Failure Distance Based Bounds for Steady-state Availability without the Knowledge of Minimal Cuts

Víctor Suñé and Juan A. Carrasco
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
Diagonal 647, plta. 9, 08028 Barcelona, Spain
{sunye, carrasco}@eel.upc.es

Abstract

We propose an algorithm to compute bounds for the steady-state unavailability using continuous-time Markov chains, which is based on the failure distance concept. The algorithm generates incrementally a subset of the state space until the tightness of the bounds is the specified one. In contrast with a previous algorithm also based on the failure distance concept, the proposed algorithm uses lower bounds for failure distances which are computed on the fault tree of the system, and does not require the knowledge of the minimal cuts. This is advantageous when the number of minimal cuts is large or their computation is time-consuming.

1. Introduction

Continuous-time Markov chain models (CTMC) are a flexible, powerful tool for computing steady-state dependability measures for fault tolerant systems such as the steady-state availability, A. However, the steady-state probability distribution of the CTMC modeling realistic systems, and, thus, A, cannot be computed exactly in many cases because of the enormous size of the state space of the CTMC. Bounding techniques are an attractive approach. Using those techniques, only a subset G of the state space of the CTMC is generated and the behavior of the system outside G is bounded somehow. Bounding techniques have been developed in the last few years and currently there exist several bounding methods [2, 3, 4, 12, 13, 14, 15, 19]. In the first of such methods [15], bounds for the steady-state unavailability *UA* $= 1 - A$ are obtained by partitioning the non-generated portion U according to the number of failed components and bounding the behavior of the chain in U using upper bounds for the failure transition rates and lower bounds for the repair transition rates. The method is, however, computationally very costly because a linear system of size $\approx |G|$ has to be solved for each return state, i.e. each state through which G can be entered from U. In the same paper, a state cloning technique is proposed which reduces the number of linear systems which have to be solved but introduces some looseness in the bounds. In [12] a refinement of the method is proposed for the particular case in which all states but the one without failed components are cloned. The technique avoids a complete reapplication of the algorithm each time G is enlarged in the search for the desired accuracy but looses up further the bounds. This additional looseness has been reduced in another paper from the same authors [13]. In the method proposed in [4], the bounds of [15] are computed without cloning states solving only four linear systems of size $|G|$. In [19] another bounding method is developed in which the bounds are iteratively refined using detailed knowledge about the model in U in the proximities of G. In [2] a bounding method based on the failure distance concept is proposed which gives bounds for *UA* which are never worse, and typically better, than those given by [15]. The method uses the cloning technique of [15] but adapts one of the algorithms developed in [4] so that only five linear systems of size $|G|$ have to be solved to compute the proposed bounds.

The previous methods assume that the state space of the CTMC is finite and that there is a transition to the left in all non-generated states of the CTMC. Both restrictions have been removed in the generalization of [15] proposed in [14]. Another generalization of [15] for finite CTMCs has been recently proposed in [3]. In that method, group repair and phase type repair distributions are allowed.

In the methods reviewed so far G includes all states of the CTMC having up to K failed components. The issue of how to generate G so that it includes as few states as possible to achieve the required accuracy has also been investigated. In [9], state space exploration techniques have been developed for the bounding method proposed in [15] with the cloning technique. However, these state space exploration techniques are expensive since they require the solution of a linear system of size $|G|$ after the expansion of every state. More efficient state space exploration techniques based on the concept of wave expansion and specifically targeted to the method developed in [2] have been proposed in [5].

The bounding method proposed in [2] requires the knowledge of the set of minimal cuts of the system, *MC*. There exist a number of algorithms to obtain *MC* [6, 8, 11, 17]. Computation of *MC* is, however, NP-hard [18], so those algorithms may break down. In addition, *MC* can be very large, thus causing a large memory overhead due to the need of holding *MC*. In this paper we develop a new bounding method which uses lower bounds for failure distances which are computed on the fault tree of the system, and thus does not require the knowledge of *MC*. The method is useful as an alternative to the method proposed in [2] when the algorithms to obtain *MC* break down or the number of minimal cuts is large. The rest of the paper is organized as follows. Section 2 defines the modeling framework and gives necessary background. Section 3 obtains the bounds for *UA* using lower bounds for failure distances. Section 4 describes the algorithm to compute lower bounds for failure distances on the fault tree. Section 5 analyzes the proposed bounding method and compares it with the bounding method proposed in [2] and the bounding method proposed in [15] with state space exploration. Finally, Section 6 includes the conclusions.

2. Preliminaries

We consider fault-tolerant systems made up of components which fail and are repaired. The operational/down state of the system is determined by the unfailed/failed state of its components by means of a coherent [1] structure function represented by a coherent fault tree. Components are grouped into types, being indistinguishable the components of the same type. Therefore, collections of components can be dealt with as bags [16]. Any bag of component types which can fail simultaneously will be called a *failure bag*. We assume known the set of failure bags of the system, E, and, for each $e \in E$, an upper bound, $\lambda_{\rm ub}(e)$, for the rate of any transition associated with e. Repair actions involve just one component and we assume also known a lower bound, $g(k) > 0$, $k > 0$, for the rate of any transition associated with a repair action in a state with k failed components.

Let $X = \{X(t);\, t \geq 0\}$ be the finite CTMC modeling the system and let Ω be its state space. We assume that there is only one state in Ω, which will be referred to as o, without failed components and that there is at least one repair action in any state in $\Omega - \{o\}$. Then, X will be irreducible and, thereby, ergodic.

Since the steady-state availability is typically very close to one, it is often preferable to compute the steady-state

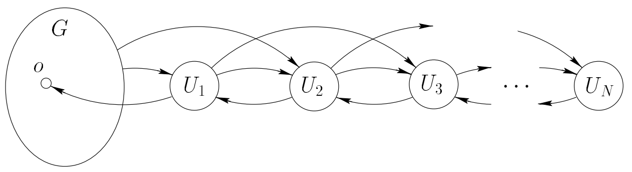


Figure 1. State transition diagram of the modified CTMC X.

unavailability, *UA*. Let D be the subset of down states of X and let $\mathbf{p} = (p_i)_{i\in\Omega}$ be the steady-state probability vector of X. We have

$$\mathit{UA} = \sum_{i \in D} p_i\,.$$

Bounds for *UA* will be computed using detailed knowledge of X in the generated subset, G, and bounding the behavior of X in $U = \Omega - G$. It will be used the state cloning technique proposed in [15]. The technique consists in modifying X by adding to U clones of the states in G with more than F failed components, accounting for the visits to the corresponding states of G after X exits G and before the number of failed components has fallen below $F + 1$. We will use the state cloning technique with $F = 0$, i.e. clones of all states $s \in G - \{o\}$ will be added to U. The selection of $F = 0$ is made to ease the generation of G. With $F = 0$, G includes states which are reachable through G from state o and generation of G from a high-level modeling formalism. With $F > 0$, G may contain states which are reachable from o through U and generation of G requires a priori knowledge about the set of states of X. The modified X has the structure depicted in Figure 1, where U_k includes all states in U with exactly k failed components and N is the number of components of the system. In the following, X will denote the modified X.

