ldots, N. \end{aligned} \tag{15}$$

We next develop computational procedures for $T_{G,s}$, $C_{G,s}$ and $\alpha_s(k)$, $s \in S$, $k \in M$, and T'_s, C'_s, C''_s, $s \in S$. We start discussing the computation of $T_{G,i}$, $i \in G$. $T_{G,i}$ is equal to the mean sojourn time in i plus the mean time to absorption of Y_G from the next visited state. Since the mean sojourn time in i is $1/\lambda_i$ and the next visited state is j with probability λ_{ij}/λ_i, we have:

$$T_{G,i} = \frac{1}{\lambda_i} + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} T_{G,j}, \quad i \in G. \tag{16}$$

$C_{G,i}$ is equal to the mean accumulated reward during a visit to i, r_i/λ_i, plus the mean accumulated reward to absorption of Y_G from the next visited state, yielding the equations:

$$C_{G,i} = \frac{r_i}{\lambda_i} + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} C_{G,j}, \quad i \in G. \tag{17}$$

$\alpha_i(k)$ is equal to the probability that X jumps from i to U_k plus the probability that X, starting in the next state, exits G through U_k, yielding:

$$\alpha_i(k) = \frac{\lambda_{i,U_k}}{\lambda_i} + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} \alpha_j(k), \quad i \in G, k \in M. \tag{18}$$

To obtain similar equations for T'_i, C'_i, and C''_i it is useful to remember (3) and that $\alpha_s(k)$ is the probability that X with initial state s exits G through U_k. This allows us to decompose $[T_{U,i}]_{ub}$ in contributions associated to the probabilities of following transitions out of G at particular steps of the embedded discrete-time Markov chain. For $T'_i = T_{G,i} + [T_{U,i}]_{ub}$ we obtain:

$$T'_i = \frac{1}{\lambda_i} + \sum_{k \in M} \frac{\lambda_{i,U_k}}{\lambda_i} T(k) + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} T'_j, \quad i \in G. \tag{19}$$

For $C'_i = C_{G,i} + [r]_{lb}[T_{U,i}]_{ub}$ and $C''_i = C_{G,i} + [r]_{ub}[T_{U,i}]_{ub}$ we have:

$$C'_i = \frac{r_i}{\lambda_i} + \sum_{k \in M} \frac{\lambda_{i,U_k}}{\lambda_i} [r]_{lb} T(k) + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} C'_j, \quad i \in G. \tag{20}$$

$$C''_i = \frac{r_i}{\lambda_i} + \sum_{k \in M} \frac{\lambda_{i,U_k}}{\lambda_i} [r]_{ub} T(k) + \sum_{\substack{j \in G \\ j \neq i}} \frac{\lambda_{ij}}{\lambda_i} C''_j, \quad i \in G. \tag{21}$$

Let $q_{ij} = \lambda_{ij}/\lambda_i$, equations (16), (17), (18), (19), (20) and (21) can be formulated as linear systems introducing the matrix $\mathbf{B} = \mathbf{I} - (q_{ij})_{i,j \in G}$ and the column vectors $\mathbf{T} = (T_{G,i})_{i \in G}$, $\mathbf{C} = (C_{G,i})_{i \in G}$, $\boldsymbol{\alpha}(k) = (\alpha_i(k))_{i \in G}$, $\mathbf{T}' = (T'_i)_{i \in G}$, $\mathbf{C}' = (C'_i)_{i \in G}$, $\mathbf{C}'' = (C''_i)_{i \in G}$, $\boldsymbol{\mu} = (1/\lambda_i)_{i \in G}$, $\mathbf{c} = (r_i/\lambda_i)_{i \in G}$, $\boldsymbol{\beta}(k) = (\lambda_{i,U_k}/\lambda_i)_{i \in G}$, $\boldsymbol{\mu}' = ((1/\lambda_i) + \sum_{k \in M} (\lambda_{i,U_k}/\lambda_i) T(k))_{i \in G}$,

