Efficient Exploration of Availability Models Guided by Failure distances*

Juan A. Carrasco, Javier Escribá and Angel Calderón
Departament d'Enginyería Electrònica
Universitat Politécnica de Catalunya
Diagonal 647, plta. 9, 08028 Barcelona, Spain
email: carrasco@eel.upc.es

Abstract

Recently, a method to bound the steady-state availability using the failure distance concept has been proposed. In this paper we refine that method by introducing state space exploration techniques. In the methods proposed here, the state space is incrementally generated based on the contributions to the steady-state availability band of the states in the frontier of the currently generated state space. Several state space exploration algorithms are evaluated in terms of bounds quality and memory and CPU time requirements. The more efficient seems to be a waved algorithm which expands transition groups. We compare our new methods with the method based on the failure distance concept without state exploration and a method proposed by Souza e Silva and Ochoa which uses state space exploration but does not use the failure distance concept. Using typical examples we show that the methods proposed here can be significantly more efficient than any of the previous methods.

1 Introduction

The steady-state availability is an important dependability measure for non mission-oriented repairable systems with long lifetimes. Several techniques have been proposed to analyze that measure: combinatoric analysis [1], closed-form solution queue networks [10], and stochastic processes. Combinatoric analysis and closed-form solution queue networks are very efficient techniques but have a relatively small scope of application. In particular, they cannot support in a general enough way important dependencies in the components' behavior which realistic availability models have to consider. These dependencies are caused by imperfect fault coverage, impact of the system configuration on fault, fault/error handling, and repair processes, and repair queuing. All these dependencies can be incorporated using stochastic processes [6]. Stochastic processes models can be solved using either numerical methods or simulation. Numerical methods provide a reliable solution of the model, but suffer from the well-known "state space explosion problem" and difficulties in dealing with non-exponential distributions. Simulation with appropriate importance sampling techniques [4], [12] can be made efficiently and can encompass non-exponential distributions [19], [20], but only provides an statistical assessment of the accuracy.

This paper is concerned with numerical solutions of Markov availability models. State pruning is an approach which has been shown very effective to alleviate the state space explosion problem. With state pruning, an approximated solution of the model is obtained using a portion of the complete state space. Of particular interest are the bounding methods recently developed which also give bounds for the measure. Using general results from Courtois and Semal [7], [8], Muntz, Souza e Silva and Goyal proposed the first bounding method for the steady-state availability [18]. In that method, all states with up to a given number of failed components, K, are generated and the behavior in the non-generated state space is modeled by a bounding submodel in which all states with the same number of failed components are aggregated. A problem of the method is that it requires the solution of a model for each return state of the generated state space G. In order to keep the number of return states small, a state cloning technique is proposed in [18], in which clones of all states of G with more than F failed components are added to the non-generated portion and aggregated in the bounding submodel. This technique introduces some looseness in the bounds. Based on [18], Lui and Muntz have proposed an adaptive method [15] which increments K till the specified error bound is achieved and reuses the solution obtained with smaller state spaces to avoid having to solve the complete model each time K is incremented. This reuse technique introduces some additional looseness in the bounds. The looseness has been reduced in a recent, more elaborated version of the method [16]. Another interesting elaboration of the original bounding method proposed in [18], which is more related to the work reported here, is the method developed by Souza e Silva and Ochoa [21]. In that method F is set to 0 and the state space is generated incrementally by expanding states in the frontier of the currently generated state space. Two heuristics for the selection of the state to be expanded are proposed in [21]. The more efficient of them selects the state with maximum mean time in each visit of the model to the currently generated state space. State space exploration techniques have also been proposed for combinatoric availability models [9], [13], [14], [22] and probabilistic verification of communication protocols [17]. Recently [5], it has been proposed a method to bound the steady-state availability which differs from the method proposed in [18] in the use of the failure distance concept to bound the behavior out of the generated state space less pessimistically than in [18].

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

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS '96 5/96 PA, USA
© 1996 ACM 0-89791-793-6/96/0005...$3.50

In this paper we develop new state space exploration algorithms specifically targeted to the bounding method described in [5]. As in [21], we take $F = 0$ so that G has only one return state, and generate incrementally the state space based on *approximated contributions* to the steady-state availability band associated to the states in the frontier of the currently generated state space. As in the methods proposed in [21] our basic state space exploration algorithms have time requirements which grow quadratically with the number of generated states. This is problematic when tens of thousands of states have to be generated to achieve the desired accuracy. In order to reduce the time requirements we propose less precise but less costly state exploration algorithms, which, as our examples illustrate, perform almost as well as the more costly algorithms. Overall, the bounding methods proposed here seem to be significantly more efficient than both the bounding method proposed in [21] and the failure distance based method without state space exploration described in [5]. The rest of the paper is organized as follows. Section 2 summarizes the basic bounding method described in [5] with the detail required to understand our state space exploration algorithms. Section 3 describes the state exploration algorithms. Section 4 discusses implementation details which are important from a computational efficiency point of view. Section 5 evaluates the state space exploration algorithms and compares the resulting bounding methods with the method proposed in [21] and the basic method without state space exploration described in [5]. Section 6 concludes the paper.

2 Preliminaries

We consider finite and irreducible (thus, ergodic) CTMC availability models $X = \{X(t); t \geq 0\}$ of repairable fault-tolerant systems made up of components with transitions modeling failures and repairs. We assume that: 1) X has a single state o without failed components, 2) repair transitions involve one component (failure transitions are allowed to involve more than one component), 3) there is at least one outgoing repair transition from every state $\neq o$ (i.e., at least one component is repaired whenever some component is failed), and 4) the operational/down state of the system is determined by the unfailed/failed states of its components through a coherent structure function [1]. The bounding method described in [5] uses the concept of failure event. A *failure event* is a bag of components which can fail simultaneously. We assume known: a) the failure events of the model, b) for each failure event e, an upper bound $\lambda_{ub}(e)$ for the sum of the rates of all failure transitions involving exactly the components in e from any state of the model, and c) a lower bound $g > 0$ for the repair rate from any state $\neq o$ of the model. We will denote by E the set of failure events of the model and by E_i the set of failure events of the model with cardinality i. The class of models under consideration is quite large and encompasses, for instance, all the models which can be specified with the SAVE modeling language [11]. Let Ω be the state space of X and let D be the subset of down states of X. Since X is ergodic, $p_i = \lim_{t\to\infty} P[X(t) = i]$ is well-defined. The steady-state availability is given by:

$$A = \sum_{i \in \Omega - D} p_i.$$

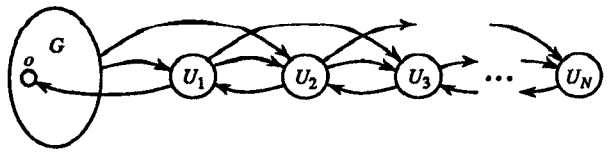

Figure 1: Structure of the modified CTMC X.

