Performability Analysis for Degradable Computer Systems

Hédi Nabli, Bruno Sericola

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Performability Analysis for Degradable Computer Systems

Hédi Nabli and Bruno Sericola

N° 2602

Juillet 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
RENNES

Performability Analysis for Degradable Computer Systems

Hédi Nabli and Bruno Sericola

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet MODEL

Rapport de recherche n ° 2602 — Juillet 1995 — 19 pages

Abstract: Degradable performance of fault-tolerant computer systems has given rise to considerable interest in mathematical models for combined evaluation of performance and reliability. Most of these models are based upon Markov processes. Several methods have been proposed for the computation of the probability distribution of performability upon an interval of time $[0, t]$. In this paper, we present a new algorithm based on the uniformization technique to compute this distribution for block degradable models. The main advantage of this method is its low polynomial computational complexity and its numerical stability. Moreover, it only deals with a non increasing sequence of positive numbers bounded by 1. This important property allows us to determine new truncation steps which improve the execution time of the algorithm. We apply this method to a degradable computer system.

Key-words: Block degradable models, degradable systems, Markovian models, performability.

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Analyse de Performabilité pour des systèmes informatiques dégradables

Résumé : Les systèmes multiprocesseurs tolérant les pannes ont donné un champ d'intérêt considérable dans les modèles mathématiques pour l'évaluation combinée des aspects de performance et de fiabilité. La plupart de ces modèles sont basés sur les processus de Markov. Plusieurs méthodes ont été développées pour le calcul de la distribution de la performabilité sur un intervalle de temps donné $[0, t]$. Dans cet article, nous présentons une nouvelle méthode basée sur l'uniformisation et applicable aux modèles dégradables par bloc. L'avantage majeur de cette méthode est sa faible complexité polynomiale et sa stabilité numérique. De plus, elle ne traite que des nombres positifs majorés par 1. Cette propriété nous permet de determiner un nouveau rang de troncation permettant ainsi d'améliorer le temps d'éxécution de l'algorithme. Nous appliquons cette méthode à un système informatique dégradable.

Mots-clé : Modèles dégradables par bloc, modèles markoviens, perfomabilité, systèmes dégradables.

1 Introduction

Reliability and availability measures are of considerable interest for the evaluation of dependability of computer systems, and more specifically, of fault-tolerant computer systems. Thanks to redundancies in both software and hardware, the system is able to reconfigure itself when failures occur. Two types of fault-tolerant computer systems are usually identified: degradable (or non repairable) systems and repairable systems. From a modeling point of view, degradable systems and repairable systems are, respectively, represented by means of acyclic and cyclic processes.

As recognized in a number of recent studies, the evaluation of fault-tolerant computer systems must simultaneously deal with aspects of both performance and reliability. As part of these studies, Meyer [1] introduces an unified measure called performability which combines the two aspects of performance and reliability. Performability is defined as the accumulated reward over a period of time $[0, t]$. Formally, the fault-repair behavior of the system is assumed to be modeled by a homogeneous Markov process. Its state space is divided into disjoint subsets, which represent the different configurations of the system. A performance level (or reward rate) is associated with each of these configurations. This reward rate quantifies the ability of the system to perform correctly in the corresponding configuration. As a consequence, performability corresponds to the accumulated reward over the mission time.

The distribution of this random variable has been studied in previous papers. In [1], Meyer obtains a closed form expression for the distribution of the performability for a degradable computer system with N processors and a buffer with finite capacity L. Furchtgott and Meyer [2, 3] define $i-$resolvable vectors to characterize the trajectories of an acyclic semi-Markovian process corresponding to a certain performance level. By enumerating all the possible trajectories of the system, they derive an integral expression for performability which they solve numerically. However, the complexity of such an algorithm is exponential in the number of states of the process. Beaudry [4] gives a method for the computation of performability in an acyclic Markovian process until absorption. Ciardo et al. [5] generalize Beaudry's approach to a semi-Markov reward process and remove the restriction requiring only the absorbing states to be associated with a zero reward rate. Goyal and Tantawi [6] derive a closed form expression (precisely a finite sum of exponential functions) for the performability of degradable heterogeneous systems. They also give an algorithm with a polynomial complexity $O(dM^3)$ in the number M of states and in the number of components d in the system.

In this paper we propose a new algorithm to compute the distribution of the accumulated reward over a finite mission time for block acyclic Markov models (*i.e.* the infinitesimal generator is block upper triangular). Our technique is based on the uniformization method and is applicable to a wider class of models than previous approaches. The main contribution of this algorithm is its numerical stablility : it only deals with a non-increasing sequence of positive numbers bounded by 1. Moreover, the computational complexity is improved by using truncation steps and the accuracy of the result can be chosen in advance.

The remainder of this paper is organized as follows. In the next section, we introduce the mathematical model of the class of degradable systems and we give some interesting results for the distribution of the performability. In section 3, we present a simple algorithm towards performability evaluation. We also discuss the computational complexity of the proposed technique. A numerical example of degradable multiprocessor system is presented and solved for a given performability measure in section 4. The main points are summarized in the concluding section.

2 Mathematical model and results

Degradable computer systems operate at various levels of performance: when a component fails, the system reconfigures itself and carries on functioning albeit with degraded performance. Because of changes in its structure, due to failures, the system has different configurations in a finite state space $E = \{1, 2, \ldots, M\}$. A reward rate $\rho(i)$ which is independent of the time is associated with each state $i \in E$. The accumulated reward random variable over a finite period of time $[0, t]$ is of interest.