$\mathbf{c}' = ((r_i/\lambda_i) + \sum_{k\in M}(\lambda_{i,U_k}/\lambda_i)[r]_{lb}T(k))_{i\in G}$, and $\mathbf{c}'' = ((r_i/\lambda_i) + \sum_{k\in M}(\lambda_{i,U_k}/\lambda_i)[r]_{ub}T(k))_{i\in G}$:

$$\mathbf{BT} = \boldsymbol{\mu}, \tag{22}$$

$$\mathbf{BC} = \mathbf{c}, \tag{23}$$

$$\mathbf{B}\boldsymbol{\alpha}(k) = \boldsymbol{\beta}(k),\ \ k \in M, \tag{24}$$

$$\mathbf{BT}' = \boldsymbol{\mu}', \tag{25}$$

$$\mathbf{BC}' = \mathbf{c}', \tag{26}$$

$$\mathbf{BC}'' = \mathbf{c}''. \tag{27}$$

Since X is irreducible, it will exit G with probability 1 from any state $i \in G$ and $\sum_{k\in M}\alpha_i(k) = 1$. Then, only $|M| - 1$ of the $|M|$ linear systems (24) have to be solved. For instance, letting $k_{min} = \min_{k\in M}\{k\}$, it suffices to solve the linear systems:

$$\mathbf{B}\boldsymbol{\alpha}(k) = \boldsymbol{\beta}(k),\ \ k \in M - \{k_{min}\}, \tag{28}$$

and compute $\alpha_i(k_{min})$ using:

$$\alpha_i(k_{min}) = 1 - \sum_{k\in M-\{k_{min}\}} \alpha_i(k). \tag{29}$$

This ends the developments associated to the basic algorithms. For clarity of exposition we give next comprehensive descriptions.

Basic algorithm 1

1. Compute the solution of the linear system (13) using (14), (15).
2. Compute $T(N)$ using (12).
3. Compute $T(k)$, $1 \le k < N$ using (10), (11).
4. Generate the transient CTMC Y_G.
5. Solve the linear systems (22), (23), (28).
6. Compute $\alpha_s(k_{min})$, $s \in S$ using (29).
7. Compute $[T_{U,s}]_{ub}$, $s \in S$ using (3).
8. Compute $[R]_{lb}$, $[R]_{ub}$ using (4), (5).

Basic algorithm 2

1. Compute the solution of the linear system (13) using (14), (15).
2. Compute $T(N)$ using (12).
3. Compute $T(k)$, $1 \le k < N$ using (10), (11).
4. Generate the transient CTMC Y_G.
5. Solve the linear systems (25), (26), (27).
6. Compute $[R]_{lb}$, $[R]_{ub}$ using (6), (7).

3 The decomposition technique

N is typically much smaller than the size of Y_G. Thus, most of the computational effort of the basic algorithms is spent in the generation of Y_G and the solution of the $|M|+1$ linear systems (22), (23), (28) or, respectively, the 3 linear systems (25), (26), (27). The non-zero pattern of the matrix $\mathbf{B}$ of the linear systems is determined by the structure of the state transition diagram of Y_G, and, very often, $\mathbf{B}$ is sparse. This makes attractive the use of iterative methods, which preserve the sparseness of the matrix. Using well-known results from iterative methods theory [13], it is easy to prove that the method converges under Jacobi and Gauss-Seidel if the matrix $\mathbf{B}$ is irreducible. First, note that the sum of the absolute values of the non-diagonal elements of the *ith* row of the matrix $\mathbf{B}$ is $\sum_{j\in G} q_{ij} \le 1$. This implies that $\mathbf{B}$ is diagonally dominant. The strict inequality occurs if and only if X has some transition from i to U. Since X is irreducible, there must be some transition from some state of G to U. Then, the strict inequality is given for at least one row of $\mathbf{B}$ and $\mathbf{B}$ is irreducibly diagonally dominant. This implies the convergence of both Jacobi and Gauss-Seidel [13, Theorem 3.4]. If the matrix $\mathbf{B}$ is reducible we can consider a normal form of $\mathbf{B}$:

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1L} \\ \mathbf{0} & \mathbf{B}_{22} & \cdots & \mathbf{B}_{2L} \\ & & \cdots & \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B}_{LL} \end{pmatrix},$$

where $\mathbf{B}_{ii}$, $1 \le i \le L$, are irreducible. Such a normal can be obtained easily finding the strongly connected components of the restriction of the state transition diagram of X to G and sorting topologically the DAG establishing the connection relationships among the components (see, for instance, [1]). Each component includes the states associated to a diagonal block of the normal form and any topological sorting of the components induces a block upper triangular pattern. This pattern can be exploited to reduce the solution of the linear systems associated to the matrix $\mathbf{B}$ to the solution of linear systems associated to the diagonal blocks $\mathbf{B}_{ii}$. Each $\mathbf{B}_{ii}$ is irreducible and diagonally dominant. The *strict inequality must* also *be satisfied for* some row, since the equality for all rows would imply that the subset of states associated to $\mathbf{B}_{ii}$ would be a trapping subset of X (X will never exit the subset), which is in contradiction with the irreducibility of X. Then, both Jacobi and Gauss-Seidel converge for the linear systems associated to $\mathbf{B}_{ii}$. Thus, whether $\mathbf{B}$ is irreducible or not, we can solve the linear systems (22), (23), (25), (26), (27), (28) using Jacobi or Gauss-Seidel with guaranteed convergence. However, the convergence of the methods is slow in the typical case in which Y_G makes many jumps before exiting G. Several more advanced iterative methods have been proposed to speed up the basic iterative methods (see [7] for a recent review): SOR, block iterative methods, etc. Block iterative methods are specially well-suited for nearly complete decomposable models (see, for instance, [8]), but, Y_G will not have usually such structure. SOR requires the selection of an appropriate value for the overrelaxation parameter, a delicate matter. We have used with success a decomposition technique which works very well when X has in G a frequently visited state. For availability models, the state in which all components are unfailed is such an state.

Without lost of generality let us assume that the frequently visited state has index 1. The components of the vectors $\mathbf{T}$, $\mathbf{C}$, $\boldsymbol{\alpha}(k)$, $k \in M$, $\mathbf{T}'$, $\mathbf{C}'$, and $\mathbf{C}''$ are the expected accumulated rewards to absorption of Y_G with given initial states for different reward rate structures v_i, $i \in G$. The reward rate structure is $v_i = 1$ for $\mathbf{T}$, $v_i = r_i$ for $\mathbf{C}$, $v_i = \lambda_{i,U_k}$ for $\boldsymbol{\alpha}(k)$, $v_i = 1 + \sum_{k\in M}\lambda_{i,U_k}T(k)$ for $\mathbf{T}'$, $v_i = r_i + \sum_{k\in M}\lambda_{i,U_k}[r]_{lb}T(k)$ for $\mathbf{C}'$, and $v_i =$

$r_i + \sum_{k\in M} \lambda_{i,U_k}[r]_{ub}T(k)$ for $\mathbf{C}''$. The assertion is true by definition for $\mathbf{C}$ and easily follows for $\mathbf{T}$, $\boldsymbol{\alpha}(k)$, $\mathbf{T}'$, $\mathbf{C}'$, and $\mathbf{C}''$ comparing (16), (18), (19), (20), and (21) with (17). To describe the decomposition technique, let us consider the generic problem of computing V_i, $i \in G$, where V_i denotes the expected accumulated reward to absorption of Y_G with initial state i for the generic reward rate structure v_i, $i \in G$. Let $\tilde{V}_i$ denote the expected accumulated reward of Y_G with initial state i to either absorption or hit to state 1. Let γ_i denote the probability that Y_G with initial state i will exit G without hitting 1. Decomposing V_i in its two contributions delimited by the time at which Y_G gets absorbed or hits state 1, we obtain:

$$V_i = \tilde{V}_i + (1-\gamma_i)V_1, \quad i \in G. \tag{30}$$

The set of equations (30) can be solved in V_i, $i \in G$, yielding:

$$V_i = \tilde{V}_i + \frac{1-\gamma_i}{\gamma_1}\tilde{V}_1, \quad i \in G. \tag{31}$$