Since A is typically very close to 1, we will use the steady-state unavailability:

$$UA = 1 - A = \sum_{i \in D} p_i.$$

Let G and U be, respectively, the generated and non-generated portion of Ω. The bounding method proposed in [5] with $F = 0$ can be justified by considering the regenerative behavior of a modified CTMC X, taking as regeneration points the times at which the modified X enters o from U. The modified X is obtained from the original X by adding to U clones of the states $s \in G - \{0\}$, accounting for the visits to the cloned states between exit from G and hit to o. The modified X has the structure depicted in Figure 1, where U_k includes the states in U with exactly k failed components and, because of assumption 2), X has a nearest neighbor structure in U. In the following we will refer to the modified X simply as X. Also, we will denote by λ_{ij} the transition rate from state i to state j, by $\lambda_i = \sum_j \lambda_{ij}$ the output rate of state i, by $\lambda_{i,B} = \sum_{j \in B} \lambda_{ij}$ the transition rate from state i to the subset of states B, and by $I(c)$ the indicator function returning value 1 if c is true and 0 if c is false.

Let T_G and T_U be the contributions of G and U to the mean time between regenerations of X; let C_G and C_U be the respective contributions to the mean down time. Using regenerative theory [6], we can write:

$$UA = \frac{C_G + C_U}{T_G + T_U}.$$

Assume that upper bounds for T_U and C_U, $[T_U]_{ub}$ and $[C_U]_{ub}$, are available. The bounding method described in [5] uses the following bounds for UA:

$$[UA]_{lb} = \frac{C_G}{T_G + [T_U]_{ub}}, \tag{1}$$

$$[UA]'_{ub} = \frac{C_G + [C_U]_{ub}}{T_G + [C_U]_{ub}}. \tag{2}$$

The lower bound (1) is as in [18]; the upper bound (2) is different from the one used in [18], which is:

$$[UA]_{ub} = \frac{C_G + [T_U]_{ub}}{T_G + [T_U]_{ub}}.$$

It is easy to prove [5] that $[UA]'_{ub} \leq [UA]_{ub}$ if $[C_U]_{ub} \leq [T_U]_{ub}$ and that $[UA]'_{ub} < [UA]_{ub}$ if $[C_U]_{ub} < [T_U]_{ub}$. A $[C_U]_{ub} \leq [T_U]_{ub}$ and, typically, $< [T_U]_{ub}$ is developed in [5] using the failure distance concept as will be explained next.

T_G and C_G can be computed from the mean times to absorption vector of the transient CTMC Y_G with initial state o obtained by directing to the absorbing state the transitions of X from states in G to U. Denoting by $\mathbf{A}_G = (a_{ij})_{i,j \in G}$ the

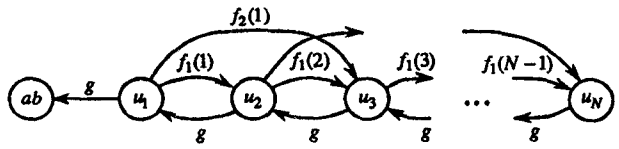

Figure 2: State transition diagram of the bounding submodel Y'_U.

restriction of the transition rate matrix of X to G ($a_{ij} = \lambda_{ij}$, $i \neq j$, $a_{ii} = -\lambda_i$) and by $\mathbf{q} = (\delta_{io})_{i \in G}$ the initial probability row vector of Y_G, the mean times to absorption row vector of Y_G, $\tau = (\tau_i)_{i \in G}$, can be obtained by solving:

$$\tau \mathbf{A}_G = -\mathbf{q}. \tag{3}$$

From τ, T_G and C_G can be obtained as:

$$T_G = \sum_{i \in G} \tau_i, \tag{4}$$

$$C_G = \sum_{i \in G \cap D} \tau_i. \tag{5}$$

Let N denote the number of components of the system. The upper bound $[T_U]_{ub}$ is computed using the same submodel, Y'_U, as in [18], shown in Figure 2, where each aggregate state u_k accounts for the subset U_k and $f_i(k) = I(k+i \leq N) \sum_{e \in E_i} \lambda_{ub}(e)$. Let $T(k)$ denote the mean time to absorption of Y'_U with initial state u_k (see [5] for an efficient computational procedure). The probability that X exits G following a transition from $i \in G$ to some state in U_k can be computed as:

$$\psi_{i,k} = \tau_i \lambda_{i,U_k}. \tag{6}$$

Let

$$\pi_k = \sum_{i \in G} \psi_{i,k} \tag{7}$$

be the conditional exit distribution from G of X through the subsets U_k. Then:

$$[T_U]_{ub} = \sum_k \pi_k T(k). \tag{8}$$

The procedure to compute $[C_U]_{ub}$ is based on the failure distance concept. The failure distance from a state s, $d(s)$, is defined as the minimum number of components which have to fail in addition to those already failed in s to take the system down. For down states, $d(s) = 0$. For operational states, $d(s)$ can be computed assuming the knowledge of the minimal cuts [1] of the structure function of the system [1] as follows. Let MC be the set of minimal cuts and let us denote by $F(s)$ the bag of failed components in s. Since a state s' is down if and only if $m \subset F(s')$ for some $m \in MC$:

$$d(s) = \min_{m \in MC} |m - F(s)|. \tag{9}$$

Let $U_{k,d}$ be the subset of U including the states with k failed components and failure distance d. $[C_U]_{ub}$ is obtained from upper bounds $C(k,d)$ for C_U conditioned to exit of G through $U_{k,d}$. The probability that X exits G following a transition from i to some state in $U_{k,d}$ can be computed as:

$$\psi_{i,k,d} = \tau_i \lambda_{i,U_{k,d}}. \tag{10}$$

Let

$$\pi_{k,d} = \sum_{i \in G} \psi_{i,k,d} \tag{11}$$

be the conditional exit distribution from G through the subsets $U_{k,d}$. Then:

$$[C_U]_{ub} = \sum_{k,d} \pi_{k,d} C(k,d). \tag{12}$$

The bounds $C(k,d)$ are obtained using an iterative improvement procedure which starts with $C(k,d) = C(k)$, where $C(k)$ upper bounds C_U conditioned to exit of G through U_k. Let $L = d(o)$ be the redundancy level of the system, i.e., the minimum number of components which have to be failed for the system to be down. Then, $C(k)$ is computed as the mean reward to absorption of Y'_U with initial state u_k with the reward rate structure $r(u_k) = I(k \geq L)$. At each iterative step, the $C(k,d)$ bounds are revised following increasing values of k and, for each k, increasing values of d. For each (k,d) pair, a new bound $C'(k,d)$ is computed using the available $C(k,d)$'s and $C'(k,d)$ is accepted as new $C(k,d)$ if $C'(k,d) < C(k,d)$. The iterations continue till no bound $C(k,d)$ experiences significant improvement. The bounds $C'(k,d)$ are computed using bounding failure rate structures $f_{i,j}(k,d)$ which are defined as follows. Let FC be the set of different cardinalities of the failure events of the model and let $F(k,d,i,r)$, $i \in FC$ be upper bounds for the sum of failure transition rates involving i components from any state in $U_{k,d}$ to states with failure distance $\leq r$. Let $w = \min\{i,d\}$, then $f_{i,j}(k,d)$ are defined by:

$$f_{i,j}(k,d) = F(k,d,i,d-j) - F(k,d,i,d-j-1), \quad 0 \leq j < w,$$

$$f_{i,w}(k,d) = F(k,d,i,d-w).$$

Using $f_{i,j}(k,d)$, $C'(k,d)$ is computed as:

$$C'(k,d) = \frac{I(d=0)}{g} + \max\{C(k-1,d), C(k-1,d+1)\} + \frac{1}{g} \sum_{i \in FC} \sum_{j=0}^{w} f_{i,j}(k,d) C(k+i, d-j).$$

The iterative procedure considers only feasible (k,d) pairs, i.e., pairs satisfying $1 \leq k \leq N$, $\max\{0, L-k\} \leq d \leq \min\{L, N-k\}$. Also, in the expression for $C'(k,d)$, $C(k',d') = 0$ for unfeasible pairs (k',d') in the right-hand side.

The correctness of the procedure to obtain the bounds $C(k,d)$ was proved [5] assuming $F(k,d,i,r)$ decreasing on d. Bounds $F(k,d,i,r)$ satisfying the requirement which can be computed with moderate effort are obtained in [5] using the concepts of importance $Imp(e)$ and activity $Act(e)$ of a failure event, defined as:

$$Imp(e) = \min_{m \in MC, m \cap e \neq \phi} |m - e|,$$

$$Act(e) = \max_{m \in MC} |m \cap e|.$$

[1] A minimal cut is a minimal collection of components whose failure implies the failure (down state) of the system. We allow component classes with indistinguishable instances and, thus, collections of components are represented as bags.

Then:

$$F(k,d,i,r) = \sum_{e\in E_i, A(e)\geq d-r, I(e)\leq k+r} \lambda_{ub}(e), \quad r < d,$$

$$F(k,d,i,d) = \sum_{e\in E_i} \lambda_{ub}(e).$$

3 State space exploration algorithms

The more efficient state space exploration algorithm proposed in [21] selects for expansion the state with the largest mean time to absorption in the transient CTMC Y_G. For the method reviewed in the last section, the large impact of the failure distance parameter d on $C(k,d)$ suggests the use of more sophisticated state exploration techniques. We start by finding an approximated expression for the unavailability band $[UA]_{band} = [UA]'_{ub} - [UA]_{lb}$ and approximate the band as a sum of contributions associated to the states in the frontier of G.

Using (1), (2):

$$\begin{aligned}[UA]_{band} &= [UA]'_{ub} - [UA]_{lb} \\ &= \frac{C_G + [C_U]_{ub}}{T_G + [C_U]_{ub}} - \frac{C_G}{T_G + [T_U]_{ub}} \\ &= \frac{C_G/T_G + [C_U]_{ub}/T_G}{1 + [C_U]_{ub}/T_G} - \frac{C_G/T_G}{1 + [T_U]_{ub}/T_G}. \end{aligned} \quad (13)$$

Assuming $[T_U]_{ub} \ll T_G$ and noting that $[C_U]_{ub} \leq [T_U]_{ub}$, we can approximate (13) by its first-order Taylor expansion on the variables $[T_U]_{ub}/T_G$, $[C_U]_{ub}/T_G$, obtaining:

$$\begin{aligned}[UA]_{band} &\approx [UA]_{band,app} \\ &= \frac{C_G}{T_G^2}[T_U]_{ub} + \frac{T_G - C_G}{T_G^2}[C_U]_{ub}. \end{aligned} \quad (14)$$

Using, respectively, (6), (7), (8) and (10), (11), (12) we obtain:

$$[T_U]_{ub} = \sum_{i\in G} \tau_i \sum_k \lambda_{i,U_k} T(k), \quad (15)$$

$$[C_U]_{ub} = \sum_{i\in G} \tau_i \sum_{k,d} \lambda_{i,U_{k,d}} C(k,d). \quad (16)$$

Then, combining with (14), noting that $\lambda_{i,U_k} = \sum_d \lambda_{i,U_{k,d}}$:

$$\begin{aligned}&[UA]_{band,app} \\ &= \sum_{i\in G} \tau_i \sum_{k,d} \lambda_{i,U_{k,d}} \left[\frac{C_G}{T_G^2} T(k) + \frac{T_G - C_G}{T_G^2} C(k,d)\right] \\ &= \sum_{i\in G} \tau_i \alpha_i = \sum_{i\in G,k,d} \tau_i \beta_i(k,d), \end{aligned}$$

with:

$$\alpha_i = \sum_{k,d} \lambda_{i,U_{k,d}} \left[\frac{C_G}{T_G^2} T(k) + \frac{T_G - C_G}{T_G^2} C(k,d)\right], \quad (17)$$

$$\beta_i(k,d) = \lambda_{i,U_{k,d}} \left[\frac{C_G}{T_G^2} T(k) + \frac{T_G - C_G}{T_G^2} C(k,d)\right]. \quad (18)$$

Thus, we can approximate the unavailability band as a sum of contributions associated to either the states in the frontier of G ($\tau_i\alpha_i$) or transition groups from states in the frontier of G to states in subsets $U_{k,d}$ ($\tau_i\beta_i(k,d)$). This is the basis for the state space exploration algorithms.