Let therefore $X = (X_s)_{s \geq 0}$ be a homogeneous Markov process over the state space E. Since two different states may have the same reward rate, we denote by $r_m > r_{m-1} > \ldots > r_0$ the $m+1$ different reward rates ($m < M$), and by B_i the set of states having r_i as reward rate, for $i \in \{0, , \ldots, m\}$. It is clear that the subsets $B_m, \ldots, B_0$ are disjoint and their union gives in the state space E. The process X is entirely determined by its infinitesimal generator $A = (a_{ij})_{i,j \in E}$ and its initial probability distribution $\alpha = (\alpha_i)_{i \in E}$. We assume α to be a row vector of the form $\alpha = (\alpha_{B_m}, 0, \ldots, 0)$. This means that the system starts in subset B_m with probability 1. Since the system is degradable, we have

$$a_{ij} = 0 \text{ if } i \in B_l,\ j \in B_k \text{ and } l < k \tag{1}$$

Note that in the same subset B_l, we can have transitions between two different states i and j even if $i < j$, which means that the process restricted on each block B_l is cyclic. Such models are called block degradable.

It is well-known [7] that if the process X is uniformized with an intensity parameter $\lambda \geq \max_{i \in E}(-a_{ii})$, then $P = \dfrac{A}{\lambda} + I$ is the transition probability matrix associated to the uniformized chain where I is the $M \times M$ identity matrix. Using the decomposition of E with respect to the partition $\{B_m, \ldots, B_0\}$ and the equation (1), the matrix P can be written as follows :

$$P = \begin{pmatrix} P_{B_m B_m} & P_{B_m B_{m-1}} & \cdots & P_{B_m B_0} \\ 0_{B_{m-1} B_m} & P_{B_{m-1} B_{m-1}} & \cdots & P_{B_{m-1} B_0} \\ \vdots & \vdots & & \vdots \\ 0_{B_0 B_m} & 0_{B_0 B_{m-1}} & \cdots & P_{B_0 B_0} \end{pmatrix}$$

where the sub-matrix $P_{B_i B_j}$ (resp. $0_{B_i B_j}$) contains the transition probabilities from states of B_i to states of B_j (resp. is a zero matrix). With the above notation, the accumulated reward over the mission time $[0, t]$ is defined by

$$Y_t = \int_0^t \rho(X_s) ds = \sum_{i=0}^{m} r_i \int_0^t \mathbb{1}_{\{X_s \in B_i\}} ds,$$

where $\mathbb{1}_c = 1$ if condition c is true and 0 otherwise.

The random variable Y_t takes its values in the interval $[r_0 t, r_m t]$ and we wish to derive $\mathbb{P}\{Y_t > s\}$. The reward rates r_i are arbitrary real numbers, but we can assume $r_0 = 0$ without loss of generality. This is obtained by replacing r_i by $r_i - r_0$ and s by $s - r_0 t$. Therefore, we take $r_0 = 0$.

In [8], the distribution of performability over $[0, t]$ for repairable computer systems, *i.e.* for cyclic models, is derived by a new method. The approach is essentially based up on the following theorem.

Theorem 2.1 *For all $j = 1, \ldots, m$ and $s \in [r_{j-1}t, r_j t[$, we have*

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_j^k (1 - s_j)^{n-k} b^{(j)}(n,k)$$

where $s_j = \frac{s - r_{j-1}t}{(r_j - r_{j-1})t}$ and $b^{(j)}(n,k) = \alpha_{B_m} b^{(j)}_{B_m}(n,k)$. The values of the column vectors $b^{(j)}_{B_l}(n,k)$ are positive bounded by 1 and are given by the following set of recursive expressions :

for all $n \geq 0$, and $j = 1, \ldots, m$

for $j \leq l \leq m$ **and** $1 \leq k \leq n$

$$\begin{aligned} b^{(j)}_{B_l}(n,0) &= \begin{cases} \mathbf{1}_{B_l} & \text{if } j = 1 \\ b^{(j-1)}_{B_l}(n,n) & \text{otherwise} \end{cases} \\ b^{(j)}_{B_l}(n,k) &= \frac{r_l - r_j}{r_l - r_{j-1}} b^{(j)}_{B_l}(n,k-1) + \frac{r_j - r_{j-1}}{r_l - r_{j-1}} \sum_{i=0}^{m} P_{B_l B_i} b^{(j)}_{B_i}(n-1,k-1) \end{aligned}$$

for $0 \leq l \leq j-1$ **and** $0 \leq k \leq n-1$

$$\begin{aligned} b^{(j)}_{B_l}(n,k) &= \frac{r_{j-1} - r_l}{r_j - r_l} b^{(j)}_{B_l}(n,k+1) + \frac{r_j - r_{j-1}}{r_j - r_l} \sum_{i=0}^{m} P_{B_l B_i} b^{(j)}_{B_i}(n-1,k) \\ b^{(j)}_{B_l}(n,n) &= \begin{cases} 0_{B_l} & \text{if } j = m \\ b^{(j+1)}_{B_l}(n,0) & \text{otherwise} \end{cases} \end{aligned}$$

$\mathbf{1}_{B_l}$ and 0_{B_l} are column vectors of dimension $|B_l|$ and their values are respectively equal 1 and 0.