Throughout the paper we will denote by $\lambda_{s,s'}$, $s, s' \in \Omega$, the transition rate from state s to state s', by $\lambda_s = \sum_{\substack{s'\in\Omega \\ s\neq s'}} \lambda_{s,s'}$, $s \in \Omega$, the output rate of s, and by $\lambda_{s,C} = \sum_{s'\in C} \lambda_{s,s'}$, $s \in \Omega$, $C \subset \Omega$, the transition rate from s to the subset of states C, all referred to X unless otherwise stated. We will also consider several transient CTMC Y. Each such Y has state space $B \cup \{a\}$, where all states in B are transient and a is an absorbing state, and has a well-defined initial probability distribution with $P[Y(0) \in B] = 1$. $\tau(s,Y)$, $s \in B$, will denote the mean time spent by Y in s before absorption, and $\tau(C,Y) = \sum_{s\in C} \tau(s,Y)$, $C \subset B$, will denote the mean time to absorption in subset C. It is well-known that the mean time to absorption vector $\boldsymbol{\tau} = (\tau(s,Y))_{s\in B}$ is the solution of the linear system $\mathbf{A}\boldsymbol{\tau} = -\mathbf{q}$, where $\mathbf{A}$ is the restriction to B of the infinitesimal generator of Y, and $\mathbf{q} = (P[Y(0) = s])_{s\in B}$. It is also known that

$\tau(s,Y)\lambda_{s,s'}$ is the expected number of times that a transition from s to s', $s \in B$, $s' \in B \cup \{a\}$, is followed.

3. Bounds for the Steady-state Unavailability

Consider the regenerative behavior of X, taking as regeneration points the times at which X enters o from U. Let T_G and T_U be the contributions of G and U to the mean time between regenerations of X, and let C_G and C_U be the respective contributions to the mean down time. From regeneration process theory (see, for instance [7]), we have

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

Assume that upper bounds $[T_U]_{\mathrm{ub}}$ and $[C_U]_{\mathrm{ub}}$ for, respectively, T_U and C_U are known. Then [2, Theorem 2]

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

$$[\mathit{UA}]_{\mathrm{ub}} = \frac{C_G + [C_U]_{\mathrm{ub}}}{T_G + [C_U]_{\mathrm{ub}}}, \tag{2}$$

are, respectively, a lower and an upper bound for *UA*.

Let Y_G be the transient CTMC with state space $G \cup \{a\}$ and initial state o built from X by directing to a the transitions from states in G to states in U. T_G and C_G can be expressed in terms of the mean time to absorption vector of Y_G, $(\tau(s,Y_G))_{s\in G}$, as

$$T_G = \sum_{s\in G} \tau(s,Y_G), \tag{3}$$

$$C_G = \sum_{s\in G\cap D} \tau(s,Y_G). \tag{4}$$

3.1. Upper bound $[T_U]_{\mathrm{ub}}$

The upper bound $[T_U]_{\mathrm{ub}}$ is the same as that of [2, 15]. Let FC be the set of different cardinalities of the failure bags of the model, let E_i be the subset of E including all failure bags of cardinality i and let $f_i = \sum_{e\in E_i} \lambda_{\mathrm{ub}}(e)$. Consider the transient CTMC Y^{u_k} with state space $\bigcup_{k=1}^{N}\{u_k\} \cup \{a\}$, initial state u_k and the state transition diagram shown in Figure 2. For each state u_k and each $i \in FC$, $k + i \le N$, there is a transition to u_{k+i} with rate f_i, and a transition to u_{k-1} if $k > 1$ and a otherwise with rate $g(k)$. Let $T(k)$ be the mean time to absorption of Y^{u_k} and let

$$\pi_k = \sum_{s\in G} \tau(s,Y_G)\lambda_{s,U_k} \tag{5}$$

be the probability that X enters U through U_k. Then [2, Theorem 4]

$$[T_U]_{\mathrm{ub}} = \sum_{k=1}^{N} \pi_k T(k) \tag{6}$$

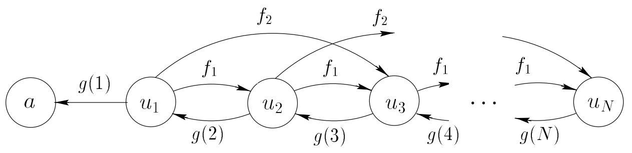


Figure 2. State transition diagram of the transient CTMC Y^{u_k}.

upper bounds T_U. An efficient method to compute $T(k)$, $1 \le k \le N$, is described in [2].

3.2. Upper Bound $[C_U]_{\mathrm{ub}}$

The upper bound $[C_U]_{\mathrm{ub}}$ is based on lower bounds for failure distances. The failure distance from a state $s \in \Omega$, $d(s)$, is defined [2] as the minimum number of components which have to fail in addition to those already failed in s to take the system down. Let $F(s)$ be the bag of failed components in $s \in \Omega$. Assume that a lower bound for $d(s)$, $\tilde{d}(s)$, is available satisfying:

A1. $0 \le \tilde{d}(s) \le d(s)$,

A2. $\tilde{d}(s) = 0$ if and only if $s \in D$, and

A3. $\tilde{d}(s) - |F(s') - F(s)| \le \tilde{d}(s') \le \tilde{d}(s)$, $F(s) \subset F(s')$.

Note that assumption A3 implies that given a transition from s to s', $s, s' \in \Omega$, associated with a failure bag $e \in E$, $\tilde{d}(s) - |e| \le \tilde{d}(s') \le \tilde{d}(s)$.

Let $\tilde{U}_{k,d}$ be the subset of U including all states s with k failed components and $\tilde{d}(s) = d$, and let $\tilde{L} = \tilde{d}(o)$. The set $\tilde{\mathcal{R}}$ of (k,d) pairs for which $\tilde{U}_{k,d}$ might be $\neq \emptyset$ is given by the constraints

$$1 \le k \le N,$$
$$\max\{0, \tilde{L} - k\} \le d \le \min\{\tilde{L}, N - k\}.$$

The constraints on k are obvious. The constraints $\tilde{L} - k \le d$ and $d \le \tilde{L}$ follow from assumption A3 and the definition of $\tilde{L}$; $0 \le d$ follows from assumption A1. Finally, $d \le N - k$ follows from assumption A3 and the fact that the structure function of the system is coherent by taking s' the state with all components failed and noting that $\tilde{d}(s') \le d(s') = 0$ and, therefore, $\tilde{d}(s') = 0$.

Let Y_U^s, $s \in U$ be the transient CTMC with state space $U \cup \{a\}$ and initial state s built from X by directing to a the transitions from states in U to o. Let C_U^s be the mean down time to absorption of Y_U^s. Recalling that

$\sum_{s' \in G} \tau(s', Y_G)\lambda_{s',s}$, $s \in U$, is the probability that X enters U through s, we have

$$C_U = \sum_{s' \in G} \sum_{s \in U} \tau(s', Y_G)\lambda_{s',s}\, C_U^s$$
$$= \sum_{s' \in G} \sum_{(k,d) \in \widetilde{\mathcal{R}}} \sum_{s \in \widetilde{U}_{k,d}} \tau(s', Y_G)\lambda_{s',s}\, C_U^s\,. \qquad (7)$$

Let $\widetilde{C}(k,d)$ be upper bounds for C_U^s, $s \in \widetilde{U}_{k,d}$, and

$$\widetilde{\pi}_{k,d} = \sum_{s \in G} \tau(s, Y_G)\lambda_{s,\widetilde{U}_{k,d}}\,. \qquad (8)$$

Let

$$[C_U]_{\text{ub}} = \sum_{(k,d) \in \widetilde{\mathcal{R}}} \widetilde{\pi}_{k,d}\, \widetilde{C}(k,d)\,. \qquad (9)$$

We have

Theorem 1. *Assume* $C_U^s \leq \widetilde{C}(k,d)$, $s \in \widetilde{U}_{k,d}$, *Then,* $C_U \leq [C_U]_{\text{ub}}$.

