Interval-Availability Distribution of 2-State Systems with Exponential Failures and Phase-Type Repairs

B. Sericola
IRISA-INRIA, Rennes

Key Words — Repairable computer system, Cumulative operation time, Interval availability, Markov process, Uniformization technique

Reader Aids — General purpose: Present a new method
Special math needed for explanations: Probability, Markov processes
Special math needed to use results: Same
Results useful to: Reliability analysts, designers of fault-tolerant computers

Summary & Conclusions — Interval availability is a dependability measure defined as the fraction of time during which a system is in operation over a finite observation period. Usually, for computing systems, the models used to evaluate interval availability distribution are Markov models. Numerous papers using these models have been published, and only complex numerical methods have been proposed as solutions to this problem even in simple cases such as the 2-state Markov model. This paper proposes a new way to compute this distribution when the model is a 2-state semi-Markov process in which the holding times have an exponential distribution for the operational state and a phase-type distribution for the non-operational one.

The main contribution of this paper is to define a new algorithm (IAD-SU) to compute the interval availability distribution for a 2-state semi-Markov model in which failures have an exponential distribution and repairs have a phase-type distribution. This measure can be interpreted as the fraction of time during the interval (0,t), spent by a Markov process in its initial state. IAD-SU is derived from the work in [4], which is reviewed in section 2. Section 3 applies IAD-SU to the 2-state semi-Markov model. Section 4 considers particular cases of phase-type repairs such as exponential & Erlang repair. Section 5 gives 2 applications of IAD-SU: 1) A critical system with n components fails if any component fails. 2) The classical M/M/1 queueing system for which we compute the fraction of time in which the server is busy (system workload) during a given time-interval. Application #2 is interesting since the state space of the system is infinite.

Acronyms
IAD interval availability distribution
IAD-SU interval availability distribution — Sericola uniformization (algorithm).

Notation
X continuous-time homogeneous Markov process
X0 state of X at time t
E finite state space of X
α initial probability distribution of X
A infinitesimal generator of X
ν uniformization rate of X
P transition probability matrix of the uniformized Markov chain associated with X
B, L [subset, number] of operational (up) states
Bc subset of non-operational (down) states
αB, αBc subvectors of α associated with partition \{B, Bc\} of E
PB, PBB, PBC, PBc submatrices of P associated with partition \{B, Bc\} of E
P j P j, for j = 1; PBB, PBB, for j > 1.
9(·) 9(·) = 1, 9(False) = 0
(0,t) interval of time
{a,b} set of integers \{a,a+1,...,b\}
O(t) cumulative amount of operational time during (0,t), a r.v.
IAV(t) interval availability over (0,t), a r.v.
11(n) column vector of 1's; dimension is (n+1)-L
N0(n) number of visits to the states of B during the first n transitions of the uniformized Markov chain associated with X; a r.v.

1Appendix A.5 briefly explains uniformization.

2Editors' note: We have assigned this acronym IAD-SU (interval availability distribution — Sericola uniformization) for simple, clear, unique reference to the concept.
$\beta(n) = (n+1) \cdot L \text{ row vector}$

$H(n) = \text{square } [(n+1) \cdot L \times (n+1) \cdot L] \text{ matrix}$

$C, N \text{ integers used in truncation for uniformization, } 0 < C \leq N$

$I = \{l_1, \ldots, l_m\} - L \text{ is usually constrained}$

$\Phi_{m, l}(I) = k! \cdot \left( \prod_{j=1}^{m} P_{l_j}^{n+1} \right) / \left( \prod_{j=1}^{m} l_j! \right)$

$I \text{ is usually constrained}$

$s = \{11, \ldots, 1m\} - I$ is usually constrained

$Pr\{NB(n) > k\} = P(n) \cdot H(n)^k \cdot 1_B(n)$, for $0 \leq k \leq n$.

2. INTERVAL AVAILABILITY DISTRIBUTION

Consider a continuous-time homogeneous Markov process, $X = \{X_t, t \geq 0\}$, over a finite state space $E$.

$O(t) = \left[ \int_{0}^{t} \delta(X_u \in B) \; du \right]$ $IAV(t) = O(t)/t$

The infinitesimal generator $A$ of $X$ verifies $A(i,j) = -\sum_{i \in E} A(i,i)$. The transition probability matrix of the uniformized Markov chain associated with $X$ [6] verifies:

$P = I + A/\nu$

$\nu \geq \max \{-A(i,i), i \in E\}$.