Proof. (See [8]) □

When the system is block degradable, new results for the sequence $b^{(j)}_{B_l}(n,k)$ can be obtained. For such models, the recursive expressions of the vectors $b^{(j)}_{B_l}(n,k)$ reduce to the following:

Lemma 2.2 *For all $n \geq 0$ and $j = 1, \ldots, m$, we have*

for $0 \leq l \leq j-1$ **and** $0 \leq k \leq n$

$$b^{(j)}_{B_l}(n,k) = 0_{B_l}$$

for $j \leq l \leq m$ **and** $1 \leq k \leq n$

$$b^{(j)}_{B_l}(n,0) = \begin{cases} \mathbf{1}_{B_l} & \text{if } j = 1 \\ b^{(j-1)}_{B_l}(n,n) & \text{otherwise.} \end{cases}$$

$$b^{(j)}_{B_l}(n,k) = \frac{r_l - r_j}{r_l - r_{j-1}} b^{(j)}_{B_l}(n,k-1) + \frac{r_j - r_{j-1}}{r_l - r_{j-1}} \sum_{i=j}^{l} P_{B_l B_i} b^{(j)}_{B_i}(n-1,k-1).$$

Furthermore, for $j = 1$, the coefficients $b^{(1)}_{B_l}(n,k)$ are independent of index n.

Proof. (See Appendix A). □

The previous lemma shows that $b^{(1)}_{B_l}(n,k)$ can be simplified in $b^{(1)}_{B_l}(k)$. The case $j=1$ means that s belongs to $[0, r_1 t[$. If we apply theorem 2.1 to this case, we can simplify the distribution of the performability. This is the object of the following corollary.

Corollary 2.3 *For all $s \in [0, r_1 t[$, we have*

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k),$$

where $b^{(1)}(k) = \alpha_{B_m} b^{(1)}_{B_m}(k)$.

Proof. (see Appendix B) □

In practise, the value of s is very close to $r_m t$ since it is generally required that the random variable Y_t should be close to its maximum value $r_m t$ with a probability close to 1. The following corollary gives the distribution of the performability when $s \in [r_{m-1} t, r_m t[$.

Corollary 2.4 *For all $s \in [r_{m-1} t, r_m t[$, we have*

$$\mathbb{P}\{Y_t > s\} \;=\; \alpha_{B_m} e^{A_{B_m} s_m t} \left[\sum_{n=0}^{\infty} e^{-\lambda t (1-s_m)} \frac{(\lambda t (1-s_m))^n}{n!} b^{(m-1)}_{B_m}(n,n) \right].$$

Proof. (See Appendix C). □

Our analysis will consist in obtaining new truncation steps by using properties of monotony of the sequence $b^{(j)}_{B_l}(n,k)$. These properties leads us to improve the numerical complexity of our algorithm.

The next theorem states a relation between $b^{(j)}_{B_l}(n,k)$ and $b^{(j)}_{B_l}(n,k-1)$ which correspond to two consecutive cells on the same line of the array associated to j (see fig. 1).

Theorem 2.5 *For all $1 \le j \le m$, $n \ge 1$, $1 \le k \le n$, and $j \le l \le m$, we have :*

$$b^{(j)}_{B_l}(n,k) \le b^{(j)}_{B_l}(n,k-1)$$

Proof. (See Appendix D). □

Theorem 2.5 expresses the decreasing of cells of the same line in each triangle containing $b^{(j)}_{B_l}(n,.)$ elements (see fig. 1). Moreover, theorem 2.6 states another relation for two consecutive terms in each column of a triangle.

Theorem 2.6 *For all $1 \le j \le m$, $k \ge 1$, $n \ge k+1$ and $j \le l \le m$, we have*

$$b^{(j)}_{B_l}(n,k) \le b^{(j)}_{B_l}(n-1,k).$$

Proof. (See Appendix D). □

Putting together these results, we obtain the following inequalities

$$b^{(j)}_{B_l}(n,k) \le b^{(j)}_{B_l}(n,k-1) \le b^{(j)}_{B_l}(n-1,k-1) \tag{2}$$

In the next section, we describe the computational aspect of the vectors $b^{(j)}_{B_l}(n,k)$. Subsequently, we give a numerical method to compute the distribution of the performability. With different results obtained for different values of s, we, successively, deal with the following cases $s \in [0, r_1 t[$, $s \in [r_{j_0-1} t, r_{j_0} t[$, $1 < j_0 < m$ and $s \in [r_{m-1} t, r_m t[$. In each case, we also analyze the numerical complexity.

3 Computational and algorithmical aspects

3.1 Computation of $b^{(j)}_{B_l}(n,k)$

Since coefficients $b^{(j)}_{B_l}(n,k)$ are equal to 0_{B_l} for $l < j$, we define for every $j = 1,\ldots,m$, the subset U_j of the state space E as $U_j = B_m \cup B_{m-1} \cup \ldots \cup B_j$. Note that the sets U_j decrease with respect to j in the sense of inclusion (*i.e.* $U_j \subset U_{j-1}$). We also define the following column vector of length $|U_j|$

$$b_{U_j}(n,k) = \begin{pmatrix} b^{(j)}_{B_m}(n,k) \\ \vdots \\ b^{(j)}_{B_j}(n,k) \end{pmatrix}$$

With this notation, fig. 1 illustrates the sequence of computations (drawn only for $m = 3$ and $n = 0,1,2$) needed to evaluate the $b^{(j)}_{B_l}(n,k)$'s for $l \geq j$. The computation of each cell (n,k), $k \geq 1$ depends on the two vectors associated to cells $(n,k-1)$ and $(n-1,k-1)$. This dependence is pictured in fig. 1 with arrows. Note that the symbol "copy" means that each diagonal cell of each triangle must be reported in the corresponding cell of the first column in the next triangle (*i.e.* $b^{(j-1)}_{B_l}(n,n) = b^{(j)}_{B_l}(n,0)$ for all $l = j,\ldots,m$). The use of relations in lemma 2.2 leads us to perform the evaluation of the $b_{U_j}(n,k)$'s in a line by line manner as shown in fig. 1.

