Interval Availability Analysis Using Denumerable Markov Processes: Application to Multiprocessor Subject to Breakdowns and Repair

Gerardo Rubino and Bruno Sericola

Abstract—**Interval availability is a dependability measure defined by the fraction of time during which a system is operational over a finite observation period. The computation of its distribution allows the user to ensure that the probability that its system will achieve a given availability level is high enough.**

The system is assumed to be modeled as a Markov process with countable state space. We propose a new algorithm to compute the interval availability distribution. One of its main advantages is that, in some cases, it applies even to infinite state spaces. This is useful, for instance, in case of models taking into account contention with unbounded buffers. This important feature is illustrated on models of multiprocessor systems subject to breakdowns and repair. When the model is finite, we show through a numerical example that the new technique can perform very well.

Index Terms— **Denumerable Markov processes, dependability prediction, interval availability distribution, repairable computer systems, transient analysis, queues with breakdowns, uniformization.**

I. Introduction

In the dependability analysis of repairable computing systems, there is an increasing interest in evaluating cumulative measures, in particular, the availability over a given period. In highly available systems, steady state measures can be very poor, even if the mission time is not small. The use of expectations also suffers from similar drawbacks. Considering, for instance, critical applications, it is crucial for the user to ensure that the probability that its system will achieve a given availability level is high enough. This paper deals with the computation of the distribution of the interval availability which is defined by the fraction of time during which a system is in operation over a finite observation period.

Formally, the system is modeled by a Markov process whose state space is divided into the subset of *up* states and the subset of *down* states. The interval availability over $(0,t)$ is then the fraction of the interval $(0,t)$ during which the process is in the *up* states. This random variable has been studied in previous papers as for instance in [1] where its distribution is calculated recursively by discretizing the observation period $(0,t)$. However, no error bounds were found for this approximation method. In [2], a particular algorithm has been developed in the case where the two sequences of sojourn times in the up and down states are independent one by one and of each other. It is moreover shown in [2] that this property can directly be checked on the transition rate matrix of the process. In [3] a method based on the uniformization technique is proposed. The algorithm is interesting because it has good numerical properties and, moreover, it allows the user to perform the computation with an error as small as desired. Improvements of this approach have been developed in [4].

In this paper we develop a new algorithm to compute the interval availability distribution, based on the uniformization technique and on the results obtained in [3] and [4]. Its space complexity depends strongly on the number of operational states. This is of interest especially when this number is small with respect to the size of the whole state space. Another important characteristic of the new algorithm is that it applies for a large class of processes with an infinite state space provided that either the number of up states or the number of down states is finite.

The remainder of the paper is organized as follows. In the following section, we describe Algorithm II of [4], which is the basis of the new proposed method. In Section III, we present the new algorithm, we illustrate its performance through a finite state space example and we compare it with Algorithm II of [4]. In Section IV, a generic example of multiprocessor systems subject to breakdowns and repair is used to show the ability of the new algorithm to deal with infinite state spaces. Numerical results are also provided. Section V proposes some conclusions.

II. Interval Availability Distribution Computation

Consider a continuous-time homogeneous Markov process $X = \{X_t, t \geq 0\}$, over a finite state space denoted by S. The states of S are divided into two disjoint subsets: U, the set of the operational or up states and D, the set of the unoperational or down states. We assume that the system, modeled by such a process, is considered during a finite interval of time denoted by $(0,t)$. The interval availability over $(0,t)$, that is, the fraction of time the system is in operation during $(0,t)$, is defined by

$$\mathrm{IAV}(t) = \frac{1}{t}\int_0^t \mathbf{1}_{\{X_s \in U\}} ds,$$

Manuscript received December 1993; revised April 1994. This work was supported in part by the Esprit project Fasst, # 5212.

The authors are with IRISA-INRIA, Campus Universitaire de Beaulieu, F-35042 Rennes Cedex, France.

IEEE Log Number 9407241.

where $\mathbf{1}_{\{c\}}$ is equal to 1 if condition c is true and 0 otherwise. The process X is, as usual, given by its infinitesimal generator, denoted by A, in which the ith diagonal entry $A(i,i)$ verifies $A(i,i) = -\sum_{j \neq i} A(i,j)$ and by its initial probability distribution α.