Decompose $P$ & $\alpha$ with respect to $\{B, B^c\}$.

$P = \begin{bmatrix}
    P_{BB} & P_{BB^c} \\
    P_{B^cB} & P_{B^c}
\end{bmatrix}$

$\alpha = (\alpha_B, \alpha_{B^c})$

The main result in [4] is Cdf\{O(t)\}, (0 \leq s < t):

$Pr\{O(t) \leq s\} = 1 - \sum_{n=0}^{+\infty} \binom{n}{s/\nu} \cdot \sum_{k=0}^{n} \binom{k}{s/\nu} \cdot (\beta(n) \cdot H(n)^k \cdot 1_B(n))$

$\beta(n) = (\alpha_B, \alpha_{B^c} \cdot P_{BB^c}, \alpha_{B^c} \cdot P_{BB^c} \cdot P_{BB^c}, \ldots, \alpha_{B^c} \cdot P_{BB^c}^{n-1} \cdot P_{B^c})$, $n > 0$

$\beta(0) = \alpha_B$

$H(0) = 0$

This theoretical result is used for a numerical algorithm.

The SF\{NB(n)\} is [4]:

$Pr\{NB(n) > k\} = \beta(n) \cdot H(n)^k \cdot 1_B(n)$, for $0 \leq k \leq n$.

3. A 2-STATE SEMI-MARKOV MODEL

Assumptions

1. The operational (up) state has exponential holding times.
2. The non-operational (down) state has phase-type holding times.

Nomenclature

Entry: component (of a vector).

This structure is equivalent to the Markov process depicted in section 2 with only 1 operational state, viz, with subset $B$ reduced to 1 state. The formula for $Cdf\{O(t)\}$ can be simplified since the $P_{B^c}$, $j > 1$, are now reduced to real numbers, verifying $0 \leq P_{B^c} \leq 1$.

3.1 Derivation of Simpler Expression for $\beta(n) \cdot H(n)^k \cdot 1_B(n)$

For a fixed $n \geq 0$ and $0 \leq k \leq n$,

$x_{n,k} = H(n)^k \cdot 1_B(n)$. (3-1)

For convenience, the first entry is denoted by $x_{n,k}(0)$ and its last entry by $x_{n,k}(n)$. To simplify the notation, let:

$x_{n,k}(m) = 0$, for $m > n$.

Since $H(0) = 0$ and $H(0)^0 = 1$, we have as first values (in the $x_{n,k}$ sequence):

$x_{0,0}(0) = 1$;

$x_{1,0}(i) = 1, \; i = 0, 1;$
SERICOLA: INTERVAL-AVAILABILITY DISTRIBUTION OF 2-STATE SYSTEMS

331

Then,

\[ x_{n,k+1} = H(n) \cdot x_{n,k} \]

\[ x_{n,k+1}(i) = \sum_{j=1}^{n} P_j \cdot x_{n,k}(j+i), \text{ for } 0 \leq i \leq n. \]  (3-2)

Theorem 3.1 is the main result of this paper.

Theorem 3.1. For all \( n, n \geq 0 \); for all \( k, 0 \leq k \leq n \); for all \( i, 0 \leq i \leq n \):

\[ x_{n,k}(i) = \sum_{l=1}^{n} \Phi_{n,k}(l), \]  (3-3)

\[ \Omega = \{ l_1, l_2, \ldots, l_n \in [0:k] | \theta_{l_n}(l) = k, i + \theta_{l_n}(l) \leq n \}. \]

Theorem 3.1 implies the relation \( (s < t) \):

\[ \Pr\{O(t) \leq s\} = 1 - \sum_{n=0}^{\infty} \text{poin}(n;\nu \cdot t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot \beta \cdot x_{n,k}(i). \]  (3-4)

In practice, the initial system-state is operational \( (\alpha_0 = 1) \); thus \( \beta = (1,0,\ldots,0) \).