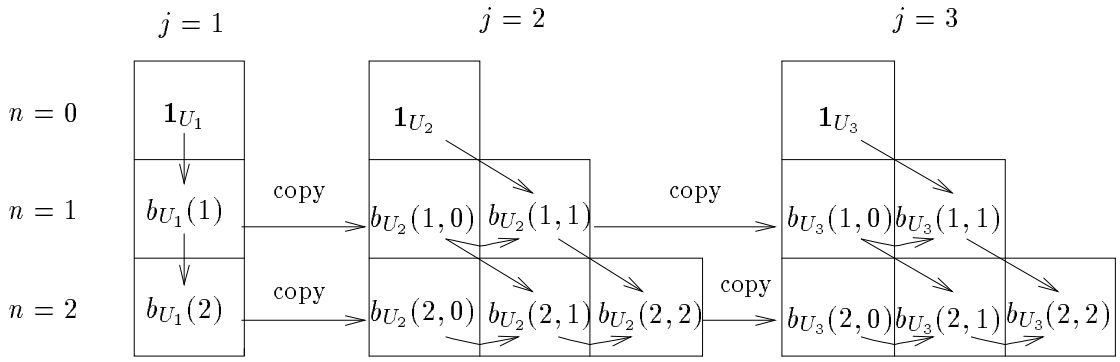

Figure 1: Computation of coefficients $b_{U_j}(n,k)$ for $m = 3$ and $n = 0,1,2$.

Property (2) allows us to define a new truncation step for computing the infinite sum. Formally, for a tolerance error ε specified by the user and a given positive number x, we define the integer $N(x) = min\{n \in \mathbb{N} \,/\, \sum_{i=0}^{n} e^{-x}\frac{x^i}{i!} > 1-\varepsilon\}$, ($N(x)$ is the truncation step of a Poisson series). We denote by d the connectivity degree of matrix P which is defined by $d = \max_{i \in E}\{NZ(i)\}$, where $NZ(i)$ is the number of nonzero entries in row i of P.

3.2 Case $s \in [0, r_1 t[$

According to corollary 2.3, the distribution of the performability Y_t is given by the following expression:

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{\infty} e^{-\lambda t s_1}\frac{(\lambda t s_1)^k}{k!} b^{(1)}(k).$$

This distribution can also be written as

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{N(\lambda t s_1)} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) + e(N(\lambda t s_1))$$

where $e(N(\lambda t s_1))$ satisfies

$$\begin{aligned} e(N(\lambda t s_1)) &= \sum_{k=N(\lambda t s_1)+1}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) \\ &\leq \sum_{k=N(\lambda t s_1)+1}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} \text{ , since } b^{(1)}(k) \leq 1 \\ &= 1 - \sum_{k=0}^{N(\lambda t s_1)} \frac{(\lambda t s_1)^k}{k!} \\ &\leq \varepsilon. \end{aligned}$$

Moreover, if we take into account the inequality $b_{U_1}(k) \leq b_{U_1}(k-1)$ given by theorem 2.5, it is interesting to use another truncation step n_1 defined by

$$n_1 = min\left(N(\lambda t s_1), min\{k \in \mathbb{N} \ / \ b^{(1)}(k+1) \leq \varepsilon\}\right).$$

By considering this new truncation, we obtain

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{n_1} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) + e(n_1).$$

The error introduced in the evaluation of the distribution of Y_t remains smaller than ε. In fact,

$$\begin{aligned} e(n_1) &= \sum_{k=n_1+1}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) \\ &\leq \left[\sum_{k=n_1+1}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!}\right] \varepsilon \text{ , since } b^{(1)}(k) \leq \varepsilon \text{ for } k \geq n_1 + 1 \\ &\leq \varepsilon. \end{aligned}$$

The computation of the distribution of Y_t mainly involves computing the coefficients $b^{(1)}(k)$. Moreover, the complexity in computing of a vector $b_{U_1}(k)$, which is one cell pictured in fig. 2, is $O(d(M-1))$. Therefore, as the total number of required vectors $b_{U_1}(k)$ is n_1+1, the complexing in evaluating $F_{B_m}(s,t)$ is $O(d(n_1+1)(M-1))$. On the other hand, the total computational effort required with the method of [8] is $O(dM[C(N-C) + mC^2/2])$, where C and N are respectively equal to $N(\lambda t s_1)$ and $N(\lambda t)$. Therefore, the algorithm in this section compares favorably with the method of [8].

3.3 Case $s \in [r_{j_0-1}t, r_{j_0}t[$, $2 \leq j_0 \leq m-1$

According to theorem 2.1, the distribution of Y_t is given by

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_{j_0}^k (1 - s_{j_0})^{n-k} b^{(j_0)}(n,k).$$

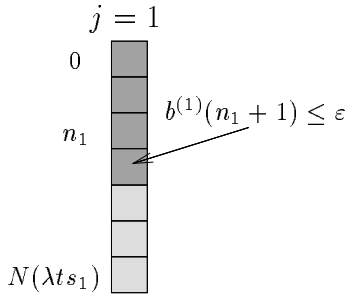

Figure 2: In dark, the computed vectors $b_{U_1}(.)$. In light, cells such that $b^{(1)}(.) \leq \varepsilon$

To compute the infinite sum, we define a truncation step n_{j_0} as follows

$$n_{j_0} = min\left(N(\lambda t), min\{n \in \mathbb{N} \ / \ b^{(j_0)}(n+1, 0) \leq \varepsilon\}\right).$$