$\tilde{V}_i$ is the expected accumulated reward to absorption of the transient CTMC $\tilde{Y}_G$ obtained from Y_G by directing to the absorbing state the transitions of Y_G to state 1. Then, $\tilde{V}_i$, $i \in G$ can be computed as V_i, $i \in G$, using the matrix $\tilde{\mathbf{B}}$:

$$\tilde{\mathbf{B}} = \begin{pmatrix} 1 & -q_{12} & \cdots & -q_{1,|G|} \\ 0 & 1 & \cdots & -q_{2,|G|} \\ & & \cdots & \\ 0 & -q_{|G|,2} & \cdots & 1 \end{pmatrix}$$

instead of $\mathbf{B}$. We can also note that γ_i and $\alpha_i'(k)$, $k \in M$ are not independent. $\alpha_i'(k)$ is the expected accumulated reward to absorption of $\tilde{Y}_G$ with reward rate structure $v_i = \lambda_{i,U_k}$ and γ_i is the expected accumulated reward to absorption of the same CTMC with reward rate structure $v_i = \lambda_{i,U} = \sum_{k\in M}\lambda_{i,U_k}$. Then:

$$\gamma_i = \sum_{k\in M} \alpha_i'(k), \tag{32}$$

Equation (32) can be used to compute γ_i, $i \in G$ and V_i, $i \in G$ can be computed from $\tilde{V}_i$, $i \in G$ and γ_i, $i \in G$ using (31). The new algorithms incorporating the decomposition technique can be described as follows:

Algorithm 1

1. Compute the solution of the linear system (13) using (14), (15).
2. Compute $T(N)$ using (12).
3. Compute $T(k)$, $1 \le k < N$ using (10), (11).
4. Generate the transient CTMC $\tilde{Y}_G$.
5. Let the column vectors $\tilde{\mathbf{T}} = (\tilde{T}_{G,i})_{i\in G}$, $\tilde{\mathbf{C}} = (\tilde{C}_{G,i})_{i\in G}$, $\tilde{\boldsymbol{\alpha}}(k) = (\tilde{\alpha}_i(k))_{i\in G}$. Solve the linear systems:
$$\tilde{\mathbf{B}}\tilde{\mathbf{T}} = \boldsymbol{\mu},$$
$$\tilde{\mathbf{B}}\tilde{\mathbf{C}} = \mathbf{c},$$
$$\tilde{\mathbf{B}}\tilde{\alpha}(k) = \beta(k), k \in M.$$
6. Compute γ_s, $s \in S \cup \{1\}$ using (32).
7. Compute $T_{G,s}$, $C_{G,s}$, and $\alpha_s(k)$, $s \in S$, $k \in M$ using:
$$T_{G,s} = \tilde{T}_{G,s} + \frac{1-\gamma_s}{\gamma_1}\tilde{T}_{G,1},$$
$$C_{G,s} = \tilde{C}_{G,s} + \frac{1-\gamma_s}{\gamma_1}\tilde{C}_{G,1},$$
$$\alpha_s(k) = \tilde{\alpha}_s(k) + \frac{1-\gamma_s}{\gamma_1}\tilde{\alpha}_1(k).$$
8. Compute $[T_{U,s}]_{ub}$, $s \in S$ using (3).
9. Compute $[R]_{lb}$, $[R]_{ub}$ using (4), (5).