Eq (3-4) reduces to:

\[ \Pr\{O(t) \leq s\} = 1 - \sum_{n=0}^{\infty} \text{poin}(n;\nu \cdot t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot \sum_{i=0}^{n} \beta(i) \cdot x_{n,k}(i). \]  (3-5)

The IAD \( (0 \leq u < 1) \) is:

\[ \Pr\{\text{IAV}(t) \leq u\} = 1 - \sum_{n=0}^{\infty} \text{poin}(n;\nu \cdot t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot \sum_{\Omega} \Phi_{n,k}(l); \]  (3-5)

\[ \sum_{\Omega} \Phi_{n,k}(l) = x_{n,k}(0); x_{0,k}(0) = 1. \]

Notation

\[ y_{n,k} = x_{n,k}(0). \]

(This notation simplifies the following presentation.)

To compute the Cdf\{IAV(t)\}, evaluate the \( y_{n,k} \) for \( n = 0, \ldots, N \) and \( k = 0, \ldots, n \) where \( N \), the truncation step of the infinite series, is chosen such that for a given error tolerance \( \epsilon \), the remainder of the series \( e'(N) \) verifies:

\[ e'(N) = \sum_{n=N+1}^{\infty} \text{poin}(n;\nu \cdot t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot y_{n,k} \]

\[ \leq \text{poifc}(N+1; \nu \cdot t). \]

The integer \( N \) is chosen such that poifc\((N+1; \nu \cdot t) \leq \epsilon \). Thus,

\[ \epsilon \leq \Pr\{\text{IAV}(t) \leq u\} - \left[ 1 - \sum_{n=0}^{N} \text{poin}(n;\nu \cdot t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot y_{n,k} \right] \leq 0. \]  (3-6)

Corollary 3.1 provides a recursion to compute the \( y_{n,k} \).

Corollary 3.1

\[ y_{n,0} = 1, \text{ for } n \geq 0; \]

\[ y_{n,k} = \sum_{j=1}^{n-k} P_j \cdot y_{n-j,k-1}, \text{ for } n \geq 1, 1 \leq k \leq n. \]

Figure 1 illustrates the recursion to compute the real numbers \( y_{n,k} \) in corollary 3.1; for instance, \( y_{3,3} = P_1 \cdot y_{2,3} + P_2 \cdot y_{1,3} + P_3 \cdot y_{0,3} \). The \( y_{n,k} \) is represented as the \((n,k)\)-entry of a \((N+1)\)-dimensioned lower triangular matrix. The \( C \) is introduced to perform another truncation over index \( k \); see lemma 3.1.
Lemma 3.1. For every \( n \geq 0 \) and \( 0 \leq k \leq n \),
\[
\begin{align*}
&c. \quad y_{n,k+1} \leq y_{n,k}; \\
&d. \quad y_{n,k} \leq y_{n+1,k};
\end{align*}
\]

Lemma 3.1 helps to show that when \( k \) becomes large enough (\( k > C \)), the values of \( y_{n,k} \) can become very small. Formally
\[
\sum_{n=0}^{N} \text{poin}(n;v,t) \cdot \sum_{k=0}^{n} \text{binm}(k;u,n) \cdot y_{n,k}
= \sum_{k=0}^{C} \sum_{n=0}^{N} \text{poin}(n;v,t) \cdot \text{binm}(k;u,n) \cdot y_{n,k}
= \sum_{n=C+1}^{N} \sum_{k=C+1}^{n} \text{poin}(n;v,t) \cdot \text{binm}(k;u,n) \cdot y_{n,k}
\leq \sum_{n=C+1}^{N} \text{poin}(n;v,t) \cdot y_{n,C+1}
\quad \text{by lemma 3.1-c}
\leq y_{N,C+1}
\quad \text{by lemma 3.1-d}
\]

That is, when computing the \( y_{n,k} \), we try to find a \( C \) such that for a given error tolerance \( \epsilon^* \), we have
\[
y_{N,C+1} \leq \epsilon^*.
\]  

The computation is made column by column as shown in figure 1. For each column \( k \), compute \( y_{n,k} \), using corollary 3.1-b, and test its value with respect to \( \epsilon^* \). If \( y_{N,k} \leq \epsilon^* \), then take \( C = k - 1 \); else compute the other elements of column \( k \), that is \( y_{n-1,k}, y_{n-2,k}, \ldots, y_{1,k}, \) and restart by computing \( y_{n,k+1} \).