Since $b^{(j_0)}_{B_m}(n, 0) = b^{(j_0-1)}_{B_m}(n, n)$, we have

$$n_{j_0} = min\left(N(\lambda t), min\{n \in \mathbb{N} \ / \ b^{(j_0-1)}(n+1, n+1) \leq \varepsilon\}\right).$$

Using inequalities (2), it is easy to establish that all terms $b^{(j_0)}(n, k)$ are smaller than ε, for $n \geq n_{j_0} + 1$ and $0 \leq k \leq n$ (see fig. 3). It follows that

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{n_{j_0}} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_{j_0}^k (1 - s_{j_0})^{n-k} b^{(j_0)}(n, k) + e(n_{j_0})$$

where $e(n_{j_0})$ satisfies

$$\begin{aligned}
e(n_{j_0}) &= \sum_{n=n_{j_0}+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_{j_0}^k (1 - s_{j_0})^{n-k} b^{(j_0)}(n, k) \\
&\leq \left[\sum_{n=n_{j_0}+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_{j_0}^k (1 - s_{j_0})^{n-k} \right] \varepsilon \\
&= \left[\sum_{n=n_{j_0}+1}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \right] \varepsilon \\
&\leq \varepsilon.
\end{aligned}$$

Note that in the case where all coefficients $b^{(j_0-1)}(n+1, n+1)$ are strictly greater than ε for all $n \leq N(\lambda t)$, n_{j_0} is equal to $N(\lambda t)$. In this case, $e(n_{j_0}) = e(N(\lambda t))$ remains smaller than ε.

Since the cardinality $|U_j|$ is less than $M - j$, the computation of each vector $b_{U_j}(n, k)$ requires at most $O(d(M - j))$ (see equations $b_{U_j}(n, k)$ in lemma 2.2). According to fig. 3, the number of cells that have to be computed in the triangle associated to index j is equal to

$$\begin{cases}
n_{j_0} + 2 & \text{if } j = 1 \\
\frac{(n_{j_0}+2)(n_{j_0}+3)}{2} & \text{if } 1 < j < j_0 \\
\frac{(n_{j_0}+1)(n_{j_0}+2)}{2} & \text{if } j = j_0.
\end{cases}$$

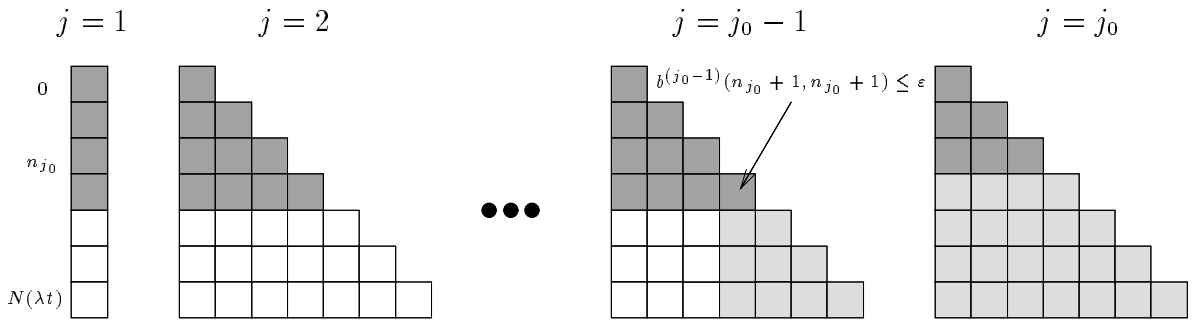

Figure 3: In dark, the computed vectors. In light, cells such that $b^{(j)}(.,.) \leq \varepsilon$

Since we have $\sum_{j=2}^{j_0}(M-j) = (j_0-1)(M-\frac{j_0+2}{2})$, and can neglect n_{j_0} with respect to $n_{j_0}^2$, we obtain a complexity for the algorithm of

$$O\left(d(M-\frac{j_0+2}{2})(j_0-1)\frac{n_{j_0}^2}{2}\right).$$

As opposed to the method relative to repairable systems, we observe that the numerical complexity of this method depends on the index j_0 and therefore on the value of s. Another improvement came from the new truncation step n_{j_0} which is less than $N(\lambda t)$.

3.4 Case $s \in [r_{m-1}t, r_m t[$

According to theorem 2.1, we have

$$\mathbb{P}\{Y_t > s\} = \alpha_{B_m} e^{A_{B_m} s_m t} \left[\sum_{n\geq 0} e^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^n}{n!} b_{B_m}^{(m-1)}(n,n)\right].$$

Similarly to the previous section, we define the truncation step n_m as follows

$$n_m = min\left(N(\lambda t(1-s_m)), min\{n \in \mathbb{N} \,/\, b_{B_m}^{(m-1)}(n+1,n+1) \leq \varepsilon \mathbf{1}_{B_m}\}\right).$$

Using inequalities (2) again, it is easy to establish that all terms $b_{B_m}^{(m-1)}(n,n)$ are smaller than ε when $n \geq n_m + 1$. It follows that

$$\mathbb{P}\{Y_t > s\} = \alpha_{B_m} e^{A_{B_m} s_m t} \left[\sum_{n=0}^{n_m} e^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^n}{n!} b_{B_m}^{(m-1)}(n,n)\right] + e(n_m)$$

where $e(n_m)$ satisfies

$$\begin{aligned}
e(n_m) &= \alpha_{B_m} e^{A_{B_m} t s_m} \left[\sum_{n\geq n_m+1} e^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^n}{n!} b_{B_m}^{(m-1)}(n,n)\right] \\
&\leq \alpha_{B_m} e^{A_{B_m} t s_m} \left[\sum_{n\geq n_m+1} e^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^n}{n!}\right] \varepsilon \mathbf{1}_{B_m} \\
&\leq \alpha_{B_m} e^{A_{B_m} t s_m} \varepsilon \mathbf{1}_{B_m} \\
&\leq \varepsilon \text{ , since } e^{A_{B_m} t s_m} \text{ is a substochastic matrix.}
\end{aligned}$$