Proof. Using (7), the fact that $C_U^s \leq \widetilde{C}(k,d)$, $s \in \widetilde{U}_{k,d}$, (8), and (9):

$$C_U = \sum_{s' \in G} \sum_{(k,d) \in \widetilde{\mathcal{R}}} \sum_{s \in \widetilde{U}_{k,d}} \tau(s', Y_G)\lambda_{s',s}\, C_U^s$$
$$\leq \sum_{s' \in G} \sum_{(k,d) \in \widetilde{\mathcal{R}}} \sum_{s \in \widetilde{U}_{k,d}} \tau(s', Y_G)\lambda_{s',s}\, \widetilde{C}(k,d)$$
$$= \sum_{s' \in G} \sum_{(k,d) \in \widetilde{\mathcal{R}}} \tau(s', Y_G)\lambda_{s',\widetilde{U}_{k,d}}\, \widetilde{C}(k,d)$$
$$= \sum_{(k,d) \in \widetilde{\mathcal{R}}} \sum_{s' \in G} \tau(s', Y_G)\lambda_{s',\widetilde{U}_{k,d}}\, \widetilde{C}(k,d)$$
$$= \sum_{(k,d) \in \widetilde{\mathcal{R}}} \widetilde{\pi}_{k,d}\, \widetilde{C}(k,d) = [C_U]_{\text{ub}}\,. \quad \square$$

Let L be the exact failure distance from state o, i.e. $L = d(o)$, and let

$$\widetilde{C}(k) = \sum_{i=\widetilde{L}}^{N} \tau(u_i, Y^{u_k})\,. \qquad (10)$$

We have

Theorem 2. $C_U^s \leq \widetilde{C}(k)$, $s \in U_k$.

Proof. By assumption A1, $\widetilde{L} \leq L$. Using that [2, Theorem 6] $C_U^s \leq \sum_{i=L}^{N} \tau(u_i, Y^{u_k})$ and (10):

$$C_U^s \leq \sum_{i=L}^{N} \tau(u_i, Y^{u_k}) \leq \sum_{i=\widetilde{L}}^{N} \tau(u_i, Y^{u_k}) = \widetilde{C}(k)\,. \quad \square$$

$\widetilde{C}(k)$, $1 \leq k \leq N$ can be computed efficiently using the method described in [2] for $C(k)$, $1 \leq k \leq N$, with L replaced by $\widetilde{L}$.

The bounds $\widetilde{C}(k,d)$ are computed using an iterative procedure which starts with $\widetilde{C}(k,d) = \widetilde{C}(k)$ and improves the bounds using potentially better bounds $\widetilde{C}'(k,d)$ until no significant improvement is achieved.

Let $s \in \widetilde{U}_{k,d}$ and consider a transition from s to $s' \in U$ associated with a failure bag $e \in E_i$, $i \in FC$. Clearly, $s' \in \widetilde{U}_{k+i,d'}$ for suitable d' values. Imposing $(k+i, d') \in \widetilde{\mathcal{R}}$, $i \leq N-k$ and $d' \leq \min\{\widetilde{L}, N-k-i\}$. Moreover, from assumptions A1 and A3, $\max\{0, d-i\} \leq d' \leq d$. Therefore, the only feasible destination subsets $\widetilde{U}_{k+i,d'}$, $i \in FC$, are those satisfying $i \leq N-k$ and (recall that $d \leq \widetilde{L}$) $\max\{0, d-i\} \leq d' \leq \min\{d, N-k-i\}$. Let $\widetilde{\mathcal{R}}' = \{(k,d,i,d') | (k,d) \in \widetilde{\mathcal{R}},\ i \in FC,\ \max\{0,d-i\} \leq d' \leq \min\{d, N-k-i\}\}$. Assume that upper bounds $\widetilde{F}(k,d,i,r)$, $(k,d,i,r) \in \widetilde{\mathcal{R}}'$, for $\sum_{d'=0}^{r} \lambda_{s,\widetilde{U}_{k+i,d'}}$, $s \in \widetilde{U}_{k,d}$, are available and let

$$\widetilde{f}_{i,j}(k,d) = \begin{cases} \widetilde{F}(k,d,i,d-j) \\ \quad -\widetilde{F}(k,d,i,d-j-1)\,, & \sigma \leq j < \omega \\ \widetilde{F}(k,d,i,d-j)\,, & j = \omega\,, \end{cases}$$

where $\sigma = \max\{0, k+d+i-N\}$ and $\omega = \min\{i, d\}$. The upper bounds $\widetilde{C}'(k,d)$ are computed using

$$\widetilde{C}'(k,d) = \frac{I_{d=0}}{g(k)} + I_{k>1}\Big[I_{d>\widetilde{L}-k}\, \widetilde{C}(k-1,d) + I_{d\leq\widetilde{L}-k}\, \widetilde{C}(k-1,d+1)\Big] + \frac{1}{g(k)} \sum_{\substack{i \in FC \\ i \leq N-k}} \sum_{j=\sigma}^{\omega} \widetilde{f}_{i,j}(k,d)\, \widetilde{C}(k+i, d-j)\,, \qquad (11)$$

where I_c is the indicator function returning 1 if c is true and 0 otherwise. The algorithm to compute the $\widetilde{C}(k,d)$ bounds is given in Figure 3. The parameter ϵ is a tolerance factor which determines when the improvement is small enough for the algorithm to stop.

Next, we prove that the $\widetilde{C}(k,d)$ computed by the algorithm of Figure 3 upper bound C_U^s, $s \in \widetilde{U}_{k,d}$, provided that $\widetilde{F}(k,d,i,r)$, $(k,d,i,r) \in \widetilde{\mathcal{R}}'$, and $\widetilde{F}(k,d,i,d)$, $(k,d,i,d) \in \widetilde{\mathcal{R}}'$, are decreasing on d. The proof will consist of a sequence of three propositions and a theorem.

Proposition 1. *Let* $(k,d) \in \widetilde{\mathcal{R}}$. *Assume that* $C_U^l \leq \widetilde{C}(k,d)$, $l \in \widetilde{U}_{k,d}$, *and that* $\widetilde{C}(k,d)$ *is decreasing on* d. *Then,* $C_U^l \leq \widetilde{C}'(k,d)$, $l \in \widetilde{U}_{k,d}$.