If such a \( C \) does not exist, then \( C = N \) and \( \epsilon^*(N,C) = 0 \); the global error is \( \epsilon^* \).

Using this last truncation, compute,
\[
1 - \sum_{k=0}^{C} \sum_{n=k}^{N} \text{poin}(n;v,t) \cdot \text{binm}(k;u,n) \cdot y_{n,k}
\]

If \( \epsilon \) becomes the global error tolerance \( (\epsilon = \epsilon^* + \epsilon^*) \), then from (3-6) - (3-8),
\[
- \epsilon \leq \Pr\{IAV(t) \leq u\} - \left[ 1 - \sum_{k=0}^{C} \sum_{n=k}^{N} \text{poin}(n;v,t) \cdot \text{binm}(k;u,n) \cdot y_{n,k} \right] \leq 0.
\]

IAD-SU for computing \( N \) \& \( C \) is similar to the method in [3]. The main advantage of IAD-SU is that it stores only scalars \( (y_{n,k}) \). The algorithm in [3] requires storage of \( N \) vectors of dimension 'cardinality of the state space of the Markov process', even if the number of operational states is reduced to 1.

IAD-SU does require computing the \( P_{ij} \), \( j = 1, \ldots, N \); however, this can be done recursively in the following way. Recall that:
\[
P_1 = P_B, \quad \text{and } P_j = P_B^{j-2} \cdot P_{BB} \text{ for } j \geq 2.
\]

Define the row vectors:
\[
Q_i = P_B^{i-2} \cdot P_{BB}^i, \quad \text{for } j \geq 2.
\]

so only 1 supplementary vector is needed to store the successive values of \( Q_j \).

4. ERLANG PHASE-TYPE REPAIR

This section considers 2 phase-type repairs for which a simple closed expression for IAD can be obtained using (3-5).

4.1 Irreducible Case

Assumptions

1. The model is a 2-state semi-Markov process.
2. The holding times in state 1 follow an exponential law with rate \( \lambda \).
3. The holding times in state 2 follow an Erlang law with \( r \) stages and parameter \( \mu \).
4. The system starts in state 1 (the unique operational state). It then reaches state 2 after a failure, comes back to state 1 after repair, etc.
5. \( \lambda \leq \mu \).

This semi-Markov process is equivalent to the following Markov process.

\[
1 \quad EXP(\lambda) \quad 2 \quad ERL(r, \mu)
\]

This semi-Markov process is equivalent to the following Markov process.
Assumption

Notation

\[ r \] number of Erlang stages
\[ k^* \] \( \text{gilb}(n-k)/r \)
\[ k^{**} \] \( \text{gilb}(n/(r+1)) \).

Apply (3-5); choose \( r = \mu \) which leads to:

\[ P_j = 1 - \lambda/\mu, \quad P_j = 0, \quad P_{r+1} = \lambda/\mu; \quad P_j = 0, \]

for \( j \geq r+2 \).

This gives, if \( p = P_1 \) and \( q = 1 - p \),

\[ y_{n,k} = \sum_{l=1}^{\infty} \left( \frac{(l)!}{(l-j)!} \right) p^j q^{l-j} \]

\[ \Omega A = \{ i, j \in [0,k] ; i+j = k, i + (r+1) \cdot j \leq n \} \]
or,

\[ y_{n,k} = \sum_{l=1}^{\infty} \binom{n}{j} p^j q^{l-j} \]

\[ \Omega B = \{ j \in [0,k] ; k + r \cdot j \leq n \} \]
or,

\[ y_{n,k} = \binom{\text{min}(k^*, k)}{k} \cdot q \]

For fixed values of \( r \geq 1 \) and \( n \geq 0 \) —

\[ \min(k^*, k) = \begin{cases} k, & \text{for } 0 \leq k \leq k^{**} \\ k^{**}, & \text{for } k^{**} < k \leq n \end{cases} \]

This leads to the closed expression:

\[ \Pr \{ \text{IAV}(t) \leq u \} = 1 - \sum_{n=0}^{\infty} \text{poim}(n; \mu; t) \cdot \binom{n}{k} \cdot q \]

\[ = \sum_{n=1}^{\infty} \text{poim}(n; \mu; t) \cdot \sum_{k=k^{**}+1}^{n} \binom{n}{k} \cdot \binom{k^{**}}{q} \cdot q \]

\[ (4-1) \]