The availability models considered in this paper have a probability distribution which typically is highly skewed according to the parameter k (number of failed components), and the sum of the mean times to absorption τ_j of the states reached from a given state i through failure transitions is typically much smaller than τ_i. Then, the impact on the unavailability band of the expansion of a state i or a transition group (i,k,d) is almost the removal of the contribution due to that state or transition group. This justifies the basic state exploration algorithms CONT_S and CONT_TG. In these algorithms, G is generated incrementally starting with the state o by selecting the state in G or transition group from G to U with the highest $\tau_i\alpha_i$ or $\tau_i\beta_i(k,d)$, respectively. The algorithms are designed so that all transitions between generated states are included in the generated portion of X. The state expansion process ends when either a relative band $rb = ([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$ smaller than or equal to *req_rb* is achieved or a number of states greater than or equal to *max_n_states* has been generated. Denoting by T the subset of transitions included in the generated portion of X, the algorithms can be described as follows. Note that C_G and (1) $[UA]_{lb}$ could be 0. This will give $rb = \infty$, a value greater than any *req_rb*.

Algorithm CONT_S(*req_rb, max_n_states*)

Compute $T(k)$'s and $C(k,d)$'s;
$G = \{o\}$, $T = \phi$, $\tau_o = 1/\lambda_o$, $T_G = \tau_o$, $C_G = 0$;
Compute $\tau_o\alpha_o$ using (17);
Compute $rb = ([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$
using (15), (16), (1), (2);
while ($rb >$ *req_rb* && *n_states* $<$ *max_n_states*){
  Pick the state s with largest $\tau_s\alpha_s$;
  Let B be the set of successors of s non included in G;
  for (each $s' \in B$){
    Add to T all transitions from s' to G;
    Add to T all transitions from G to s';
    $G = G \cup \{s'\}$;
  }
  Compute τ_i, $i \in G$ solving (3) and T_G, C_G using (4), (5);
  Compute $\tau_i\alpha_i$, $i \in G$ using (17);
  Compute $rb = ([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$
  using (15), (16), (1), (2);
}

Algorithm CONT_TG(*req_rb, max_n_states*)

Compute $T(k)$'s and $C(k,d)$'s;
$G = \{o\}$, $T = \phi$, $\tau_o = 1/\lambda_o$, $T_G = \tau_o$, $C_G = 0$;
Compute $\tau_o\beta_o(k,d)$'s using (18);
Compute $rb = ([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$
using (15), (16), (1), (2);
while ($rb >$ *req_rb* && *n_states* $<$ *max_n_states*){
  Pick the transition group (s,k,d) with largest $\tau_s\beta_s(k,d)$;
  Let B be the set of successors of s with k failed
  components and failure distance d non included in G;
  for (each $s' \in B$){
    Add to T all transitions from s' to G;
    Add to T all transitions from G to s';
    $G = G \cup \{s'\}$;
  }
  Compute τ_i, $i \in G$ solving (3) and T_G, C_G using (4), (5);
  Compute $\tau_i\beta_i(k,d)$'s, $i \in G$ using (18);
  Compute $rb = ([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$
  using (15), (16), (1), (2);
}

Note that in both algorithms, the linear system (3) is solved at each expansion stage. This is somehow desirable because a expansion stage may affect the contributions of unexpanded states in the frontier of G. Having refreshed values of the band contributions after each expansion stage guarantees an accurate implementation of the expansion heuristic, but compromises the time efficiency of the algorithms. Being r the average number of states included in G at each expansion stage and m the average number of transitions from the states in G during the expansion process, the solution of the linear systems (3) adds a term to the overall time complexity of the algorithm of order $O(|G|^2 m/r)$. This term is dominating and makes the algorithm very slow in practice when G has of the order of thousands of states. An approach to alleviate the problem is to reorganize the expansion so that it is accomplished in waves. This means that several states or transition groups are expanded without recomputing the vector τ, but taking care of changes in the contributions $\tau_i\alpha_i$, $\tau_i\beta_i(k,d)$ resulting from changes in the transition rates $\lambda_{i,U_{k,d}}$. These revisions do not introduce a significant overhead because they have time complexity similar to the updates of the transition set T. At an extreme, we can include in each wave the state or transition group expansions which are estimated necessary to reduce the relative band rb below the required one req_rb. We can also be more conservative and allow only a band reduction up to BR times the band before the wave, where $0 \leq \text{BR} < 1$. Lower values of BR result in less solutions of linear systems (3) but less accurate state space explorations. Higher values of BR give potentially more accurate state space explorations, but involve more linear system solutions. To clarify the discussion we show below the waved version of CONT_S, called CONT_S_W. A similar version of CONT_TG, CONT_TG_W, will be considered.

Algorithm CONT_S_W(*req_rb, max_n_states*)

```
Compute T(k)'s and C(k,d)'s;
G = {o}, T = φ, τ_o = 1/λ_o, T_G = τ_o, C_G = 0;
Compute τ_o α_o using (17);
Compute rb = ([UA]'_ub − [UA]_lb)/[UA]_lb
using (15), (16), (1), (2);
while (rb > req_rb && n_states < max_n_states){
    Let appb be the current sum of
    all band contributions τ_i α_i;
    target_appb = max{BR * appb, (req_rb/rb) * appb};
    while (appb > target_appb
          && n_states < max_n_states){
        Pick the state s with largest τ_s α_s;
        Let B be the set of successors of s
        non included in G;
        for (each s' ∈ B){
            Add to T all transitions from s' to G
            updating α_s' and appb;
            Add to T all transitions from G to s'
            updating α_i, i ∈ G and appb;
            G = G ∪ {s'};
        }
    }
    Compute τ_i, i ∈ G solving (3) and T_G, C_G using (4), (5);
    Compute τ_i α_i, i ∈ G using (17);
    Compute rb = ([UA]'_ub − [UA]_lb)/[UA]_lb;
    using (15), (16), (1), (2);
}
```