Proof. Let $l \in \widetilde{U}_{k,d}$. By assumption A2, $d = 0$ if and only if $l \in D$. Therefore, C_U^l is equal to the mean time in

```
for (all $(k,d) \in \widetilde{\mathcal{R}}$) $\widetilde{C}(k,d) = \widetilde{C}(k)$;
do {
  $\epsilon' = 0$;
  for ($k = 1;\ k \le N;\ k$++)
    for ($d = \max\{0, \widetilde{L} - k\};\ d \le \min\{\widetilde{L}, N-k\};$
        $d$++) {
      Compute $\widetilde{C}'(k,d)$ using (11);
      if ($\widetilde{C}'(k,d) < \widetilde{C}(k,d)$) {
        $\epsilon' = \max\{\epsilon', (\widetilde{C}(k,d) - \widetilde{C}'(k,d))/\widetilde{C}'(k,d)\}$;
        $\widetilde{C}(k,d) = \widetilde{C}'(k,d)$;
      }
    }
} while ($\epsilon' \ge \epsilon$);
```

Figure 3. Algorithm to compute the $\widetilde{C}(k,d)$ bounds.

l, if $d = 0$, plus the mean down time from the next state m, if $m \in U$. Let us discuss next to which subsets $\widetilde{U}_{k',d'}$ m may belong. By assumption A3, a transition associated with a repair action involving one component can only lead to $m \in \widetilde{U}_{k-1,d'}$, $k > 1$ (if $k = 1$, $m = o \notin U$), $d \le d' \le d+1$. $d' = d$ is possible only if $(k-1,d) \in \widetilde{\mathcal{R}}$, i.e. $d > \widetilde{L} - k$; similarly, $d' = d+1$ requires $d < \widetilde{L}$. Consider now transitions associated with failure bags $e \in E_i$, $i \in FC$. Clearly, $m \in \widetilde{U}_{k+i,d-j}$ for suitable j values. Imposing $(k+i, d-j) \in \widetilde{\mathcal{R}}$, $i \le N-k$ and $d - j \le \min\{\widetilde{L}, N-k-i\}$. Furthermore, from assumptions A1 and A3, $\max\{0, d-i\} \le d-j \le d$. Therefore, the only feasible $\widetilde{U}_{k+i,d-j}$ subsets are those satisfying (recall that $d \le \widetilde{L}$) $\max\{0, k+d+i-N\} \le j \le \min\{i,d\}$. Based on this discussion we can write

$$C_U^l = \frac{I_{d=0}}{\lambda_l} + I_{k>1}\Big[I_{d>\widetilde{L}-k} \sum_{m \in \widetilde{U}_{k-1,d}} \frac{\lambda_{l,m}}{\lambda_l} C_U^m + I_{d<\widetilde{L}} \sum_{m \in \widetilde{U}_{k-1,d+1}} \frac{\lambda_{l,m}}{\lambda_l} C_U^m\Big] + \sum_{\substack{i \in FC \\ i \le N-k}} \sum_{j=\max\{0,k+d+i-N\}}^{\min\{i,d\}} \sum_{m \in \widetilde{U}_{k+i,d-j}} \frac{\lambda_{l,m}}{\lambda_l} C_U^m .$$

Using that, by assumption, $C_U^m \le \widetilde{C}(k',d')$, $m \in \widetilde{U}_{k',d'}$, and introducing the notation $g_j(l) = \lambda_{l,\widetilde{U}_{k-1,d+j}}$, $f_{ij}(l) = \lambda_{l,\widetilde{U}_{k+i,d-j}}$, $J_m(i) = \max\{0, k+d+i-N\}$, and $J_M(i) = \min\{i,d\}$,

$$C_U^l \le T_1 + T_2 + \sum_{\substack{i \in FC \\ i \le N-k}} T_3(i),$$

with

$$T_1 = \frac{I_{d=0}}{\lambda_l}$$

$$T_2 = I_{k>1}\Big[I_{d>\widetilde{L}-k} \frac{g_0(l)}{\lambda_l} \widetilde{C}(k-1,d) + I_{d<\widetilde{L}} \frac{g_1(l)}{\lambda_l} \widetilde{C}(k-1,d+1)\Big] ,$$

$$T_3(i) = \sum_{j=J_m(i)}^{J_M(i)} \frac{f_{ij}(l)}{\lambda_l} \widetilde{C}(k+i, d-j) .$$

From this point, the proof continues exactly as in [2, Proposition 1] setting $F = 0$ and substituting L, $C(k,d)$, $F(k,d,i,r)$, and $f_{i,j}(k,d)$ by, respectively, $\widetilde{L}$, $\widetilde{C}(k,d)$, $\widetilde{F}(k,d,i,r)$, and $\widetilde{f}_{i,j}(k,d)$. □

Proposition 2. *Assume that* $\widetilde{C}(k,d)$, $(k,d) \in \widetilde{\mathcal{R}}$, $\widetilde{F}(k,d,i,r)$, $(k,d,i,r) \in \widetilde{\mathcal{R}}'$, *and* $\widetilde{F}(k,d,i,d)$, $(k,d,i,d) \in \widetilde{\mathcal{R}}'$, *are decreasing on* d. *Then*

$$\widetilde{A}(k,d,i) = \sum_{j=\max\{0,k+d+i-N\}}^{\min\{i,d\}} \widetilde{f}_{i,j}(k,d)\, \widetilde{C}(k+i, d-j) ,$$

$i \in FC$, $i \le N-k$, *is decreasing on* d.

Proof. The proof is exactly as in [2, Proposition 2] replacing $A(k,d,i)$, $\mathcal{R}$, $C(k,d)$, $F(k,d,i,r)$, $F(k,d,i,d)$, and $f_{i,j}(k,d)$ by, respectively, $\widetilde{A}(k,d,i)$, $\widetilde{\mathcal{R}}$, $\widetilde{C}(k,d)$, $\widetilde{F}(k,d,i,r)$, $\widetilde{F}(k,d,i,d)$ and $\widetilde{f}_{i,j}(k,d)$. □

Proposition 3. *Assume that* $\widetilde{C}(k,d)$, $(k,d) \in \widetilde{\mathcal{R}}$, $\widetilde{F}(k,d,i,r)$, $(k,d,i,r) \in \widetilde{\mathcal{R}}'$, *and* $\widetilde{F}(k,d,i,d)$, $(k,d,i,d) \in \widetilde{\mathcal{R}}'$, *are decreasing on* d. *Then* $\widetilde{C}'(k,d)$, $(k,d) \in \widetilde{\mathcal{R}}$, *is decreasing on* d.

Proof. Let (k,d), $(k,d+1) \in \widetilde{\mathcal{R}}$. Using (11):

$$\widetilde{C}'(k,d) - \widetilde{C}'(k,d+1) = T_1 + T_2 + \sum_{\substack{i \in FC \\ i \le N-k}} T_3(i) ,$$

with

$$T_1 = \frac{I_{d=0} - I_{d+1=0}}{g(k)} ,$$

$$T_2 = I_{k>1}\Big[I_{d>\widetilde{L}-k} \widetilde{C}(k-1,d) + I_{d\le\widetilde{L}-k} \widetilde{C}(k-1,d+1) - I_{d+1>\widetilde{L}-k} \widetilde{C}(k-1,d+1) - I_{d+1\le\widetilde{L}-k} \widetilde{C}(k-1,d+2)\Big] ,$$

$$T_3(i) = \frac{\widetilde{A}(k,d,i) - \widetilde{A}(k,d+1,i)}{g(k)} ,$$

where $\widetilde{A}(k, d, i)$ is as defined in Proposition 2. We will show that T_1, T_2 and $T_3(i)$ are all ≥ 0. Since $(k, d) \in \widetilde{\mathcal{R}}$, $d \geq 0$ and $d + 1 > 0$. Therefore, $T_1 = I_{d=0}/g(k) \geq 0$. Regarding T_2, three cases must be considered: a) $k = 1$, b) $k > 1$, $d > \widetilde{L} - k$, and c) $k > 1$, $d \leq \widetilde{L} - k$. In case a, $T_2 = 0$; in case b, $T_2 = \widetilde{C}(k-1, d) - \widetilde{C}(k-1, d+1) \geq 0$ because $\widetilde{C}(k', d')$, $(k', d') \in \widetilde{\mathcal{R}}$, is assumed decreasing on d; in case c, $d + 1 > \widetilde{L} - k$ because (k, d), $(k, d+1) \in \widetilde{\mathcal{R}}$, and, thereby, $T_2(i) = \widetilde{C}(k-1, d+1) - \widetilde{C}(k-1, d+1) = 0$. Finally, $T_3(i) \geq 0$ by Proposition 2. $\square$