Let us denote by Z the uniformized Markov chain with respect to the uniformization rate λ and by P its transition probability matrix [5]. The uniformization rate λ verifies $\lambda \geq \sup(-A(i,i), i \in S)$ and P is related to A by $P = I + A/\lambda$, where I denotes the identity matrix. We decompose P and the initial probability (row) vector α with respect to the partition $\{U, D\}$ of S as

$$P = \begin{pmatrix} P_U & P_{UD} \\ P_{DU} & P_D \end{pmatrix}, \alpha = (\alpha^U, \alpha^D).$$

It has been shown in [4] that for every $p < 1$, we have

$$P(\mathrm{IAV}(t) > p) = \sum_{n=0}^{+\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k p^k q^{n-k} Y_{n,k} \quad (1)$$

where $q = 1 - p$ and $Y_{n,k}$ is the probability that the uniformized Markov chain Z visits more than k states of U during the first n transitions. Let $Y_{n,k}^S$ (resp. $Y_{n,k}^U$, $Y_{n,k}^D$) be the column vector such that its ith entry, $i \in S$ (resp. $i \in U$, $i \in D$), is the probability that Z visits more than k states of U during the first n transitions, given that the initial state is i. We have $Y_{n,k} = \alpha Y_{n,k}^S$. Vectors $Y_{n,k}^U$ and $Y_{n,k}^D$ can be computed using the following recurrence relations [4]:

$$Y_{n,k}^U = P_U Y_{n-1,k-1}^U + P_{UD} Y_{n-1,k-1}^D$$

and

$$Y_{n,k}^D = P_{DU} Y_{n-1,k}^U + P_D Y_{n-1,k}^D \quad (2)$$

with initial conditions $Y_{n,0}^U = 1^T$ and $Y_{0,0}^D = 0^T$, where 1^T (resp. 0^T) denotes a column vector with each entry equal to 1 (resp. to 0), $()^T$ denoting the transpose operator. From relation (1), the computation proceeds as follows. It is shown in [4, Section 4] that if $H_{t,p}$ is defined by

$$H_{t,p} = \sum_{k=0}^{N-C'} \sum_{n=k}^{C'+k} e^{-\lambda t} \frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} Y_{n,k} + \sum_{k=N-C'+1}^{N-C''-1} \sum_{n=k}^{N} e^{-\lambda t} \frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} Y_{n,k}, \quad (3)$$

then $0 \leq P(\mathrm{IAV}(t) < p) - H_{t,p} \leq \varepsilon$, where, for a given specified error tolerance ε, integers N, C', and C'' are chosen as

$$N = \min\left\{ n \in N \middle| \sum_{j=0}^{n} e^{-\lambda t} \frac{(\lambda t)^j}{j!} \geq 1 - \frac{\varepsilon}{2} \right\},$$

$$C' = \min\left\{ c \leq N \middle| \sum_{k=0}^{c} e^{-\lambda t q} \frac{(\lambda t q)^k}{k!} \geq 1 - \frac{\varepsilon}{4} \right\}$$

and

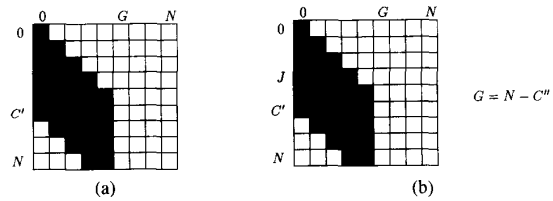


Fig. 1. (a) Algorithm A. In cell (n,k):$Y_{n,k}^S$. (b) Algorithm B. In cell (n,k):$Z_{n,k}^U$.

$$C'' = \max\left\{ c \leq N \middle| \sum_{k=0}^{c} e^{-\lambda t q} \frac{(\lambda t q)^k}{k!} \leq \frac{\varepsilon}{4} \right\}.$$

The general computational scheme of the $Y_{n,k}$'s is shown in Fig. 1(a), where we indicate in gray the cells that are effectively filled. The contents of cell in row n and column k is vector $Y_{n,k}^S$. This algorithm is now called Algorithm A. When C'' does not exist, we can define $C'' = -1$ and still use relation (3). In this case the error tolerance $\varepsilon/4$ in the definition of C' can be replaced by $\varepsilon/2$. Following the recurrence relation (2), the computation can be made in a column by column manner so that we only need to store $C' + 1$ vectors with dimension equal to the number of states of process X, as shown in Fig. 1(a).