The exponential repair case \( (r=1) \) reduces to:

\[ \Pr \{ \text{IAV}(t) \leq u \} = 1 - \sum_{n=0}^{\infty} \text{poim}(n; \mu; t) \cdot \binom{n}{k} \cdot q \]

\[ = \sum_{n=1}^{\infty} \text{poim}(n; \mu; t) \cdot \sum_{k=k^{**}+1}^{n} \binom{n}{k} \cdot \binom{k^{**}}{q} \cdot q \]

4.2 Absorbing Case

Assumptions

1. The system has 3 states.
2. One state is absorbing, such that it can be completely down either after an operational period (with probability, \( 1 - p_1 \)) or after an unsuccessful repair period (with probability, \( 1 - p_{r+1} \)).
3. \( \lambda \leq \mu \).

We then obtain the following Markov process in which the two up arrows (without destination) are to the absorbing state.

State 1 (initial state) is the unique operational state. Apply (3-5). We choose \( r = \mu \) and have:

\[ P_j = 1 - \lambda/\mu, \quad P_j = 0, \quad P_{r+1} = \lambda \cdot p_1 \]

\[ = \mu \]

\[ P_j = 0 \] for every \( j \geq r+2 \).

Notation

\[ p = P_1 \]
\[ q = P_{r+1} \]

We obtain (4-1) even though \( q \) does not have the same value; here \( p + q \neq 1 \). If \( p_1 = p_{r+1} = 1 \), we obtain (4-1).

4.3 Discussion

In these two examples, the computation can be performed simply by truncating the infinite series as in (3-6).

5. APPLICATIONS

5.1 A Critical System

Assumptions

1. A hardware system has \( n \) components, and is 1-out-of-\( n \)-G (series).
2. Component failures are mutually \( s \)-independent.
3. Repair times are \( s \)-independent of component lives.
4. Maintenance policy is unrestricted, ie, the number of repairmen available is equal to the number of system components.
5. \( \lambda_i \leq \mu_i \).
6. \( \lambda_i = i/1000 \) hours; \( \mu_i = \mu = 1/\text{hour} \).
Notation

- $n$: number of components in the system
- $i$: component index, $i = 1, \ldots, n$
- $x_i$: (component $i$ is up)
- $x$: binary vector
- $A_i$: transition rate matrix for component $i$
- $\lambda_i$, $\mu_i$: [failure, repair] rate of component $i$
- $I_m$: $m$-dimensional identity matrix
- $\Pi, \Sigma$: implies the [product, sum] over $\rho$ from 1 to $n$
- $\mathbb{N}$: set of non-negative integers.

These assumptions lead to a Markov model in which the number of system states be $M = 2^n$. Any system state can be represented by $x$. The only operational state is $(1, \ldots, 1)$. The transition rate matrix $A(n)$ of the system can be easily generated using Kronecker algebra as follows.

$$A(n) = A(n-1) \otimes A_n = \begin{pmatrix}
A(n-1) - \lambda_i I_{n-1} & \lambda_i I_{n-1} \\
\mu_i I_{n-1} & A(n-1) - \mu_i I_{n-1}
\end{pmatrix},$$

for $n \geq 2$.

The uniformization rate is:

$$v = \sum_{\rho} \mu(\rho).$$

All the repair rates being equal does not simplify the IAD computation. Figure 2 shows $Pr\{IAV(t) > 0.9\}$ vs time for several values of $n$.