Theorem 3. *Assume that $\widetilde{F}(k, d, i, r)$, $(k, d, i, r) \in \widetilde{\mathcal{R}}'$, and $\widetilde{F}(k, d, i, d)$, $(k, d, i, d) \in \widetilde{\mathcal{R}}'$, are decreasing on d. Then, the $\widetilde{C}(k, d)$ computed by the algorithm of Figure 3 upper bound C^s_U, $s \in \widetilde{U}_{k,d}$, and are decreasing on d.*

Proof. Consider the algorithm split into phases, where each phase includes the operations performed within the k-loop, and let $\widetilde{C}^m(k, d)$, $m \geq 0$, be the bounds $\widetilde{C}(k, d)$ available after phase m. The proof will be by induction over m. $\widetilde{C}^0(k, d) = \widetilde{C}(k)$, which are (non-strictly) decreasing on d and, by Theorem 2, upper bound C^s_U, $s \in U_k$. Assume now that the $\widetilde{C}^m(k, d)$ upper bound C^s_U, $s \in \widetilde{U}_{k,d}$, and are decreasing on d. Let k' be the value of k for which the bounds are updated in phase $m + 1$. According to (11), $C^{m+1}(k', d)$ only depend on $C^m(k, d)$ for $k \neq k'$, and all $C^{m+1}(k', d)$ are computed using the same set of bounds $C^m(k, d)$. Then, Proposition 1 guarantees that $C'(k', d)$ are correct, and Proposition 3 that they are decreasing on d. Using the induction hypothesis, this implies that $C^{m+1}(k', d) = \min\{C^m(k', d), C'(k', d)\}$ are correct and decreasing on d. $\square$

We conclude this section by deriving suitable upper bounds $\widetilde{F}(k, d, i, r)$ for $\sum_{d'=0}^{r} \lambda_{s,\widetilde{U}_{k+i,d'}}$, $s \in \widetilde{U}_{k,d}$. Trivially,

$$\widetilde{F}(k, d, i, \min\{d, N-k-i\}) = f_i$$

upper bounds $\sum_{d'=\max\{0,d-i\}}^{\min\{d,N-k-i\}} \lambda_{s,\widetilde{U}_{k+i,d'}}$. Let $\widetilde{\eta}(e)$, $e \in E$, be the lower bound for the failure distance from a state whose bag of failed components is e. Let $s' \in U$ be a state reached from $s \in \widetilde{U}_{k,d}$ through a transition which has associated with it the failure bag e. Since $F(s') = F(s) + e$, we have, by assumption A3, that $\widetilde{d}(s')$ cannot have been reduced with respect to $\widetilde{\eta}(e)$ by more than k, i.e. $\widetilde{d}(s') \geq \widetilde{\eta}(e) - k$. Then

$$\widetilde{F}(k, d, i, r) = \sum_{\substack{e \in E_i \\ \widetilde{\eta}(e) \leq k+r}} \lambda_{\mathrm{ub}}(e)\,,$$

$\max\{0, d-i\} \leq r < \min\{d, N-k-i\}$ upper bounds $\sum_{d'=\max\{0,d-i\}}^{r} \lambda_{s,\widetilde{U}_{k+i,d'}}$, $r < \min\{d, N-k-i\}$. Trivially, both $\widetilde{F}(k, d, i, r)$, $(k, d, i, r) \in \widetilde{\mathcal{R}}'$ and $\widetilde{F}(k, d, i, d) = f_i$, $(k, d, i, d) \in \widetilde{\mathcal{R}}'$, are (non-strictly) decreasing on d, thus fulfilling the conditions imposed by Theorem 3.

4. Lower Bounds for Failure Distances

In this section we derive lower bounds for failure distances fulfilling assumptions A1–A3 of Section 3.2. We also derive efficient algorithms to compute the bounds.

4.1. Definition of Lower Bounds for Failure Distances

We assume, without loss of generality, that the fault tree of the system is made up of a set P of AND and OR gates and a set I of inputs. We will denote by C the set of component types of the system, and by g_r the root gate of the fault tree. Each input has the form $c[n]$, $c \in C$, meaning the failure of n components of type c. The existence of types of components introduces some dependencies among the inputs of the fault tree. It will be said that two inputs are related if they involve components of the same type, i.e. are of the form $c[n]$, $c[n']$, $n \neq n'$. To avoid trivialities, we assume that there are no related inputs feeding the same gate. This is not a real restriction since for $n' > n$ and denoting by $\wedge$ and $\vee$ the logical "and" and "or" operators, respectively, $c[n] \vee c[n']$ can be substituted by $c[n]$ and $c[n] \wedge c[n']$ by $c[n']$. Each node x (i.e. an input or a gate) of the fault tree feeds a set of gates $\mathrm{fo}(x)$ and each gate y is fed by a set of nodes $\mathrm{fi}(y)$. Let $\mathrm{val}(x) \in \{0, 1\}$ be the value of node x. If x is a gate, $\mathrm{val}(x)$ is determined as usual from the values of its inputs. If $x = c[n]$ is an input, $\mathrm{val}(x) = 1$ if and only if n or more components of type c have failed. Since the value of a node is equal to 1 if and only if a suitable collection of components of the system being modeled have failed, nodes will also be referred to as events. In this regard, the event $x \in I \cup P$ will be said to be realized if $\mathrm{val}(x) = 1$.

Let us denote a bag of failed components, F, as $F = c_1[n_1]c_2[n_2]\ldots c_k[n_k]$, $c_l \neq c_m$, $l \neq m$, meaning that F contains n_i instances of component type c_i. It will be said that $c_i[n_i]$ is part of F. The distance from a bag of failed components F to an event $x \in I \cup P$, $d_b(F, x)$, is defined as the minimum number of components which have to fail in addition to those which are already part of F to realize x. From that definition, given the bag of failed components $F(s)$ of a state $s \in \Omega$, $d(s) = d_b(F(s), g_r)$. Let $\widetilde{d}_b(F, x)$, $x \in I \cup P$, be a lower bound for $d_b(F, x)$. The lower bounds $\widetilde{d}(s)$, $s \in \Omega$, are $\widetilde{d}(s) = \widetilde{d}_b(F(s), g_r)$. Such lower bounds will be computed on the fault tree using the concept of module, defined as a gate such that the subtree hanging from it has that gate as only exit point and every input of the subtree does not have related inputs outside the subtree. The gates which are modules can be determined using the algorithm LTA/DR of [10] with a small modification to take into account types of components: during the first depth-first, left-most traversal of the fault tree (step no. 2 of the algorithm), a visit to $c[n] \in I$ implies simultaneous visits (i.e. with the same "time stamp" as for $c[n]$) to all its related inputs.