The global computational scheme using the truncation step n_m is shown in fig. 4, where only the dark part has to be computed. The number of dark cells is

$$n_m + (m-2)\frac{n_m(n_m+1)}{2}.$$

Similarly to the previous case, we prove that computing the sum $\sum_{n=0}^{n_m} e^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^n}{n!} b_{B_m}^{(m-1)}(n,n)$ has complexity

$$O\left(d(M-\frac{m+1}{2})(m-2)\frac{n_m^2}{2}\right).$$

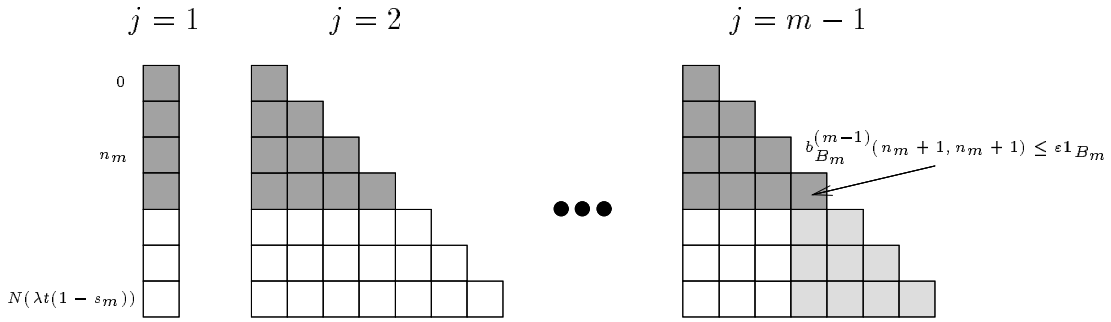

Figure 4: In dark, the computed vectors. In clear, cells such that $b_{B_m}^{(m-1)}(.,.) \leq \varepsilon \mathbf{1}_{B_m}$

Next, to compute the matrix $e^{A_{B_m} s_m t}$, we use the uniformization technique. Therefore, we first denote respectively by λ_m and d_m the uniformization rate (*i.e.* $\lambda_m = \max_{i \in B_m}(-a_{ii})$) and the degree of connectivity of matrix A_{B_m}. It is easy to verify that $\lambda_m \leq \lambda$ and $d_m \leq d$. Then, the computational effort required to multiply the matrix A_{B_m} by a given column vector has complexity $O(d_m|B_m|)$. Moreover, it is well-known that the truncation step in evaluating $e^{A_{B_m} s_m t}$ is equal to $N(\lambda_m s_m t)$. Thus, computing the product of matrix $e^{A_{B_m} s_m t}$ by a column vector of dimension $|B_m|$ has a total cost of $O(d_m|B_m|N(\lambda_m s_m t))$.

We can therefore conclude in this case that the numerical complexity to compute the distribution of the performability is

$$O\left(d(M-\frac{m+1}{2})(m-2)\frac{n_m^2}{2} + d_m|B_m|N(\lambda_m s_m t)\right).$$

Knowing that the evaluation of matrix $e^{A_{B_m} s_m t}$ introduces an error less than ε, it is easy to verify that the total error in computing $\mathbb{P}\{Y_t > s\}$, $s \geq r_{m-1}t$, is less than 2ε.

4 A numerical example

We consider in this section a mutiprocessor system described and modeled in [6]. The system contains N processors each with its local memory, M shared memory modules, and B busses to communicate among the processors and the memory modules. We study the multiprocessor

system when $N = M = B = 2$. The processors, shared memory modules, and busses are subject to random failures independently of each other. When one of these components fail, the number of available components decreases and consequently the processing power of the system decreases. The life-time of each processor (resp. bus) is assumed to be exponentially distributed with rate λ_p (resp. λ_b). The life-time of each shared memory module is assumed to have a probability distribution of phase type with 2 states, with initial probability distribution $\beta = (1, 0, 0)$ and infinitesimal generator T given by

$$T = \begin{pmatrix} -(\mu_{1,2} + \mu_{1,0}) & \mu_{1,2} & \mu_{1,0} \\ \mu_{2,1} & -(\mu_{2,1} + \mu_{2,0}) & \mu_{2,0} \\ 0 & 0 & 0 \end{pmatrix}.$$

The performance measure used here is the processing power of the system, that is the processor utilization rate multiplied by the number of available processors. The values of the processing power have been taken from [6] and are the following. We define $pp(n, m, b)$ as the processing power of the system when there are n processors, m shared memory modules, and b busses available. The values of $pp(n, m, b)$ have been taken from [6] and are the following:

$$pp(1, m, b) = 0.8 \text{ for } 1 \leq m, b \leq 2$$

$$pp(2, m, 1) = 1.56 \text{ for } 1 \leq m \leq 2$$

$$pp(2, 1, 2) = 1.56 \text{ and } pp(2, 2, 2) = 1.58.$$

When one of the three n, m, and b is 0 we set $pp(n, m, b) = 0$. The values of the failure rates λ_p and λ_b are also chosen as in [6], that is $\lambda_p = 6 \times 10^{-5}$ per hour and $\lambda_b = 4 \times 10^{-5}$ per hour. The parameters of the phase type distribution of the life time of each memory modules are given by:

$$\mu_{1,2} = 14 \times 10^{-5} \ , \ \mu_{1,0} = 6 \times 10^{-5} \ , \ \mu_{2,1} = (4/7) \times 10^{-5} \ , \ \mu_{2,0} = (24/7) \times 10^{-5}.$$

With these values, we get a mean life time for each shared memory module equal to 0.25×10^5 hours which is the same expected value as in [6].