4 Implementation details

The state space exploration algorithms have been implemented and integrated in METFAC [2]. The tool includes a model specification language based on production rules. Each production rule includes an action and may include several responses. Actions have associated rates and responses have associated probabilities. Both have associated guard expressions describing the conditions on the state under which the action/response is active. The interface for model generation includes several functions which are automatically obtained from the formal model specification. One of them, called `actresp` gives a list of the action/response pairs which are active in a given state. Another called `successor` gives the description (in terms of state variables) of the state reached from another state following a certain action/response pair. Other functions, called `rate` and `prob` give respectively the value of the rate of an action and the probability of a response in a particular state. Another function called `ar_active` tells whether an action/response pair is active in a particular state. That function is used to reduce the computational complexity associated to the maintenance of the set of transitions T among the generated states and the transition rates $\lambda_{i,U_{k,d}}$. The problem is that each time a new state s' is added to G, it is necessary to check whether the already generated states have transitions to s', add those transitions and subtract the rates from the corresponding $\lambda_{i,U_{k,d}}$. To make that efficiently, we have implemented hashing procedures based on the sets of components failed in the states to efficiently select partially expanded states which may have a transition to s'. For each failure and repair event of the model we compute the set of failed components corresponding to the states which can be predecessors of s' through a transition with that failure/repair event. Then, using hashing procedures, we determine the candidate states associated to the event, for each candidate, using `ar_active`, we determine which of the action/response pairs associated to the event are active in the candidate, and for each active pair we find, using `successor`, the description of the state reached from the candidate predecessor through the action/response pair; if the description matches the description of s', the transition is added to T and its rate subtracted from the corresponding $\lambda_{i,U_{k,d}}$ rate. The preprocessor of METFAC was specifically modified so that it included the function `ar_active`, which was not generated in the former version. The use of `ar_active` instead of `actresp` provides a 3-fold reduction in the CPU times and makes small the overhead associated to the maintenance of the set T.

The implementation of the bounding method requires the computation of the failure distance from the states in the frontier of U. In addition, we also have to compute the failure distances from the successors in U of the states in the frontier of G (required to obtain the transition rates $\lambda_{i,U_{k,d}}$). A trivial computation of these failure distances based on (9) can be time consuming if the number of minimal cuts is large. Most of the frontier transitions will be typically of the failure type. To compute more efficiently the failure distances associated to these failure transitions we have introduced the concept of *after minimal cut*. The after minimal cut associated to a minimal cut m and a failure event $e \in E$ is $m' = m - e$. Let AMC_e be the set of after minimal cuts associated to failure event e, i.e., $AMC_e = \{m' | m' = m - e, m \in MC, m \cap e \neq \phi\}$. Then, the failure distance from any state reached from s through a failure transition with failure event e, $ad(s,e)$, can be obtained

as:

$$ad(s,e) = \min\{d(s), \min_{m \in AMC_e} |m - F(s)|\}. \quad (19)$$

Using (19) instead of (9), the number of distances to minimal cuts $|m - F(s)|$ which have to be computed to determine $ad(s,e)$, $e \in E$, assuming $d(s)$ is known, is reduced from $|E||MC|$ to the typically much smaller $\sum_{e \in E} |AMC_e|$. Further reduction in the number of minimal cut touches and the associated overhead can be obtained as follows.

Assume that an upper bound ub for $d(s)$ is known (for instance, $ub = L = d(o)$). Since at most $|F(s)|$ components can be failed in any minimal cut we only need to consider the minimal cuts m with $|m| - |F(s)| < ub$. Assume also that we can access the minimal cuts indexed by order and selectors (bags included in the minimal cut) of order $\leq R$. For $|m - F(s)| < ub$, m must contain a selector p with all components failed and $|m| - |p| < ub$, i.e., $|p| \geq |m| - ub + 1$. Thus, for each possible minimal cut order c we can restrict our attention to the minimal cuts of order c containing selectors p with all components failed and $|p| = \min\{R, c - ub + 1\} = r$. Possible selectors can be examined by generating all bags of order r included in $F(s)$. Actual selectors can be identified easily if all selectors are kept in a hash table. The discussion justifies the following algorithm for the computation of $d(s)$:

Algorithm to compute $d(s)$

```
d(s) = L;
for (increasing minimal cut order c
     while c < d(s) + |F(s)|){
    r = min{R, c - d(s) + 1};
    Let P be the set of bags of order r included in F(s);
    for (each p ∈ P){
        for (each minimal cut m with |m| = c and p ⊂ m){
            d(s) = min{d(s), |m - F(s)|};
        }
    }
}
```

A similar scheme can be used to compute $ad(s,e)$, $e \in E$, assuming knowledge of $d(s)$. To reduce the overhead associated to the control of the algorithm we use one bound and index the selectors for all the failure events of the model together. The bound is initialized using the after failure distances from the state o with all components unfailed. The algorithm is:

Algorithm to compute $ad(s,e)$, $e \in E$

```
for (each e ∈ E) ad(s,e) = min{d(s), ad(o,e)};
adub = max_{e∈E}{ad(s,e)};
for (increasing after minimal cut order c
     while c < adub + |F(s)|){
    r = min{R, c - adub + 1};
    Let P be the set of bags of order r included in F(s);
    for (each p ∈ P){
        for (each after minimal cut m' with |m'| = c
             and p ⊂ m'){
            Let e be the failure event associated to m';
            ad(s,e) = min{ad(s,e), |m' - F(s)|};
        }
    }
}
```

These algorithms have been implemented and integrated with the state space exploration algorithms. They are used as follows. $d(o)$ and $ad(o,e)$, $e \in E$ are computed using (9) and (19). The failure distances from the generated states are kept in the state descriptions. When a state is expanded, failure distances from the new states reached through failure transitions are computed using the algorithm for $ad(s,e)$, $e \in E$; the failure distances from the new states reached through repair transitions are computed using the algorithm for $d(s)$. Since typically most of the new states are reached through failure transitions, the algorithm for $ad(s,e)$, $e \in E$ is invoked much more often than the other.

5 Analysis and comparison

In this section we analyze the performance of the bounding methods with state space exploration proposed here and compare them with the bounding method with state space exploration proposed in [21] and the bounding method without state space exploration described in [5]. We will also examine the performance of the state space exploration algorithm described in [21] combined with our bounding method [5]. The results were obtained in a SPARC10 workstation. The linear systems were solved by Gauss-Seidel, using the state cutting technique described in [3] for the linear systems (3), taking o as the cut-off state. The technique improved significantly the convergence of the iterative method.