Let F be a bag of failed components. $\widetilde{d}_b(F,x), x \in I \cup P$, is recursively defined as follows. If $x = c[n] \in I$:

$$\widetilde{d}_b(F,x) \equiv \begin{cases} n & \text{if no } c[n'] \text{ is part of } F \\ \max\{0, n-n'\} & \text{otherwise} \end{cases}; \quad (12)$$

if x is an OR gate:

$$\widetilde{d}_b(F,x) \equiv \min_{y \in \mathrm{fi}(x)} \{\widetilde{d}_b(F,y)\}\,; \quad (13)$$

and if x is an AND gate,

$$\widetilde{d}_b(F,x) \equiv \sum_{y \in A(x)} \widetilde{d}_b(F,y) \quad (14)$$
$$+ \max\Big\{ \sum_{y \in B(x)} \widetilde{d}_b(F,y)\,,\ \max_{y \in C(x)} \{0, \widetilde{d}_b(F,y)\}\Big\}\,,$$

where $A(x) \equiv \{y \in \mathrm{fi}(x) | \, y \text{ is a module} \wedge |\mathrm{fo}(y)| = 1\}$, $B(x) \equiv \{y \in \mathrm{fi}(x) | \, y \text{ is a module} \wedge |\mathrm{fo}(y)| > 1 \vee y \text{ is not a module} \wedge y \in I\}$ and $C(x) \equiv \{y \in \mathrm{fi}(x) | \, y \text{ is not a module} \wedge y \in P\}$.

The lower bounds for failure distances recursively defined by (12), (13) and (14) fulfill assumptions A1–A3 of Section 3.2 [20, Theorems 2 and 4].

4.2. An Algorithm for the Computation of Lower Bounds for Failure Distances

Expressions (12), (13) and (14) allow to compute $\widetilde{d}_b(F, g_r)$ traversing the fault tree depth-first, left-most starting at g_r. However, this procedure could be expensive if the fault tree is large. Next, we develop more efficient algorithms to compute $\widetilde{d}(s)$.

Each node x of the fault tree holds a "distance variable", $dv(x)$, and the fault tree is initialized so that $dv(x) = \widetilde{d}_b(\emptyset, x)$. Such an initialization is performed traversing depth-first, left-most the fault tree starting at g_r and using (12), (13) and (14). Note that after the initialization procedure, $\widetilde{L} = \widetilde{d}_b(\emptyset, g_r)$ is known. $\widetilde{d}_b(F(s),x) < \widetilde{d}_b(\emptyset,x)$, $x \in P$, requires $\widetilde{d}_b(F(s),y) < \widetilde{d}_b(\emptyset,x)$ for some $y \in \mathrm{fi}(x)$. Therefore, since $\widetilde{d}(s) \leq \widetilde{L}$, $\widetilde{d}(s) = \widetilde{d}_b(F(s),g_r)$ will be equal to $\widetilde{L}$ unless $\widetilde{d}_b(F(s),y) < \widetilde{L}$ for some $y \in \mathrm{fi}(x)$. The same argument can be iteratively applied going down the fault tree until the inputs. This justifies the following algorithm which computes $\widetilde{d}(s)$ processing the fault tree from inputs to g_r. For each $c[n]$ which is part of $F(s)$, based on (12) we make $dv(c[n']) = \max\{0, dv(c[n']) - n\}$ for each input $c[n']$. Each update of $dv(x)$ for an input x which results in $dv(x) < \widetilde{L}$ is propagated depth-first, left-most up the fault tree using (13) and (14) while $dv(z) < \widetilde{L}$ for the visited node z. Note that the algorithm computes the correct lower bounds for distances from $F(s)$ to the nodes x for which $\widetilde{d}_b(F(s),x) < \widetilde{L}$, so, at the end, $dv(g_r)$ will hold $\widetilde{d}(s)$. Note also that the initialization $dv(x) = \widetilde{d}_b(\emptyset, x)$ needs to be performed only once if while computing $\widetilde{d}(s)$ the changes in the $dv(x)$ variables are incrementally kept in a suitable data structure, e.g. a stack. We call generically the algorithm *comp_d*(F, ub, DS), where F is a bag of failed components, ub is an upper bound for the distance to be computed and *DS* is a stack. The algorithm returns $\widetilde{d}_b(F, g_r)$ if that value is $< ub$. In this regard, $\widetilde{d}(s) =$ *comp_d*$(F(s), \widetilde{L}, \mathit{DS})$. Once $\widetilde{d}(s)$ has been computed, the fault tree is restored to its initial state simply undoing the changes kept in *DS*. We call this procedure *restore_d*(*DS*).

Let s be a state in the frontier of G and let S be the set of states s' reached from s in a single failure transition. Computation of $[C_U]_{\mathrm{ub}}$ using (8) and (9) requires the computation of $\widetilde{d}(s')$, $s' \in S$. We describe next how $\widetilde{d}(s')$, $s' \in S$ are computed assuming that $\widetilde{\eta}(e)$ is known for all $e \in E$ ($\widetilde{\eta}(e) =$ *comp_d*$(e, \widetilde{L}, \mathit{DS})$). Each transition from s to $s' \in S$ has associated with it a failure bag $e_{s'} \in E$ and $F(s') = F(s) + e_{s'}$. Moreover, we have that $lb_{s'} = \max\{0, \widetilde{d}(s) - |e_{s'}|\} \leq \widetilde{d}(s') \leq \min\{\widetilde{d}(s), \widetilde{\eta}(e_{s'})\} = ub_{s'}$. The $dv(x)$ variables of the fault tree are set to $\widetilde{d}_b(F(s),x)$ if $\widetilde{d}_b(F(s),x) < \widetilde{L}$ and $\widetilde{d}_b(\emptyset,x)$ otherwise by calling *comp_d*$(F(s), \widetilde{L}, \mathit{DS})$. Next, for each $s' \in S$, $\widetilde{d}(s') = lb_{s'}$ if $lb_{s'} = ub_{s'}$. Otherwise, we set $d^* =$ *comp_d*$(e, ub_{s'}, \mathit{DS}')$. Because of the characteristics of the algorithm *comp_d*$(\cdot)$, we will have $d^* = \widetilde{d}(s')$ if $\widetilde{d}(s') < ub_{s'}$ and $d^* = \widetilde{d}(s)$ otherwise. Therefore, since $ub_{s'} \leq \widetilde{d}(s)$, $\widetilde{d}(s') = \min\{d^*, ub_{s'}\}$. Next, we call *restore_d*(*DS*') to allow another $\widetilde{d}(s')$ to be computed, and once $\widetilde{d}(s')$ has been computed for all $s' \in S$, we call *restore_d*(*DS*) to restore the fault tree.