III. A New Algorithm

The new algorithm, denoted Algorithm B in the sequel, also computes the $Y_{n,k}$'s numbers but in a different way. We assume first, for simplicity, that the system starts in an operational state, that is we suppose that $\alpha^U 1^T = 1$. The general case is detailed in Appendix A. With this assumption, we have, for every $n \geq 0$ and $0 \leq k \leq n$, $Y_{n,k} = \alpha^U Y_{n,k}^U$. Relations (2) can be partially solved to get

$$Y_{n,k}^U = \sum_{j=1}^{n-k+1} P_j Y_{n-j,k-1}^U, \quad (4)$$

where the square matrices P_j's, whose dimension is equal to the number of operational states, are defined by $P_1 = P_U$ and $P_j = P_{UD} P_D^{j-2} P_{DU}$ for $j \geq 2$. A probabilistic interpretation of (4) is discussed in Appendix A. Using now relation (3), we see that the number of P_j's matrices involved in (4) is equal to $C' + 1$. This number can be reduced when the Markov process X is not absorbing in the following way. If X is not absorbing, the matrix $\sum_{j=1}^{+\infty} P_j$ is stochastic. Let $\varepsilon' > 0$ and let J be the smallest integer not greater than $C'+1$ such that $\sum_{j=1}^{J} P_j 1^T \geq (1 - \varepsilon') 1^T$ where an inequality between vectors means that the inequality holds for each entry. Instead of computing the $Y_{n,k}$'s numbers, we will work with $Z_{n,k} = \alpha^U Z_{n,k}^U$ where $Z_{n,k}^U$ is the column vector defined by

$$Z_{n,k}^U = \sum_{j=1}^{\min(J,n-k+1)} P_j Z_{n-j,k-1}^U, \quad (5)$$

with initial condition $Z^U_{n,0} = Y^U_{n,0} = 1^T$. Relation (3) can then be written as

$$H_{t,p} = \sum_{k=0}^{N-C'} \sum_{n=k}^{C'+k} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} Z_{n,k} + \sum_{k=N-C'+1}^{N-C''-1} \sum_{n=k}^{N} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} Z_{n,k} + \text{err}. \quad (6)$$

Defining $E_{n,k} = Y_{n,k} - Z_{n,k}$, the value err verifies

$$\text{err} = \sum_{k=1}^{N-C'} \sum_{n=J+k}^{C'+k} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} E_{n,k} + \sum_{k=N-C'+1}^{N-C''-1} \sum_{n=J+k}^{N} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} E_{n,k}. \quad (7)$$

The following lemma gives a way to bound the value err. It is proved in Appendix B.

Lemma 3.1: For every $n \geq 1$ and $1 \leq k \leq n$ such that $n - k + 1 > J$, we have $E_{n,k} \leq k\varepsilon'$.

Using this lemma, we can then write from relation (7),

$$\begin{aligned}
\text{err} &\leq \varepsilon' \sum_{k=1}^{N-C'} k \sum_{n=J+k}^{C'+k} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} + \varepsilon' \sum_{k=N-C'+1}^{N-C''-1} k \sum_{n=J+k}^{N} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k p^k q^{n-k} \\
&= \varepsilon' \sum_{k=1}^{N-C'} k e^{-\lambda tp}\frac{(\lambda tp)^k}{k!} \sum_{n=J}^{C'} e^{-\lambda tq}\frac{(\lambda tq)^n}{n!} + \varepsilon' \sum_{k=N-C'+1}^{N-C''-1} k e^{-\lambda tp}\frac{(\lambda tp)^k}{k!} \sum_{n=J}^{N-k} e^{-\lambda tq}\frac{(\lambda tq)^n}{n!} \\
&\leq \varepsilon' \sum_{k=1}^{N-C''-1} k e^{-\lambda tp}\frac{(\lambda tp)^k}{k!} \sum_{n=J}^{C'} e^{-\lambda tq}\frac{(\lambda tq)^n}{n!} \\
&= \varepsilon' \lambda tp \sum_{k=0}^{N-C''-2} e^{-\lambda tp}\frac{(\lambda tp)^k}{k!} \sum_{n=J}^{C'} e^{-\lambda tq}\frac{(\lambda tq)^n}{n!}.
\end{aligned}$$