**Figure 2.** $Pr\{IAV(t) > 0.9\}$ vs Time $[n = 7(1)10]$.
Let, 
\[ B = \{0\}, \quad B^c = \{i \in \mathbb{N} \mid i \geq 1\}, \]
\[ p = \lambda(\lambda + \mu), \quad q = 1 - p. \]
Then,
\[ P_B = q, \quad P_{B^c} = (p, 0, 0, \ldots). \]
\[ P_{B^cB} = (q, 0, 0, \ldots), \] and
the non-zero entries of matrix \( P_B \) are:
\[ P_{Bc(i,i-1)} = q, \quad \text{and} \quad P_{Bc(i,i+1)} = p, \quad \text{for} \quad i \geq 1. \]

The Cdf\{IPS\( (t) \}\} or Sf\{BPS\( (t) \}\} is given by (3-8) with an error less than \( \epsilon \):
\[ \Pr\{BPS(t) > x\} = \Pr\{IPS(t) \leq 1-x\} \]
\[ = 1 - \sum_{k=0}^{C} \sum_{n=k}^{N} \text{poin}(n; (\lambda+\mu) \cdot t) \cdot \text{binm}(k; 1-x, n) \cdot y_{n,k}; \]

the \( N \) & \( C \) are as in section 3. The values of \( y_{n,k} \) are (for \( n \geq 0 \)):
\[ y_{n,k} = \begin{cases} 1, & \text{for} \quad k = 0 \\ \sum_{j=1}^{n-k+1} P_{j} y_{n-j,k-1}, & \text{for} \quad 1 \leq k \leq n. \end{cases} \]

So, we need only the values of \( P_{i} \) to compute Sf\{BPS\( (t) \}\}. These values are given by lemma 5.1.

**Lemma 5.1.** \( P_{1} = q \), and for all \( j \geq 1 \)
\[ P_{2j} = \left( 2^{(j-1)} \right) \cdot (p \cdot q)^j / j, \]
\[ P_{2j+1} = 0. \]

The \( P_{i} \) can be easily computed recursively. Figure 3 shows the probability that the server is occupied for at least 95% of the time, as a function of the \( \lambda \) (0 \( \leq \lambda \leq 2 \)) and \( t \) (0 \( \leq t \leq 100 \)); the service rate \( \mu = 1.0 \).

![Figure 3. Pr\{BPS(t) > 95%\} vs \lambda & t](image-url)

It is well-known that the steady-state workload of the M/M/1 queueing system is:
\[ \text{BPS}(\infty) = \begin{cases} \lambda / \mu, & \text{for} \quad \lambda < \mu \\ 1, & \text{otherwise} \end{cases} \]

Figure 3 shows this limiting behavior. For example, for \( \lambda = 1 \),
\[ \Pr\{\text{workload > 95%}\} = 0.28, \quad \text{for} \quad t \in (0, 100). \]
\[ \Pr\{\text{workload > 95%}\} = 1, \quad \text{for} \quad t \in (0, \infty). \]

**APPENDIX**

A.1 Proof of Theorem 3.1
For \( n = 0 \), the result is trivial.
The proof can be made by induction on integer \( k \), for fixed integer \( n \geq 1 \).
For \( k = 0 \), we obtain \( x_{n,0}(i) = 1 \) for every \( i, \quad 0 \leq i \leq n \) which is in accord with (3-1).
For \( k = 1 \), we obtain for \( 0 \leq i \leq n-1 \),
\[ x_{n,1}(i) = \sum_{j=1}^{n-i} P_{j}, \]
\[ x_{n,1}(n) = 0. \]
Alternatively, (3-2) gives,
\[ x_{n,1}(i) = \sum_{j=1}^{n} P_{j} \cdot x_{n,0}(j+i). \]