5. Analysis and Comparison

In this section we analyze the performance of the proposed bounding method and compare it with the bounding method proposed in [2] using the same state space exploration algorithm. The bounding method proposed in [2] is analogous to the one proposed here except that it uses exact failure distances computed using the set of minimal cuts of the fault tree of the system, and bounding structures $F(k,d,i,r)$ upper bounding $\sum_{d'=0}^{r} \lambda_{s,U_{k+i,d'}}$, $s \in U_{k,d}$, the subset $U_{k,d}$ including the states with k failed components and failure distance d. We will also compare the method proposed here with the bounding method proposed in [15]. In that method,

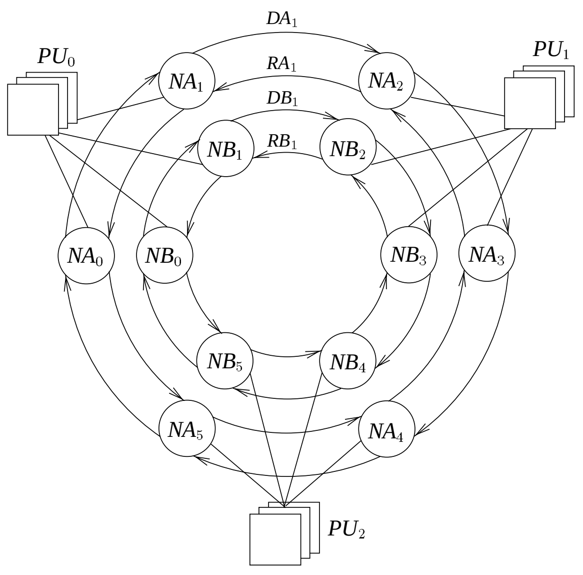


Figure 4. Architecture of the first example.

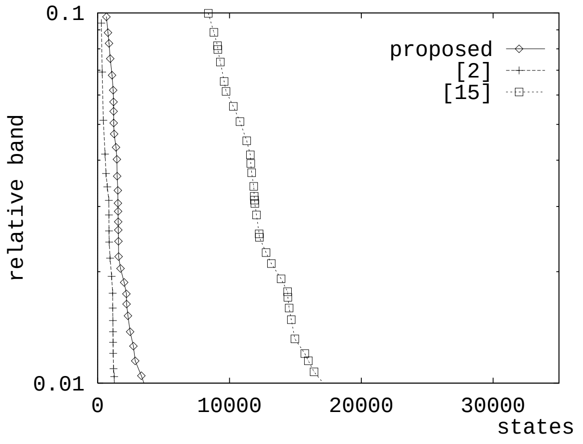
the lower bound for *UA* is also $[UA]_{\rm lb}$ but the upper bound is

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

In all cases the subset G is incrementally generated until the relative unavailability band, $([UA]_{\rm ub} - [UA]_{\rm lb})/[UA]_{\rm lb}$, is smaller than or equal to the desired one. For the proposed bounding method and the method proposed in [2] the generation is done using the algorithm CONT_TG_W proposed in [5]. For the bounding method described in [15] we use an analogous algorithm CONT_TG_W, where the contributions to the relative unavailability band are associated only with the parameter k (number of failed components of the successors). Both algorithms allow to tradeoff the number of times $(\tau(s, Y_G))_{s \in G}$ is computed against how accurately the state space is explored by means of a control parameter BR, $0 \leq BR < 1$ (the larger BR, the more accurate but more costly the exploration). After some experimentation we have found $BR = 0.1$ to be a reasonable choice. The R parameter used in the algorithm for the computation of exact failure distances described in [2] was set to 2.

The analysis and comparison will be made using two examples. The first example, whose architecture is depicted in Figure 4, includes three processing clusters which communicate through two independent double-ring networks A and B. Processing cluster i, $0 \leq i \leq 2$, includes three identical processing units PU_i. Network A includes six nodes NA_i, $0 \leq i \leq 5$, and direct (clockwise) and reverse (counter-clockwise) links, DA_i and RA_i, respectively, linking nodes NA_i and $NA_{i+1 \bmod 6}$. Network B has the same structure as network A and its direct and reverse links are called, respectively, DB_i and RB_i. The system is operational if each processing cluster has at least an unfailed processing unit and all processing clusters can communicate using one of the networks. The operational configuration of the system includes two processing units for the processing clusters with two or three unfailed processing units, one processing unit for the processing clusters with one unfailed processing unit, and the components of either network A or B, with priority given to network A, required to build one of the operational configurations of the networks described next. The network configuration which is tried first is a direct ring including all nodes and direct links. The second configuration which is tried is a reverse ring including all nodes and reverse rings. The third configuration is used when parallel direct and inverse link i fail and it includes all nodes and links except the links between nodes i and $i + 1 \bmod 6$. The last configuration is used when node i fails and it includes all nodes except node i and all links except those between node i and nodes $i \pm 1 \bmod 6$. A fault in a processing unit of a cluster contaminates another unfailed unit in the same cluster with probability 0.05. The components included in the operational configuration of the system are called active. Active processing units, active nodes and active links fail with rates 4.6×10^{-4} h^{-1}, 2.3×10^{-4} h^{-1} and 1.1×10^{-4} h^{-1}. Inactive components fail with the same rates multiplied by a dormancy factor of 0.2. We assume that there is a single repairman who takes for repair failed components at random. Repair rates for processing units, nodes and links are, respectively, 0.5 h^{-1}, 0.7 h^{-1} and 1.0 h^{-1}. Components continue to fail when the system has failed. The second example is as the first one but with the number of nodes of both networks increased up to ten and the number of processing clusters increased up to five. For both examples $L = 3$ and $\widetilde{L} = 2$. The fault tree has 8,653 minimal cuts for the first example and 87,031 for the second one.

We show in Figure 5 the relative unavailability band as a function of the number of states in G for the proposed bounding method and the methods described in [2] and [15]. The results have been obtained in a 128 MB UltraSparc workstation. It can be seen that the proposed bounding method outperforms significantly the bounding method described in [15] in terms of the size of G. Thus, for the first example, the number of states required by the method described in [15] to achieve a given relative unavailability band ranges from 4.9 to 12.6 times the number of states required by the proposed bounding method. With regard to the method described in [2], the proposed method requires a number of states about 2.6 times larger. However, the bounding method described in [2] requires to bookkeep in memory the minimal cuts and related data structures, and this overhead may make the memory consumption (which is really the parameter of interest) larger than that of the proposed method. Both the



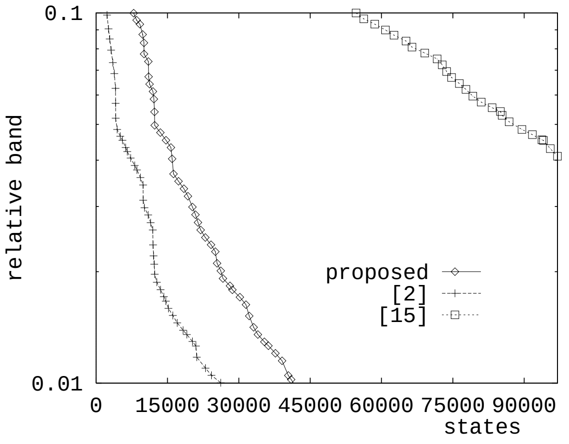


Figure 5. Relative unavailability band as a function of the number of states in G for the proposed bounding method, the method described in [2] and the method described in [15], for the example with three processing clusters (left) and five processing clusters (right) and $BR = 0.1$.

proposed bounding method and the method described in [2] have to hold, for each state in the frontier of G, lists of contributions to the unavailability band associated with the parameters k and d while the lists of contributions which have to be held for those states in the method described in [15] are associated only with the parameter k. Then, it is meaningful to compare the three bounding methods in terms of memory consumption. The comparison is done in Figure 6, which plots unavailability relative band against (estimated) memory consumption. The proposed bounding method is again far more efficient than the method described in [15] and more efficient than the method described in [2]. The smaller the relative unavailability band, however, the smaller the difference in terms of memory consumption between the proposed method and the method described in [2]. This is due to the fact that $|G|$ increases and, thereby, the memory consumption due to storing the state descriptions of G and the lists of contributions to the unavailability band becomes relatively more important than the overhead introduced by storing the minimal cuts and related data structures.