To have err $\leq \varepsilon/6$, it suffices to choose ε' such that

$$\varepsilon' = \varepsilon \left[6\lambda tp \sum_{k=0}^{N-C''-2} e^{-\lambda tp}\frac{(\lambda tp)^k}{k!} \sum_{n=J}^{C'} e^{-\lambda tq}\frac{(\lambda tq)^n}{n!} \right]^{-1},$$

the value of N being the same as for Algorithm A and the values of C' and C'' being computed with respect to $\varepsilon/6$ instead of $\varepsilon/4$. Algorithm B consists then in computing the $Z_{n,k}$'s numbers using (5), in a column by column manner as shown in Fig. 1(b). The sum of the first two terms in the right hand side of (6) gives $P(\text{IAV}(t) > p)$ with a tolerance equal to ε.

TABLE I
COMPUTATION TIMES OF ALGORITHMS A AND B ON AN EXAMPLE (256 STATES WHERE 9 ARE OPERATIONAL)

p	$\mathbb{P}(IAV(100) > p)$	A	B	$\mathbb{P}(IAV(1000) > p)$	A	B
0.990	0.9120	2:42 min	1:24 min	0.9989	2:08 hrs	2:55 hrs
0.995	0.8159	1:48 min	0:48 min	0.9054	1:16 hrs	1:12 hrs
0.999	0.6685	0:54 min	0:18 min	0.1619	0:26 hrs	0:11 hrs

A. Complexity Analysis

Let us denote by M the total number of states and by L the number of operational states. The space complexity of the new algorithm is bounded by $O(C'L^2)$. This follows from the fact that we need to store $C' + 1$ vectors of dimension L plus J square matrices (the P_j's) of dimension $L \times L$, and from the relationship $J \leq C' + 1$. Of course, this can be reduced if sparse techniques are used.

To evaluate the time complexity, let us consider first the computation of the P_j's matrices. If η is the number of nonnull elements in P, assuming that each matrix-matrix product when building the sequence needs $O(\eta L(M - L))$, we obtain a total cost in $O(\eta JL(M - L))$, which can be bounded by $O(\eta C'L(M - L))$ since $J \leq C' + 1$. The rest of the cost is proportional to the number of cells to be computed. Using the relationships $J \leq C' + 1$ and $C'' < C'$, we obtain a cost in $O(L^2(NC'^2/2 + (C'^3 - C''^3)/6))$ for computing all the needed cells.

These results show first the interest of the new algorithm when the number L of up states is small. Second, we can observe that Algorithm B behaves better if p is close to one: observe that C' is $O(\lambda tq)$ and recall that $q = 1 - p$.

1) Numerical Examples: Let us consider the hardware system, described in [4], which consists of n identical components behaving independently, with failure rate equal to 0.01 and repair rate equal to 1. We assume that the maintainance policy is unrestricted. This leads to a Markov model with $M = 2^n$ states. The system operates on a "k out of n" basis, that is, the system is up when at least k components are up. We note that the state space can be reduced to $n + 1$ states; here, it is not simplified for testing purposes. We compute the probability of achieving an availability of at least p, that is $P(\text{IAV}(t) > p)$, for different values of the parameters p and t. The chosen global error ε is equal to 0.00001. Table I shows the computation times of Algorithms A and B in the case of $n = 8$ and $k = 7$, for $p = 0.990, 0.995, 0.999$ and for $t = 100, 1000$. This leads to a Markov process with 256 states, 9 of them being operational. The programs were run on a Sun 4/50.

IV. MULTIPROCESSOR SYSTEMS

In this section, we illustrate the application of Algorithm B to an infinite model. We consider a system where K identical parallel processors serve a common, unbounded job queue. The processors break down from time to time, and return to an operational state after being repaired. When operational, each processor serves jobs one at a time and each job is