Algorithm 2

1. Compute the solution of the linear system (13) using (14), (15).
2. Compute $T(N)$ using (12).
3. Compute $T(k)$, $1 \le k < N$ using (10), (11).
4. Generate the transient CTMC $\tilde{Y}_G$.
5. Let the column vectors $\tilde{\mathbf{T}}' = (\tilde{T}_i')_{i\in G}$, $\tilde{\mathbf{C}}' = (\tilde{C}_i')_{i\in G}$, $\tilde{\mathbf{C}}'' = (\tilde{C}_i'')_{i\in G}$. Solve the linear systems:
$$\tilde{\mathbf{B}}\tilde{\mathbf{T}}' = \boldsymbol{\mu}',$$
$$\tilde{\mathbf{B}}\tilde{\mathbf{C}}' = \mathbf{c}',$$
$$\tilde{\mathbf{B}}\tilde{\mathbf{C}}'' = \mathbf{c}'',$$
6. Compute γ_s, $s \in S \cup \{1\}$ using (32).
7. Compute T_s', C_s', and C_s'', $s \in S$ using:
$$T_s' = \tilde{T}_s' + \frac{1-\gamma_s}{\gamma_1}\tilde{T}_1',$$
$$C_s' = \tilde{C}_s' + \frac{1-\gamma_s}{\gamma_1}\tilde{C}_1',$$
$$C_s'' = \tilde{C}_s'' + \frac{1-\gamma_s}{\gamma_1}\tilde{C}_1'',$$
8. Compute $[R]_{lb}$, $[R]_{ub}$ using (6), (7).

Now the algorithms require the solution of, respectively, $|M|+2$ and 4 linear systems of size of G. The matrix $\tilde{\mathbf{B}}$ also has the properties which guarantee the convergence of Jacobi and Gauss-Seidel, but, since 1 is an often visited state, $\tilde{Y}_G$ moves fast to its absorbing state, the convergence for the new linear systems is much better and the new algorithms are typically much faster. Algorithm 1 involves less linear systems than algorithm 2 when $|M| = 1$ and more when $|M| > 2$. Thus, algorithm 1 should be used when $|M| = 1$ and algorithm 2 should be used when $|M| > 2$. For $|M| = 2$ both algorithms involve 4 linear systems and should have similar performance.

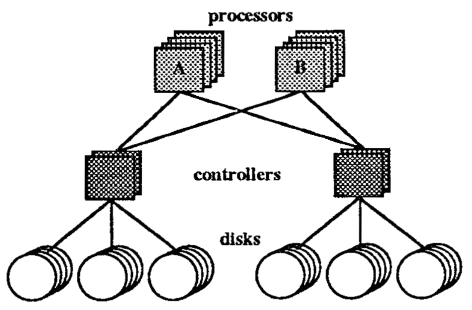

Figure 4: Block diagram of the fault-tolerant system of the example.

4 Application example

In this section we illustrate the performance of the new bounding algorithms using an availability model of a repairable fault-tolerant system. The example is a variant of the large example used in [11]. Figure 4 shows the block diagram of the system. The system is operational (up) if at least one processor of type A or B is unfailed, at least one controller of each set is unfailed, and at least three disks of each disk cluster are unfailed. Only one processor of each set is active. Non-active processors do not fail. A fault in the active processor A is propagated to the active processor B with probability 0.10. Active processors and controllers of one set fail with rate 1/2,000. Controllers of the other set fail with rate 1/4,000. Disks fail with a different rate for each cluster. The disk failure rates are 1/6,000, 1/8,000, 1/10,000, 1/12,000, 1/14,000 and 1/16,000. Components can fail in two modes with equal probabilities. There is only one repairman which selects the component to be repaired at random from the set of failed components. The repair rate of a component depends on the failure mode of the component and on the operational/down state of the system. When the system is operational repair rates are 0.1 in one failure mode and 0.05 in the other failure mode. The repair rates are 10 times larger when the system is down. The difference in repair rates between the operational and down states of the system can be due to more careful repair procedures in the operational state to avoid system crashes as a consequence of erroneous maintenance operations. The system has a moderate complexity (36 components of 10 different types) but a very large state space: of the order of 10^{10} states. The dependencies introduced by failure propagation, repair queuing, and impact of the system operational/down state on the repair rates preclude the computation of the steady-state unavailability by combinatoric techniques. The size of the state space precludes an exact numerical solution of the Markov model. Thus, the example illustrates the type of models for which bounding methods are an attractive approach.