Eq (A-a) & (A-b) are the same using the convention that \( x_{n,k}(m) = 0 \) for every \( m > n \).
Let the result be true for integers \( 0, 1, \ldots, k \leq n \); then compute \( x_{n,k+1}(i) \) using (3-2) for every \( i, \quad 0 \leq i \leq n \):
\[ x_{n,k+1}(i) = \sum_{h=1}^{n} P_{h} \cdot x_{n,k}(h+i) \]
\[ = \sum_{h=1}^{n} P_{h} \cdot \sum_{l \in \Omega} \theta_{l,h}(l). \]
\[ \Omega = \{l_{1}, l_{2}, \ldots, l_{n} \in [0, k] \mid \theta_{l,h}(l) = k, \quad i + h + \theta_{l,h}(l) \leq n\}. \]
In the \( \sum_{\Omega} \) sum, change \( l_{h} = l_{h} + 1. \)
The \( l_k \) can start with 0 since all the corresponding terms will be 0; thus,
\[
\Omega_4 = \{ l_1, \ldots, l_{k-1}, l_k, \ldots, l_n \in [0:k] \} | l_k \in [1:k+1],
\]
\[
\theta_{1,n}(l) = k+1, \quad i + \theta_{2,n}(l) \leq n \}.
\]
The r.h.s of the previous equation can be decomposed into two terms: 1) in which \( l_k \in [0:k] \), and 2) in which \( l_k = k+1 \). In term 2, \( l_k = k+1 \) implies that all the other \( l_i \) are 0. Therefore,
\[
x_{n,k+1}(i) = \sum_{h=1}^{n} \sum_{\Omega_5} l_h \Phi_{h,k}(l);
\]
\[
\Omega_5 = \{ l_1, \ldots, l_{k-1}, l_k, \ldots, l_n \in [0:k] \} | l_k \in [0:k+1],
\]
\[
\theta_{1,n}(l) = k+1, \quad i + \theta_{2,n}(l) \leq n \}.
\]
The r.h.s of the previous equation can be decomposed into two terms: 1) in which \( l_k \in [0:k] \), and 2) in which \( l_k = k+1 \). In term 2, \( l_k = k+1 \) implies that all the other \( l_i \) are 0. Therefore,
\[
x_{n,k+1}(i) = \sum_{h=1}^{n} \sum_{\Omega_6} l_h \Phi_{h,k}(l) + \sum_{h=1}^{n} \sum_{\Omega_7} l_h \Phi_{h,k+1}(l);
\]
\[
\Omega_6 = \{ l_1, \ldots, l_{k-1}, l_k, \ldots, l_n \in [0:k] \} | \theta_{1,n}(l) = k+1, \quad i + \theta_{2,n}(l) \leq n \};
\]
\[
\Omega_7 = \{ l_1, \ldots, l_n \in [0:k] \} | i = 0, \quad \text{for } p \neq h, \quad l_h = k+1, \quad i + \theta_{2,n}(l) \leq n \}.
\]
The relation between disjoint sets is:
\[
\{ l_1, \ldots, l_n \in [0:k+1] \} | \theta_{1,n}(l) = k+1, \quad i + \theta_{2,n}(l) \leq n \}
\]
\[
= \{ l_1, \ldots, l_n \in [0:k] \} | \theta_{1,n}(l) = k+1, \quad i + \theta_{2,n}(l) \leq n \} 
\]
\[
\cup (\bigcup_{h=1}^n \{ l_1, \ldots, l_n \in [0:k] \} | l_h = 0 \text{ for } p \neq h, \quad l_h = k+1, \quad i + \theta_{2,n}(l) \leq n \}).
\]
Thus, from \( k+1 - k \), we obtain (3-3). Q.E.D.

A.2 Proof of Corollary 3.1

Relation #a is easily deduced from theorem 3.1.

Relation #b can be proved using (A-1), which can be written for integers \( n \geq 1, \quad 1 < k \leq n \) and \( i = 0 \) as:
\[
y_{n,k} = \sum_{h=1}^{n} P_h \sum_{\Omega_8} \Phi_{h,k-1}(l), \quad (A-2)
\]
\[
\Omega_8 = \{ l_1, l_2, \ldots, l_n \in [0:k-1] | \theta_{1,n}(l) = k-1,
\]
\[
\theta_{2,n}(l) \leq n-h \}.
\]
The conditions \( \{ \theta_{1,n}(l) = k-1 \} \) and \( \{ \theta_{2,n}(l) \leq n-h \} \) imply that \( k-1 \leq n-h \), i.e., \( h \leq n-k+1 \). So, if \( h > n-k+1 \), then the terms in the \( \Omega_8 \) sum of (A-2) are all 0. Therefore,
\[
y_{n,k} = \sum_{h=1}^{n-k+1} P_h \sum_{\Omega_8} \Phi_{h,k-1}(l). \quad (A-3)
\]
In the second sum of (A-3), since the integer \( n-h+1 \) verifies that \( 1 \leq n-h+1 \leq n \), then the condition,
\[
\{ \theta_{2,n}(l) \leq n-h \} \text{ implies that,}
\]
\[
l_{n-k+1} = \ldots = l_n = 0.
\]
Eq (A-3) then becomes,
\[
y_{n,k} = \sum_{h=1}^{n-k+1} P_h \sum_{\Omega_8} \Phi_{h,k-1}(l).
\]
Using theorem 3.1, we obtain,
\[
y_{n,k} = \sum_{h=1}^{n-k+1} P_h y_{n-h,k-1}.
\]
Q.E.D.