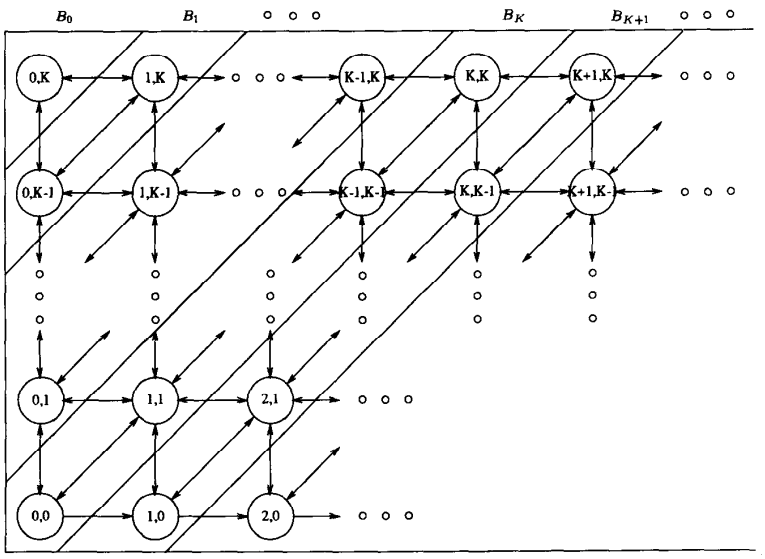


Fig. 2. Partition of the state space and transitions between states.

allowed to occupy at most one operational processor at a time. It is assumed that the evolution of the system is modeled by an irreducible Markov process X with state space $S = \{0, 1, \cdots\} \times \{0, 1, \cdots, K\}$. The system is in state $(i, j) \in S$ when there are i jobs (including the jobs being in the servers) and j operational processors. The evolution of X proceeds according to the following set of possible transition rates:

[a] $(i,j) \overset{\beta(i,j)}{\rightarrow} (i+1,j)$ [b] $(i+1,j) \overset{\mu(i,j)}{\rightarrow} (i,j)$

[c] $(i,j) \overset{\delta(i,j)}{\rightarrow} (i,j-1)$ [d] $(i,j-1) \overset{\gamma(i,j)}{\rightarrow} (i,j)$

[e] $(i+1,j+1) \overset{\nu(i,j)}{\rightarrow} (i,j)$ [f] $(i,j) \overset{\eta(i,j)}{\rightarrow} (i+1,j+1)$.

A transition of type [a] (resp. [b]) represents a job arrival (resp. a job departure). Transitions of types [c], [e] (resp. [d], [f]) represent the breakdown (resp. the repair) of a processor. This enables the consideration of models with and without loss of jobs. The purpose of the analysis is to determine the distribution of the percentage of time over the interval $(0, t)$ during which there are no jobs in the waiting room of the queue. Let us define the partition $(B_k, k \geq 0)$ of the state space S as follows. For $0 \leq k \leq K$, we define $B_k = \{(k-i, K-i), 0 \leq i \leq k\}$ and for $k \geq K+1$, we define $B_k = \{(k-i, K-i), 0 \leq i \leq K\}$. This partition of the state space and the transitions between the states are illustrated in Fig. 2.

Now, let us take as subset of operational states $U = \bigcup_{k=0}^{K} B_k$, and then as subset of down states $D = \bigcup_{k=K+1}^{+\infty} B_k$. To use the uniformization technique to evaluate the distribution of $\mathrm{IAV}(t)$, we need to assume that the infinitesimal generator A of process X is uniformly bounded. This assumption is not a practical restriction and can be expressed as: $\sup\{-A(u,u), u \in S\} < \infty$. This value will be taken as the uniformization rate λ. Clearly, the results in [3] and [4] cannot apply here since the submatrices P_{UD}, P_D, and P_{DU} are infinite. The number of states in subset U is equal to $(K+2)(K+1)/2$ and the number of states in subset D is infinite. Algorithm B can apply if the square matrices P_j, $j \geq 1$, which are of dimension $(K+2)(K+1)/2$, can be computed.