In order to get a Markov process to describe the behaviour of the system, we take as state description the vector $(n, (ph_1, ph_2), b)$ where $1 \leq n, b \leq 2$ and ph_i, $i = 1, 2$, denotes the number of memory modules in phase i. The value $m = ph_1 + ph_2$ gives the number of available memory modules ($0 \leq ph_1, ph_2 \leq 2$). All the states corresponding to the down state of the system, that is when $n = 0$ or $m = 0$ or $b = 0$, are lumped into only one absorbing state having a processing power equal to 0. Thus, we get a Markov process with 21 states, one of them being an absorbing state. There are 4 distinct reward rates: $r_3 = 1.58$, $r_2 = 1.56$, $r_1 = 0.8$ and $r_0 = 0$.

Fig. 5 shows the probability to get a cumulative reward rate over the specified period $t = 10^5$ hours greater than the corresponding value on the s axis. It is interesting to note that for $s = 5 \times 10^4$, the truncation steps n_1 is equal to 59 while $N(\lambda t s_1)$ is equal to 96. Fig. 6 shows the probability that the processing power during $(0, t)$ averaged over time is greater than 99% of the maximum processing power (that is 1.58) of the system.

5 Conclusion

The current method to evaluate the performability distribution for degradable computer systems, based on the uniformization technique, leads to a new algorithm with a low polynomial

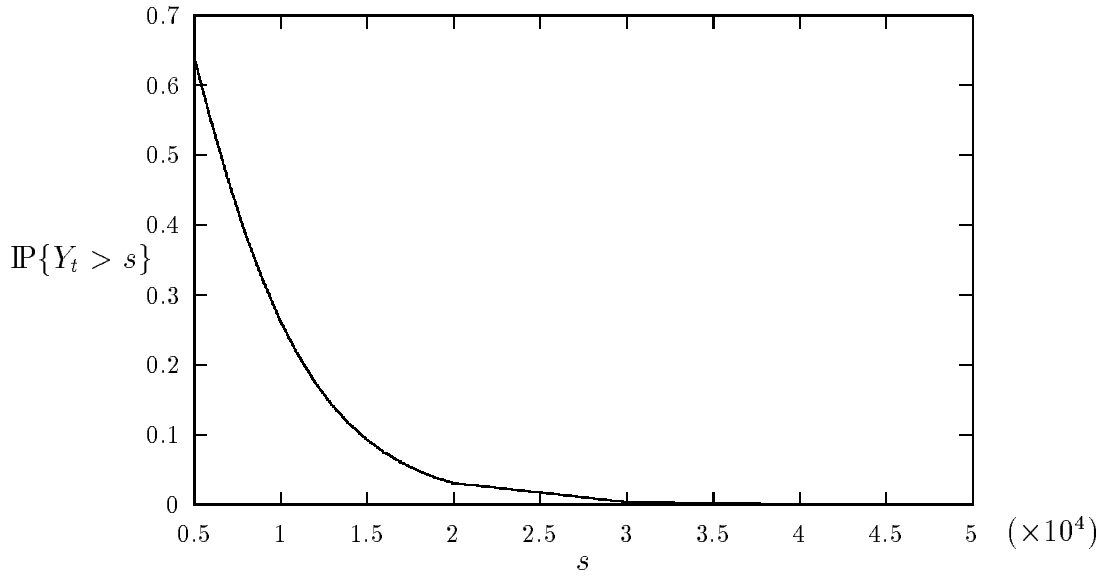

Figure 5: Distribution of Y_t for $t = 10^5$ hours

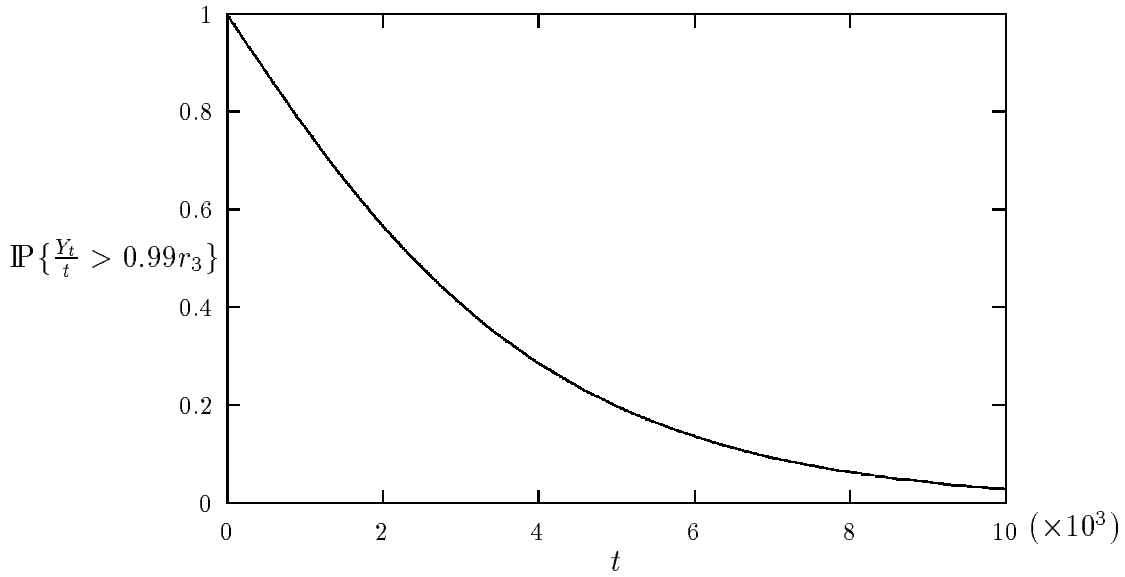

Figure 6: Distribution of $\frac{Y_t}{t}$

computational complexity. Its main advantage involves a complexity at most quadratic in a truncation step which is smaller than the Poisson's one usually used in all methods based on uniformization. Another advantage of this algorithm is an improvement in its stability because only positive numbers bounded by 1 are involved.