We apply the bounding algorithms including in G all the states with up to K failed components. U is partitioned according to the number of failed components, i.e., U_k includes all states with $K+k$ failed components. We take $f_i(k)$ $(i = 1, 2)$ equal to the sum of the maximum rates of all failure events of the model involving i components and $g(k)$ equal to slowest repair rate of the model. This yields $f_1(k) = 4.936 \times 10^{-3}$, $f_2(k) = 5 \times 10^{-5}$, and $g(k) = 0.05$. The bounding portions of the CTMC's used in the algorithm described in [11] are built using the same upper bounds for failure rates and lower bound for repair rates.

Table 1: CPU times in seconds (number of iterations) for algorithm 1, basic algorithm 1, and algorithm 2.

K	algorithm 1	basic alg. 1	algorithm 2
2	0.28 (41)	13.0 (9,968)	0.27 (41)
3	2.18 (54)	1,529 (113,410)	2.15 (53)
4	17.2 (62)	$> 10,000$	17.0 (61)
5	106 (67)	$> 10,000$	104 (65)

For the example, $|M| = 2$ and both algorithms with the decomposition technique solve 4 linear systems of size $|G|$. Table 1 compares algorithm 1 with the basic algorithm 1 (without the decomposition technique) and algorithm 2 for several values of K (the sizes $|G|$ of the generated models are given in Table 2). We give the CPU times and total number of iterations required for the solution of the linear systems (matrix $\tilde{\mathbf{B}}$ is irreducible). The decomposition technique is applied selecting the state with all components unfailed as the frequently visited state. Gauss-Seidel is used to solve all linear systems, with a convergence criterium of a relative tolerance in all components of the solution between two iterations smaller than 10^{-8}. The CPU times include the generation of the Markov models and were measured in a SPARC-10 workstation. Generation times were about 50% of the total CPU times of our algorithms with the decomposition technique. The results clearly illustrate the efficiency of the decomposition technique and the fact that algorithms 1 and 2 have very similar performances for the case $|M| = 2$. Table 2 shows the bounds, CPU times, and total number of iterations required for the solution of the linear systems for the algorithm described in [11] with $(F < K)$ and without $(F = K)$ state cloning and our algorithm 1 for several values of K. For the algorithm described in [11] and $|S| > 1$ we use the solution of a linear system as a first iterate for the solution of the linear system associated to the next state in S. We also give the size $|G|$ of the generated state space and the number of linear systems $|S|$ which are solved in the algorithm proposed in [11]. Our algorithm 1 gives exactly the same bounds as that algorithm without state cloning in much smaller CPU times and slightly smaller CPU times than that algorithm with $F = 0$, which gives significantly less tighter bounds. The smaller CPU times of our algorithm are not only due to a smaller number of iterations, but also to a smaller cost per iteration. This smaller cost comes from the fact that the matrix $\tilde{\mathbf{B}}$ does not include neither the bounding submodel Y'_U nor the elements corresponding to the transition rates λ_{i,U_k} and $\lambda_{i,1}$, $i \in G$ which are included in the transition rate matrices of the models X'_s solved in the bounding algorithm described in [11].

Table 2: Bounds, CPU times in seconds and number of iterations (n_I) for the algorithm given in [11] and algorithm 1.