A. Computation of the Matrices P_j

From the partition $(B_k, k \geq 0)$ of the state space S, the infinitesimal generator A of X becomes a block tridiagonal matrix. For every nonnegative integers k and m, let $A_{k,m}$ (resp. $P_{k,m}$) be the block corresponding to transition rates (resp. transition probabilities) from subset B_k to subset B_m. It follows (see Fig. 2) that the only nonzero blocks are blocks $A_{0,0}$, $A_{0,1}$, $A_{k,k-1}$, $A_{k,k}$, and $A_{k,k+1}$ for $k \geq 1$. Obviously, matrix P has the same structure. Recall that matrices P_j, $j \geq 1$, are defined by $P_1 = P_U$ and $P_j = P_{UD} P_D^{j-2} P_{DU}$, for $j \geq 2$. Now, since the only nonzero block of matrix P_{UD} (resp. matrix P_{DU}) is block $P_{K,K+1}$ (resp. block $P_{K+1,K}$), matrices P_j can be written as $P_j = \begin{pmatrix} 0 & 0 \\ 0 & F_j \end{pmatrix}$, for $j \geq 2$, where the matrices F_j are of dimension $K+1$, for every $j \geq 2$. To compute matrices F_j, for every $j \geq 2$, we proceed as follows. Since P_D is a block tridiagonal matrix, for every integer $j \geq 1$ the matrix P_D^j, for which we denote by $P_{k,m}^{(j)}$ the square block of dimension $K+1$ corresponding to transitions from subset B_k to subset B_m, is such that the blocks $P_{K+k,K+j+m}^{(j)}$ and $P_{K+j+m,K+k}^{(j)}$ are equal to 0 for every $m \geq k+1$. This is because the jth power ($j \geq 1$) of any tridiagonal matrix is a $(2j+1)$-diagonal matrix (i.e., having $2j+1$ diagonals). We then have

$$F_j = P_{K,K+1} P_{K+1,K+1}^{(j-2)} P_{K+1,K}, \text{ for } j \geq 2.$$

Writing now $P_D^j = P_D P_D^{j-1}$, we get the following recurrence: for every $j \geq 1$ and $2 \leq i \leq j$,

$$P_{K+1,K+1}^{(j)} = P_{K+1,K+1} P_{K+1,K+1}^{(j-1)} + P_{K+1,K+2} P_{K+2,K+1}^{(j-1)}$$

$$P^{(j)}_{K+i,K+1} - P_{K+i,K+i-1}P^{(j-1)}_{K+i-1,K+1} + P_{K+i,K+i}P^{(j-1)}_{K+i,K+1} + P_{K+i,K+i+1}P^{(j-1)}_{K+i+1,K+1}$$

$$P^{(j)}_{K+j+1,K+1} = P_{K+j+1,K+j}P^{(j-1)}_{K+j,K+1} + P_{K+j+1,K+j+1}P^{(j-1)}_{K+j+1,K+1}.$$

For $j \geq 1$ and $1 \leq i \leq j$, let $H_i^{(j)}$ be the square matrix of dimension $K+1$ defined by

$$H_i^{(j)} = P^{(j)}_{K+i,K+1}P_{K+1,K}.$$

The previous recurrence becomes, for every $j \geq 1$ and $2 \leq i \leq j$,

$$H_1^{(j)} = P_{K+1,K+1}H_1^{(j-1)} + P_{K+1,K+2}H_2^{(j-1)}$$

$$H_i^{(j)} = P_{K+i,K+i-1}H_{i-1}^{(j-1)} + P_{K+i,K+i}H_i^{(j-1)} + P_{K+i,K+i+1}H_{i+1}^{(j-1)}$$

$$H_{j+1}^{(j)} = P_{K+j+1,K+j}H_j^{(j-1)} + P_{K+j+1,K+j+1}H_{j+1}^{(j-1)}$$

with initial conditions $H_1^{(0)} = P_{K+1,K}$ and $H_2^{(0)} = 0$. Finally, we obtain $F_j = P_{K,K+1}H_1^{(j-2)}$ for $j \geq 2$. To compute the distribution of $\mathrm{IAV}(t)$ using Algorithm B, we have first to compute matrices $F_2, \cdots, F_J$, where integer J is given in the description of that algorithm.

V. Application and Numerical Results

To illustrate the obtained results, let us consider a $M/M/K$ queue with arrival rate β and service rate μ. Each server breaks down with constant rate δ and is repaired, assuming there is only one repairman, with constant rate γ. We assume that when a processor breaks down, the job (if any) that was being processed by this processor is not lost. This application leads to the following values of the general rates introduced in the previous section. For every $(i,j) \in S$, $\beta(i,j) = \beta$, $\mu(i,j) = (i+1)\mu$ if $i+1 \leq j$ and $j\mu$ otherwise, $\delta(i,j) = j\delta$, $\gamma(i,j) = \gamma$ and $\nu(i,j) = \eta(i,j) = 0$.