A.3 Proof of Lemma 3.1

Since \( y_{n,k} = \Pr\{N_B(n) > k\} \), then lemma 3.1-c is evident.

By definition of \( N_B(n) \), we have \( N_B(n) \leq N_B(n+1) \). It follows that \( N_B(n) > k \) implies \( N_B(n+1) > k \); thus,
\[
y_{n,k} = \Pr\{N_B(n) > k\} \leq \Pr\{N_B(n+1) > k\} = y_{n+1,k}.
\]
This proves lemma 3.1-d. Q.E.D.

A.4 Proof of Lemma 5.1

It is clear that \( P_1 = q \) since \( P_1 = P_B \).

For \( j \geq 2 \),
\[
P_j = P_{BB^0} \cdot P_{B^0}^{j-2} \cdot P_{B^0},
\]
which can be interpreted as,
\[
P_j = p \cdot \Pr\{\text{reaching state 0 after exactly } j-2 \text{ transitions into } B^0 | X_0 = 1\}.
\]
This last probability is clearly 0 when \( j \) is odd. It follows that for all \( j \geq 1 \), \( P_{j+1} = 0 \). Furthermore, for every \( j \geq 1 \),
\[
P_{2j} = p \cdot \Pr\{\text{reaching state 0 after exactly } 2(j-1) \text{ transitions into } B^0 | X_0 = 1\}
which is also

\[ P_{j} = p \cdot \Pr\{'\text{number of customers served in a busy period}' = j\}. \]

It is well known [7] that \( \Pr\{\text{serving } j \text{ customers in a busy period}\} \) is:

\[ \left( \frac{2(j-1)}{j-1} \right) \cdot p^{j-1} \cdot q^{j}. \]

It follows, therefore, that, for every \( j \geq 1 \),

\[ P_j = \left( \frac{2(j-1)}{j-1} \right) \cdot (p \cdot q)^j / j. \]

Q.E.D.

A.5 Explanation of Uniformization

When studying the transient behavior of a Markov process (continuous time Markov chain), the solution to the Chappman forward/backward differential equations follows a matrix exponential, \( \exp(A \cdot t) \), yielding the "transition functions" — analogous to the 1-step transition matrix for discrete-time chains. Generally, computation of the transition functions must be approached numerically, e.g., eigen-analysis to compute \( \exp(A \cdot t) \). However, it is possible to trade a complicated Markov process for one of simpler structure but of the same probability law. This simpler process is such that the subordinate point process (times between jumps) is Poisson (instead of the complicated non-renewal subordinate point process of the original continuous chain — an amazing result) and thus is independent of the embedded (discrete) Markov chain governing state transitions.

Uniformization is the well-known technique for creating this simpler Markov process. An advantage in numerical computations is sometimes gained by appealing to the properties of Poisson processes and the straightforward computations required to study the transient behavior of the (discrete) embedded chain.

REFERENCES


AUTHOR

Dr. Bruno Sericola; IRISA-INRIA; Campus de Beaulieu; 35042 Rennes Cedex; FRANCE.

Bruno Sericola was born in 1959. He received a PhD (1988) in Computer Science from the University of Rennes, France. He then worked on an ESPRIT project at INRIA and in 1989 he obtained a Research position at INRIA. His research activity includes computer-system performance modeling, dependability evaluation of fault-tolerant computing systems, and applied stochastic processes.


IEEE Log Number 92-10653

---

A Decomposition Method for Optimization of Large-System Reliability

There is an error in [1: p 187, (26a)]; that equation should be:

\[ \lambda_c = \frac{2[\pi G_1 \cdot [1 + (G^*/G_1)^2]]}{(26a)} \]

\[ G^* = 1 - \sum_i G_i \cdot \tan(\frac{\sqrt{2} \pi R_i}) \]

REFERENCES


Correction received 1994 January 28

Original IEEE Log Number 92-00843

---

Authorized licensed use limited to: IEEE Xplore. Downloaded on March 20, 2009 at 12:16 from IEEE Xplore. Restrictions apply.