						CPU time (n_I)	
K	F	$\lvert G\rvert$	$\lvert S\rvert$	lower bound	upper bound	[11]	algorithm 1
2	0	265	1	2.9965×10^{-5}	1.7579×10^{-3}	0.39 (79)	
	1		55	2.9972×10^{-5}	1.1930×10^{-3}	1.41 (496)	
	2		209	2.9972×10^{-5}	6.7213×10^{-4}	11.2 (5,001)	0.28 (41)
3	0	1,796	1	3.5522×10^{-5}	1.9900×10^{-4}	3.04 (77)	
	1		55	3.5524×10^{-5}	1.5832×10^{-4}	10.5 (439)	
	2		209	3.5526×10^{-5}	1.2123×10^{-4}	95.3 (4,485)	
	3		1,531	3.5526×10^{-5}	8.4473×10^{-5}	701 (35,276)	2.18 (54)
4	0	10,496	1	3.6233×10^{-5}	4.9960×10^{-5}	24.3 (76)	
	1		55	3.6233×10^{-5}	4.7222×10^{-5}	83.2 (379)	
	2		209	3.6233×10^{-5}	4.4729×10^{-5}	730 (3,876)	
	3		1,531	3.6233×10^{-5}	4.2237×10^{-5}	5872 (31,042)	
	4		8,700	3.6233×10^{-5}	3.9768×10^{-5}	$> 10,000$	17.2 (62)
5	0	51,391	1	3.6306×10^{-5}	3.7359×10^{-5}	137 (75)	
	1		55	3.6306×10^{-5}	3.7184×10^{-5}	396 (305)	
	2		209	3.6306×10^{-5}	3.7024×10^{-5}	3,535 (3,148)	
	5		40,895	3.6306×10^{-5}	3.6542×10^{-5}	$> 10,000$	106 (67)

5 Conclusions

We have proposed two new algorithms for the computation of bounds for the steady-state reward rate of rewarded CTMC models with slow forward and fast backward transitions. Availability models with failure and repair transitions are an example of that class of models and we have illustrated the performance of the method with an example of that type. The algorithms compare favorably with previous algorithms. They achieve the same bounds with, typically, much smaller CPU times and give tighter bounds than previous methods incorporating state cloning techniques with slightly smaller CPU times.

Acknowledgements

The authors are grateful to the helpful comments of the anonymous reviewers, which helped to improve the presentation.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1983.

[2] J. A. Carrasco, "Failure distance-based Simulation of Repairable Fault-Tolerant Systems," in *Computer Performance Evaluation. Modelling Techniques and Tools*, G. Balbo and G. Serazzi eds., Elsevier, pp. 351–366.

[3] J.A. Carrasco, "Improving Availability Bounds using the Failure Distance Concept," in *Dependable Computing for Critical Applications vol. 9*, Springer-Verlag, 1995, pp. 479–497.

[4] E. Çinlar, *Introduction to Stochastic Processes*, Prentice-Hall, 1975.

[5] A. Goyal, W.C. Carter, E. de Souza e Silva and S.S. Lavenberg, "The System Availability Estimator," in *Proc. 16th IEEE Int. Symp. on Fault-Tolerant Computing FTCS-16*, Vienna, July 1986, pp. 84–89.

[6] A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola, and P.W. Glynn, "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems," *IEEE Trans. on Computers*, vol. 41, no. 1, pp. 36–51, January 1992.

[7] U.R. Krieger, B. Müller-Clostermann, and M. Sczittnick, "Modeling and Analysis of Communication Methods for Markov Chains," *IEEE J. on Selected Areas in Communications*, vol. 8, no. 9, pp. 1630–1648, December 1990.

[8] J.R. Koury, D.F. McAllister, and W.J. Stewart, "Iterative Methods for Computing Stationary Distributions of Nearly Completely Decomposable Markov Chains," *SIAM J. on Algebraic and Discrete Methods*, vol. 5, pp. 164–185, 1984.

[9] J.C.S. Lui and R.R. Muntz, "Evaluating Bounds on Steady State Availability from Markov Models of Repairable Systems," in *Numerical Solution of Markov Chains*, William J. Stewart, ed., Marcel Dekker, New York, 1991, pp. 435-454.

[10] J.C.S. Lui and R.R. Muntz, "Computing Bounds on Steady State Availability of Repairable Computer Systems," *Journal of the ACM*, vol. 41, no. 4, July 1994, pp. 676–707.

[11] R.R. Muntz, E. De Souza e Silva and A. Goyal, "Bounding availability of Repairable computer systems," *IEEE Trans. on Computers*, vol. 38, no. 12, December 1989, pp. 1714–1723.

[12] E. de Souza e Silva and P.M. Ochoa, "State Space Exploration in Markov Models," *Performance Evaluation Review*, vol. 20, no. 1, June 1992, pp. 152–166.

[13] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, 1962.