The overhead due to the computation of lower bounds for failure distances is negligible regarding memory consumption. The overhead in terms of CPU time consumption depends on the size of the fault tree. The fault tree of the first example has 39 inputs, 32 gates and 431 edges. The fault tree of the second example has 65 inputs, 48 gates and 1,193 edges. We have profiled our code and have found a time overhead due to the computation of lower bounds for failure distances of 7.8% in the first example with $|G|$ =8,608 states and 15.8% in the second example with $|G|$ =9,568 states. Then, although increasing with the size of the fault tree, the overhead in CPU time due to the computation of lower bounds for failure distances is reasonable.

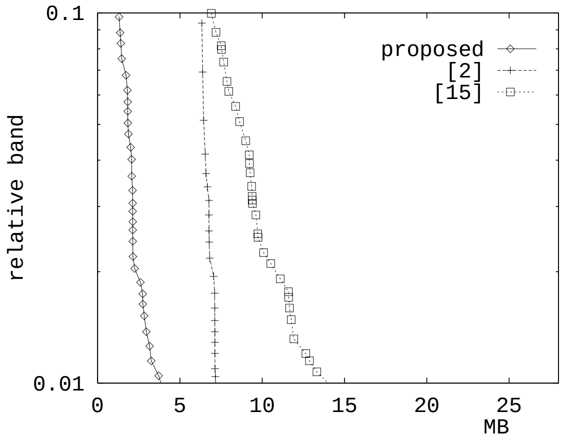
6. Conclusions

In this paper we have proposed a new method to compute bounds for the steady-state unavailability, which is based on lower bounds for failure distances. We have developed algorithms to compute such distances on the fault tree of the system. The proposed bounding method generates incrementally a subset of the state space using a previously proposed state space exploration algorithm. Numerical analysis has shown that in terms of number of states needed to achieve a given accuracy, the proposed method outperforms a previously proposed method not based on the failure distance concept and is worse than a previously proposed method which uses exact failure distances. However, when compared in terms of memory consumption, the proposed bounding method can outperform the bounding method which uses exact failure distances if the number of minimal cuts is large. Therefore, the proposed method is a good tradeoff between bounds tightness and memory consumption when the number of minimal cuts of the system is large.

References

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

[2] J. Carrasco. Tight steady-state availability bounds using the failure distance concept. *Performance Evaluation*, 34:27–64, 1998.



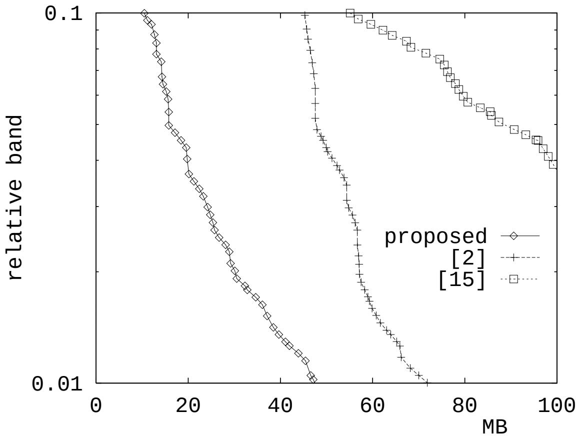


Figure 6. Relative unavailability band as a function of (estimated) memory consumption in MB for the proposed bounding method, the method described in [2] and the method described in [15], for the example with three processing clusters (left) and five processing clusters (right) and $BR = 0.1$.

[3] J. Carrasco. Bounding steady-state availability models with group repair and phase type repair distributions. *Performance Evaluation*, 35:193–214, 1999.

[4] J. Carrasco, A. Calderón, and J. Escrivá. Two new algorithms to compute steady-state bounds for Markov models with slow forward and fast backward transitions. In *Proc. 4th Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, pages 89–95, February 1996.

[5] J. Carrasco, J. Escrivá, and A. Calderón. Efficient exploration of availability models guided by failure distances. *Performance Evaluation Review*, 24(1):242–251, May 1996.

[6] J. Carrasco and V. Suñé. An algorithm to find minimal cuts of s-coherent fault trees with event classes using a decision tree. *IEEE Transactions on Reliability*, 48(1):31–41, March 1999.

[7] E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[8] O. Coudert and J. C. Madre. Metaprime: An interactive fault-tree analyzer. *IEEE Transactions on Reliability*, 43(1):121–127, March 1994.

[9] E. de Souza e Silva and P. M. Ochoa. State space exploration in Markov models. *Performance Evaluation Review*, 20(1):152–166, June 1992.

[10] Y. Dutuit and A. Rauzy. A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability*, 45(3):422–425, September 1996.

[11] W. Hennings and N. Kuznetsov. FAMOCUTN & CUTQN: Programs for fast analysis of large fault trees with replicated & negated gates. *IEEE Transactions on Reliability*, 44(3):368–376, September 1995.

[12] J. C. S. Lui and R. R. Muntz. Evaluating bounds on steady-state availability of repairable systems from Markov models. In W. J. Stewart, editor, *Numerical Solution of Markov Chains*, pages 435–454. Marcel Dekker, New York, 1991.

[13] J. C. S. Lui and R. R. Muntz. Computing bounds on steady-state availability of repairable computer systems. *Journal of the ACM*, 41(4):676–707, July 1994.

[14] S. Mahévas and G. Rubino. Bounding asymptotic dependability and performance measures. In *Proc. 2nd IEEE Int. Performance and Dependability Symposium*, pages 176–186, Urbana-Champaign, USA, September 1996.

[15] R. R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. *IEEE Transactions on Computers*, 38(12):1714–1723, 1989.

[16] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, London, 1981.

[17] A. Rauzy. New algorithms for fault trees analysis. *Reliability Engineering and System Safety*, 40:203–211, 1993.

[18] A. Rosenthal. A computer scientist looks at reliability computations. In R. Barlow, J. Fusell, and N. Singpurwalla, editors, *Reliability and Fault Tree Analysis*, pages 133–152. SIAM, Philadelphia, 1975.

[19] P. Semal. Refinable bounds for large Markov chains. *IEEE Transactions on Computers*, 44(10):1216–1222, October 1995.

[20] V. Suñé and J. Carrasco. A failure-distance based method to bound the reliability of non-repairable fault-tolerant systems without the knowledge of minimal cuts. Technical report, Universitat Politècnica de Catalunya, January 1999. Available at ftp://ftp-eel.upc.es/techreports under the name DMSD_99_1.ps.