Fig. 3 shows the probability that the percentage of time during which there is no job waiting in the queue is greater than 0.99, with an error tolerance $\varepsilon = 0.00001$, as a function of the number of processors. The arrival rate is $\beta = 2.0$. For each processor, the service rate is $\mu = 1.0$, the failure rate $\delta = 0.0001$ and the repair rate is $\gamma = 0.1$.

It is interesting to note the important difference in the results for 5 and 6 processors when $t = 100$: during the interval $(0, 100)$, the queue will be empty more than 99% of the time with probability 0.2538 for 5 processors and with probability 0.7924 for 6 processors.

VI. Conclusion

The contribution of this paper is to provide a new algorithm to compute the interval availability distribution for Markov processes. A main feature of the algorithm is that it applies not only for finite models but also when the state space is infinite, if at least one of the two involved subsets of states (operational and unoperational) is finite and when the infinitesimal

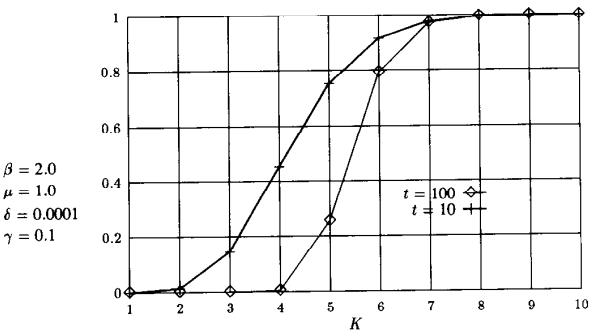


Fig. 3. $P(\mathrm{IAV}(t) > 0.99)$ as a function of the number K of processors.

generator of the process is uniformizable and block tridiagonal. The method is also shown to be computationally interesting for finite state models when the number of operational states (resp. nonoperational states) is small (resp. large, see Appendix A) compared to the total number of states. For instance, it can be very efficient if we consider phase type repair since in that case the number of nonoperational states can be large.

Appendix A
General Initial Probability Distribution

Let us relax in this Appendix the assumption that the Markov process X has its initial probability restricted to the operational states. We mentioned in Section III that relation (2) leads to $Y^U_{n,k} = \sum_{j=1}^{n-k+1} P_j Y^U_{n-j,k-1}$. This relation can also be derived from renewal arguments. Note first that entry (i,h) of matrix P_j is the probability that the next visited state of U after j transitions will be state h, given that the initial state is $i \in U$ (note that $P_1 = P_U$). Relation (4) is obtained by conditioning on the next visited state of U.

In the same way, (2) leads, for $Y^D_{n,k}$, to $Y^D_{n,k} = \sum_{j=1}^{n-k} P_D^{j-1} P_{DU}\, Y^U_{n-j,k}$, and so, since $Y_{n,k} = \alpha^U Y^U_{n,k} + \alpha^D Y^D_{n,k}$, we get

$$Y_{n,k} = \alpha^U Y^U_{n,k} + \sum_{j=1}^{n-k} \alpha^D P_D^{j-1} P_{DU} Y^U_{n-j,k} = \sum_{j=k}^{n} \alpha^U_{n-j+1} Y^U_{j,k},$$

where the vectors α^U_h are defined by $\alpha^U_1 = \alpha^U$ and for $h \geq 2, \alpha^U_h = \alpha^D P_D^{h-2} P_{DU}$. When the number of operational states is close to the total number of states, the computation of the distribution of $\mathrm{IAV}(t)$ can be performed using the random variable $D(t)$ which denotes the percentage of unoperational time during $(0,t)$, since we have $\mathrm{IAV}(t) + D(t) = 1$, that is, for every $t > 0$, for every $p \in]0,1[$, $P(\mathrm{IAV}(t) > p) = P(D(t) < q) = 1 - P(D(t) > q)$. The matrices involved in the computation of the distribution of $D(t)$ are obtained by a permutation of the role of subsets U and D and of the role of parameters p and q. It follows that for infinite states models, the interval availability distribution can be computed when the number of down states is finite if the corresponding matrices P_j ($P_1 = P_D$ and $P_j = P_{DU} P_U^{j-2} P_{UD}$, $j \geq 2$) can be computed.