The analysis and comparison will be made using two examples. The examples correspond to fault-tolerant systems having several tens of components and yield state spaces with of the order of 10^{10} states. We start considering the large example of [18], a distributed fault-tolerant database system, whose block diagram is given in Figure 3. The system includes two processor types (A and B), two sets of dual-ported controllers with two controllers per set and six disk clusters with four disks. Each set of controllers controls three clusters. Each processor type has three spares. The system is operational if at least one processor of any type is unfailed, at least one controller in each set is unfailed and at least three disks in each cluster are unfailed. Thus, the redundancy level is $L = 2$. A failure in the active processor A is propagated to the active processor B with probability 0.10. Processors and controllers fail with rate 1/2000, disks fail with different rates from one cluster to another. These rates are 1/6000, 1/8000, 1/10000, 1/12000, 1/14000, and 1/16000. Any component is failed in one of two modes with equal probabilities. The repair rate is 1 for one mode and 0.5 for the other. Components are repaired by a single repairman who chooses components at random from the bag of failed components. Unfailed components continue to fail when the system is down. The second example is a modified version of the first in which the number of controllers in each set is increased to 3 and the disks in each cluster to 5, without modifying any other aspect. The redundancy level of the second example is $L = 3$. The unavailability of the first example is 3.319×10^{-6}; the unavailability of the second one is 4.727×10^{-9}.

Figures 4 and 5 show the behavior of the bounding method with state space exploration described in [21] and our failure distance based bounding method with the state space exploration algorithm proposed in [21] (called MT), CONT_S and CONT_TG. The relative band $([UA]'_{ub} - [UA]_{lb})/[UA]_{lb}$ is given as a function of the number of states in G. The results show that the failure distance based bounding method with the state space exploration algorithms developed here significantly outperforms the method proposed in [21], specially for the second example. Thus, the number of states required to achieve a relative band of 10^{-3} for the example with $L = 2$ with the failure distance based bounding method under either CONT_S or CONT_TG is about 3 times smaller than the number of states required with the method with state space exploration proposed in [21].

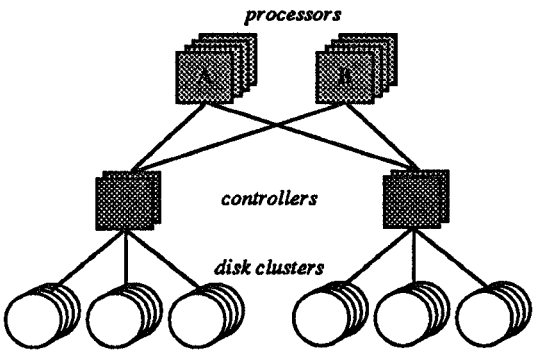

Figure 3: Block diagram of the first example.

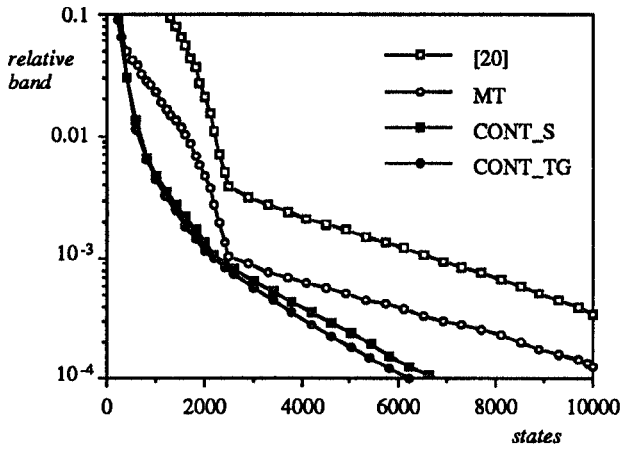

Figure 4: Behavior of the bounding method with state exploration proposed in [21] and the failure distance based bounding method with the state space exploration algorithms MT, CONT_S and CONT_TG for the example with redundancy level $L = 2$.

The difference in performance for the example with $L = 3$ is significantly greater. The state space exploration algorithm MT is significantly less efficient than CONT_S and CONT_TG for the failure distance based bounding method. We also experimented with the state space exploration algorithms CONT_S and CONT_TG combined with the bounding method proposed in [21] and found that their performance was very similar to that of MT. Thus, the proposal made in [21] is a good one for the basic bounding method used there. Finally, CONT_TG outperforms CONT_S, but only slightly. This was a little bit surprising, since we expected a more pronounced difference between both algorithms. Since, CONT_TG has a more complex implementation than CONT_S this opens the issue of whether CONT_TG really outperforms CONT_S in terms of the memory and CPU time required to achieve a given relative band. This issue will be explored next.

Tables 1 and 2 compare the failure distance based bounding method without state space exploration with the method with the state space exploration algorithms MT, CONT_S and CONT_TG. In the method without state space exploration G includes all states with up to K failed components. The values given in Table 1 correspond to the choices

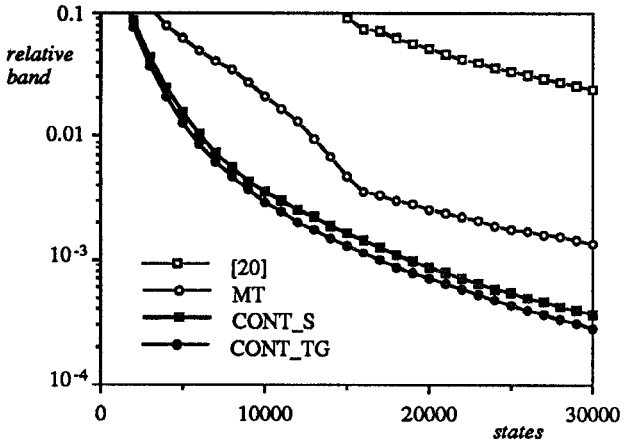

Figure 5: Behavior of the bounding method with state exploration proposed in [21] and the failure distance based bounding method with the state space exploration algorithms MT, CONT_S and CONT_TG for the example with redundancy level $L = 3$.