APPENDIX B
PROOF OF LEMMA 3.1

Let $E^U_{n,k}$ be the difference $Y^U_{n,k} - Z^U_{n,k}$ and define $Q_J = \sum_{j=1}^{J} P_j$. We then have $E_{n,k} = \alpha^U E^U_{n,k}$. Let us first prove that for $n \geq 1$ and $1 \leq k \leq n$ such that $n - k + 1 > J$, we have $E^U_{n,k} \leq \sum_{l=0}^{k-1} (Q_J)^l E^U_{n-k+1,1}$. Note that $E^U_{n,1} = \sum_{j=J+1}^{n} P_j 1^T$, and so we have $E^U_{n,1} \leq E^U_{n+1,1}$. The proof is made by recurrence over the index k. For $k = 1$, the relation holds since its right hand side is equal to $E^U_{n,1}$. Suppose the relation true until step $k-1$. Since $n - k + 1 > J$, we have, using the recurrence hypothesis,

$$\begin{aligned} E^U_{n,k} &= \sum_{j=1}^{J} P_j E^U_{n-j,k-1} + \sum_{j=J+1}^{n-k+1} P_j Y^U_{n-j,k-1} \\ &\leq \sum_{j=1}^{J} P_j E^U_{n-j,k-1} + \sum_{j=J+1}^{n-k+1} P_j 1^T \\ &= \sum_{j=1}^{J} P_j E^U_{n-j,k-1} + E^U_{n-k+1,1} \\ &\leq \sum_{j=1}^{J} P_j \sum_{l=0}^{k-2} (Q_J)^l E^U_{n-j-k+2,1} + E^U_{n-k+1,1} \\ &\leq \sum_{j=1}^{J} P_j \sum_{l=0}^{k-2} (Q_J)^l E^U_{n-k+1,1} + E^U_{n-k+1,1}. \end{aligned}$$

Since $\sum_{j=1}^{J} P_j = Q_J$, we obtain

$$\begin{aligned} E^U_{n,k} &\leq Q_J \sum_{l=0}^{k-2} (Q_J)^l E^U_{n-k+1,1} + E^U_{n-k+1,1} \\ &= \sum_{l=0}^{k-1} (Q_J)^l E^U_{n-k+1,1}. \end{aligned}$$

Now, since $n - k + 1 > J$, we have $E^U_{n-k+1,1} = \sum_{j=J+1}^{n-k+1} P_j 1^T \leq \varepsilon' 1^T$. So, for every $n \geq 1$ and $1 \leq k \leq n$ such that $n - k + 1 > J$, we get

$$E_{n,k} = \alpha^U E^U_{n,k} \leq \alpha^U \sum_{l=0}^{k-1} (Q_J)^l 1^T \varepsilon' \leq k\varepsilon',$$

which ends the proof.

REFERENCES

[1] A. Goyal and A. N. Tantawi, "A measure of guaranteed availability and its numerical evaluation," *IEEE Trans. Comput.*, vol. C-37, pp. 25–32, 1988.

[2] G. Rubino and B. Sericola, "Interval availability analysis using operational periods," *Perform. Eval.*, vol. 14, pp. 257–272, 1992.

[3] E. de Souza e Silva and H. R. Gail, "Calculating cumulative operational time distributions of repairable computer systems," *IEEE Trans. Comput.*, vol. C-35, pp. 322–332, 1986.

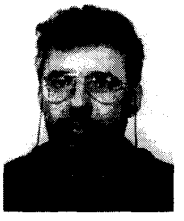
[4] G. Rubino and B. Sericola, "Interval availability distribution computation," in *Proc. IEEE, 23-th Fault-Tolerant Computing Symp.*, Toulouse, France, 1993, pp. 49–55.

[5] S. M. Ross, *Stochastic Processes*. New York: Wiley, 1983.

Gerardo Rubino received the Ph.D. degree in computer science from the University of Rennes I in 1984.

He has been with INRIA (Institut National de Recherche en Informatique et Automatique, a public research French laboratory) since 1985. His research work is centered on the quantitative evaluation of computer and communication systems from the performance and dependability points of view, in particular using Markov models and stochastic graphs.

Dr. Rubino is an Associate Editor of the international journal of Operations Research *Naval Research Logistics*.



Bruno Sericola received the Ph.D. degree in computer science from the University of Rennes I in 1988.

He has been with INRIA (Institut National de Recherche en Informatique et Automatique, a public research French laboratory) since 1989. His main research activity is in computer systems performance evaluation, dependability and performability analysis of fault-tolerant architectures and applied stochastic processes.