$K = 2, 3, 4$; the values given in Table 2 correspond to the choices $K = 3, 4, 5$. Except for the case $K = 2$ of Table 1, the methods with state space exploration with the algorithms CONT_S and CONT_TG outperform the method without state space exploration. The difference in the case in which state space exploration gives higher number of states is small and can be explained by the low accuracy of the approximated band contribution estimates which are available when G contains a small number of states. The improvement given by state space exploration is certainly moderate. However, we should note that state space exploration allows to adjust precisely the size of G (and thus the memory requirements and CPU time) to the desired accuracy for the bounds. The state space exploration algorithm MT is slightly worse than the method without state space exploration.

We also analyzed the qualitative characteristics of the state space exploration algorithms CONT_S and CONT_TG. Figure 6 gives the distribution of the number of states in G according to the values of the parameters k, d as the expansion progresses for the algorithm CONT_TG and the example with $L = 2$. The states are included in G following increasing values of k and increasing values of d. First, the few states with $k < 3$ are included. The next included states are those with $k = 3$ and $d = 0$, followed by those with $k = 3$ and $d = 1$, followed by the few states with $k = 3$ and $d = 2$. Then, states with $k = 4$ are included, most of them having $d = 0$. Only a few states with $k = 5$ are included at the end. The actual behavior matches very well the intuition about what should be a good behavior, thus supporting qualitatively the proposed state space exploration algorithms.

We noted in the previous section that the basic state space exploration algorithms CONT_S and CONT_TG are expensive in terms of CPU time for large $|G|$ because of the large number of linear systems (3) which have to be solved. Waved versions of those algorithms were proposed to overcome the problem. It remains however to see if waving deteriorates significantly the state space exploration and how the control parameter BR affects the efficiency of the algorithm. These issues are explored in Tables 3 and 4 which give the number of states and CPU times required to achieve several relative bands for the basic and waved

Table 1: Number of states required to achieve a given relative unavailability band for the example with redundancy level $L = 2$ without state exploration and with the state space exploration algorithms MT, CONT_S and CONT_TG.

rel. band	wo/ exp.	MT	CONT_S	CONT_TG
0.0733	231	302	280	250
2.16×10^{-3}	1,763	2,289	1,613	1,526
4.96×10^{-5}	10,464	12,705	8,523	8,028

Table 2: Number of states required to achieve a given relative unavailability band for the example with redundancy level $L = 3$ without state exploration and with the state space exploration algorithms MT, CONT_S and CONT_TG.

rel. band	wo/ exp.	MT	CONT_S	CONT_TG
0.169	1,771	2,429	1,267	1,199
6.15×10^{-3}	10,616	14,309	7,617	6,979
1.76×10^{-4}	52,916	$> 60,000$	40,100	36,207

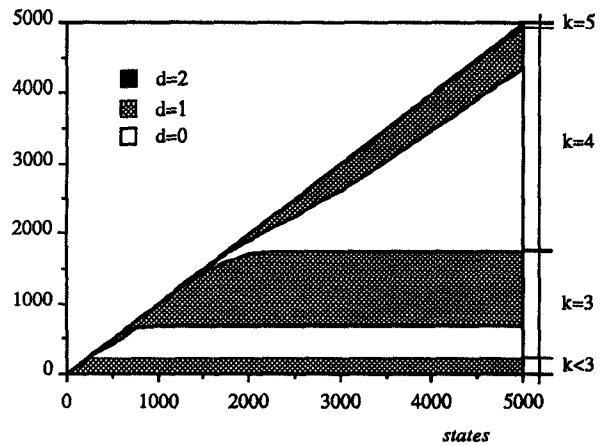

Figure 6: Evolution of the distribution of states according to the parameters k and d in the state space exploration algorithm CONT_TG for the example with redundancy level $L = 2$.

algorithms. We can note that the waved algorithms have significantly smaller CPU times with almost identical state space exploration quality. The differences in CPU times and required states are small for the values of BR explored. It seems that BR $= 0.1$ would be a sensible choice in general. Regarding the relative performance of CONT_S_W and CONT_TG_W we can note that the latter has smaller CPU times. CONT_TG_W also gives smaller $|G|$, but has some memory overhead over CONT_S_W because of the largest number of contributions per state in the frontier of G which have to be kept during the expansion process. We analyzed that overhead and found that it was significantly smaller than the memory savings resulting from a smaller G. Thus, in conclusion, the algorithm CONT_TG_W with BR $= 0.1$ seems to be a good choice.

Compared with the failure distance based bounding method without state space exploration [5], CONT_TG_W is about 2 times slower for the same number of generated states. This overhead can be regarded as reasonable taking into account the advantages of state space exploration. Regarding the efficiency of the algorithms for failure distances computation, for the example with $L = 2$, algorithm CONT_TG with BR $= 0.1$, and target relative band $= 10^{-4}$ (6,249 states), the total number of minimal cut touches was only 5,786 for $R = 2$, i.e., less than one touch per state. A trivial computation of the failure distances based on (9) would involve about 100 touches per state (the structure function has 9 minimal cuts, the model has 11 failure events and 10 repair events, but only a few repair events have associated transitions when few components are failed). The algorithms consumed a 0.9% of the total CPU time and the part which depends on the number of minimal cuts only a 0.1%. Thus, we feel that models with many more minimal cuts than the 9 of the example can be managed with small overhead.

6 Conclusions

State space exploration is attractive because: 1) it allows a precise adjustment of the size of the generated state space to the accuracy requirements for the bounds, and 2) it can, potentially, reduce the size of the required state space. Several heuristics can be used to guide the state space exploration. In this paper we have proposed and analyzed the

Table 3: Number of states (top) and CPU times in secs. (bottom) to achieve a given relative band for the steady-state unavailability of the example with redundancy level $L = 2$ under several state space exploration algorithms.

relative band	0.05	0.01	5×10^{-3}	10^{-3}	5×10^{-4}	10^{-4}
CONT_S	333	704	982	2,312	3,532	6,746
	0.91	2.40	3.98	23.9	55.1	238
CONT_S_W, BR $= 0.5$	333	704	982	2,312	3,532	6,746
	0.74	1.66	2.55	3.95	14.3	37.6
CONT_S_W, BR $= 0.1$	333	704	982	2,312	3,532	6,746
	0.75	1.61	2.41	7.32	13.9	36.2
CONT_S_W, BR $= 0$	345	722	969	2,315	3,534	6,746
	0.78	1.69	2.40	7.45	14.2	35.3
CONT_TG	324	651	944	2,216	3,237	6,253
	0.76	2.17	4.08	26.4	55.5	236
CONT_TG_W, BR $= 0.5$	324	651	944	2,216	3,237	6,253
	0.76	1.49	2.31	8.18	12.4	32.7
CONT_TG_W, BR $= 0.1$	324	651	944	2,216	3,237	6,249
	0.75	1.47	2.28	6.85	11.9	31.9
CONT_TG_W, BR $= 0$	324	667	947	2,220	3,240	6,249
	0.76	1.49	2.25	6.73	12.5	30.6

Table 4: Number of states (top) and CPU times in secs. (bottom) to achieve a given relative band for the steady-state unavailability of the example with redundancy level $L = 3$ under several state space exploration algorithms.

relative band	0.1	0.05	0.01	5×10^{-3}	10^{-3}	5×10^{-4}
CONT_S	1,894	2,798	6,141	8,415	18,739	25,915
	14.9	32.8	186	381	2,436	4,989
CONT_S_W, BR $= 0.5$	1,894	2,798	6,141	8,415	18,739	25,923
	6.07	10.4	32.6	56.1	212	389
CONT_S_W, BR $= 0.1$	1,894	2,798	6,150	8,415	18,695	25,862
	5.93	9.72	31.4	55.5	207	376
CONT_S_W, BR $= 0$	2,607	3,073	6,138	8,326	18,716	25,897
	9.10	11.3	31.6	52.2	203	381
CONT_TG	1,736	2,558	5,617	7,698	16,873	23,484
	17.5	36.2	195	407	2,330	4,876
CONT_TG_W, BR $= 0.5$	1,736	2,558	5,617	7,698	16,873	23,466
	5.32	8.87	29.3	46.8	174	320
CONT_TG_W, BR $= 0.1$	1,736	2,558	5,617	7,698	16,879	23,507
	5.12	8.69	28.5	46.8	174	308
CONT_TG_W, BR $= 0$	2,509	2,930	5,657	7,730	16,915	23,511
	8.70	10.3	27.9	44.4	168	306

performance of several state space exploration algorithms targeted to a failure distance based steady-state availability bounding method recently developed. The algorithms are different from previous proposals in that they are focused directly to the reduction of the band with a minimum number of expansions. In order to reduce the CPU time requirements we have introduce the concept of waved expansion and have shown its effectiveness. From the algorithms we have considered, the waved transition group expansion algorithm CONT_TG_W has consistently shown a better performance both in memory and CPU time requirements. The failure distance based bounding method combined with that state space exploration algorithm significantly outperforms previous methods.

References

[1] R.E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing. Probability Models*, McArdle Press, Silver Spring, 1981.

[2] J. A. Carrasco and J. Figueras, "METFAC: Design and Implementation of a Software Tool for Modeling and Evaluation of Complex Fault-Tolerant Computing Systems," in *Proc. 16th Int. Symp. on Fault-Tolerant Computing FTCS-16*, 1986, pp. 424–429.

[3] J. A. Carrasco, "Analysis of Sparse Numerical Methods for Dependability Evaluation," in *Proc. 16th IAESTED Int. Conf. on Identification, Modeling and Simulation*, 1987, pp. 437–441.

[4] J. A. Carrasco, "Failure distance-based Simulation of Repairable Fault-Tolerant Systems," in *Computer Performance Evaluation*, Elsevier, 1992, pp. 351–366.

[5] J. A. Carrasco, "Improving Availability Bounds using the Failure distance Concept", in *Dependable Computing and Fault-Tolerant Systems, vol. 9*, Springer-Verlag, 1995, pp. 479–497.

[6] E. Çinlar, *Introduction to Stochastic Processes*, Prentice-Hall, Inc., New Jersey, 1975.

[7] P.J. Courtois and P. Semal, "Bounds for the positive eigenvectors of nonnegative matrices and for their approximations," *Journal of the ACM*, vol. 31, no. 4, pp. 804–825, October 1984.

[8] P.J. Courtois and P. Semal, "Computable bounds for conditional steady-state probabilities in large Markov chains and queuing models," *IEEE J. of Selected Areas in Communications*, vol. SAC-4, no. 6, September 1986, pp. 926–937.

[9] S.-N. Chiou and V.O.K. Li, "Reliability Analysis of a Communication Network with Multimode Components." *IEEE J. of Selected Areas in Communications*, vol. SAC-4, no. 7, October 1986, pp. 1156–1161.

[10] M. Dal Cin, "Availability Analysis of a Fault-Tolerant Computer System," *IEEE Trans. on Reliability*, vol. R-29, no. 3, August 1980, pp. 265–268.

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

[12] 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, January 1992, pp. 36–51.

[13] Y.F. Lam and V.O.K. Li, "An Improved Algorithm for Performance Analysis of Networks with Unreliable Components," *IEEE Trans. on Communications*, vol. COM-34, no. 5, May 1986, pp. 496–497.

[14] V.O.K. Li and J.A. Silvester, "Performance Analysis of Networks with Unreliable Components," *IEEE Trans. on Communications*, vol. COM-32, no. 10, October 1984, pp. 1105–1110.

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

[16] J.C.S. Lui and 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.

[17] N.F. Maxemchuk and K. Sabnani, "Probabilistic Verification of Communication Protocols," in *Proc. of the IFIF 7th Int. Conf. on Protocol Specification, Testing, and Verification*, North-Holland, 1987, pp. 307–320.

[18] 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, pp. 1714–1723, December 1989.

[19] V.F. Nicola, M. Nakayama, P. Heidelberger and A. Goyal, "Fast simulation of dependability models with general failure, repair and maintenance processes," in *Proc. 20th IEEE Int. Symp. on Fault-Tolerant Computing FTCS-20*, 1990, pp. 491–498.

[20] V.F. Nicola, P. Shahabuddin, P. Heidelberger, and P.W. Glynn, "Fast simulation of steady-state availability in non-markovian highly dependable systems," in *Proc. 23th IEEE Int. Symp. on Fault-Tolerant Computing FTCS-23*, 1993, pp. 38–47.

[21] 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.

[22] Ch.-L. Yang and P. Kubat, "Efficient Computation of Most Probable States for Communication Networks with Multimode Components," *IEEE Trans. on Communications*, vol. 37, no. 5, May 1989, pp. 535–538.