Appendix A

Proof of lemma 2.2: Using the recurrence relationship given by the theorem 2.1, we can prove,by recurrence on n and k, that $b_{B_l}^{(j)}(n,k)$ is equal to 0_{B_l} for all index $l < j$. If we also use equality (1) (*i.e* $P_{B_lB_k} = 0_{B_l}$ for $l < k$), the first part of the lemma is immediately proved. For the case $j = 1$, we can show by recurrence that $b_{B_1}^{(1)}(n,k) = P_{B_1B_1}^k \mathbf{1}_{B_1}$. Thus, the coefficients $b_{B_l}^{(1)}(n,k)$ do not depend on index n for $l = 1$. We suppose now that this property is true up to $l-1$, that is

$$\forall\, 1 \leq i \leq l-1,\ \forall\, n \geq 1,\ \text{and}\ \forall\, 1 \leq k \leq n,\ b_{B_i}^{(1)}(n,k) \text{ is independent of } n.$$

To prove the property for index l, we proceed by recurrence on k. In fact, for $k = 0$, the term $b_{B_l}^{(1)}(n,0)$ is always equal to $\mathbf{1}_{B_l}$, so it is independent of n. On the other hand, since

$$b_{B_l}^{(1)}(n,k) = \frac{r_l - r_1}{r_l} b_{B_l}^{(1)}(n,k-1) + \frac{r_1}{r_l} \sum_{i=1}^{l} P_{B_lB_i} b_{B_i}^{(1)}(n-1,k-1),$$

we get: $b_{B_l}^{(1)}(n,k)$ is independent of n. □

Appendix B

Proof of corollary 2.3: Using theorem 2.1 and lemma 2.2, we can write

$$\begin{aligned}
\mathbb{P}\{Y_t > s\} &= \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} C_n^k s_1^k (1-s_1)^{n-k} b^{(1)}(k) \\
&= \sum_{k=0}^{\infty} \sum_{n=k}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} C_n^k s_1^k (1-s_1)^{n-k} b^{(1)}(k) \\
&= e^{-\lambda t} \sum_{k=0}^{\infty} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) \sum_{n=k}^{\infty} \frac{(\lambda t(1-s_1))^{n-k}}{(n-k)!}.
\end{aligned}$$

But $\displaystyle\sum_{n=k}^{\infty} \frac{(\lambda t(1-s_1))^{n-k}}{(n-k)!} = e^{\lambda t(1-s_1)}$, then

$$\begin{aligned}
\mathbb{P}\{Y_t > s\} &= e^{-\lambda t} e^{\lambda t(1-s_1)} \sum_{k=0}^{\infty} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k) \\
&= \sum_{k=0}^{\infty} e^{-\lambda t s_1} \frac{(\lambda t s_1)^k}{k!} b^{(1)}(k).
\end{aligned}$$

□

Appendix C

Proof of corollary 2.4: Since

$$b_{B_m}^{(m)}(n,0) = b_{B_m}^{(m-1)}(n,n) \text{ and } b_{B_m}^{(m)}(n,k) = P_{B_m B_m} b_{B_m}^{(m)}(n-1,k-1),$$

we get

$$b_{B_m}^{(m)}(n,k) = P_{B_m B_m}^k b_{B_m}^{(m)}(n-k,0) = P_{B_m B_m}^k b_{B_m}^{(m-1)}(n-k,n-k).$$

Applying theorem 2.1 for $j = m$, we obtain

$$\begin{aligned}
\mathbb{P}\{Y_t > s\} &= \sum_{n\geq 0} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\sum_{k=0}^{n} C_n^k s_m^k (1-s_m)^{n-k}\alpha_{B_m} b_{B_m}^{(m)}(n,k) \\
&= \sum_{n\geq 0} e^{-\lambda t}\frac{(\lambda t)^n}{n!}\sum_{k=0}^{n} C_n^k s_m^k (1-s_m)^{n-k}\alpha_{B_m} P_{B_m}^k b_{B_m}^{(m-1)}(n-k,n-k) \\
&= \sum_{k\geq 0}\sum_{n\geq k} e^{-\lambda t}\frac{(\lambda t)^n}{n!} C_n^k s_m^k (1-s_m)^{n-k}\alpha_{B_m} P_{B_m}^k b_{B_m}^{(m-1)}(n-k,n-k) \\
&= e^{-\lambda t}\sum_{k\geq 0}\frac{(\lambda t s_m)^k}{k!}\alpha_{B_m} P_{B_m}^k \sum_{n\geq k}\frac{(\lambda t(1-s_m))^{n-k}}{(n-k)!} b_{B_m}^{(m-1)}(n-k,n-k).
\end{aligned}$$

If we write $e^{-\lambda t}$ as the product of $e^{-\lambda t s_1}$ and $e^{-\lambda t(1-s_1)}$, we obtain

$$\begin{aligned}
&\mathbb{P}\{Y_t > s\} \\
&= \alpha_{B_m}\left[\sum_{k\geq 0} e^{-\lambda t s_m}\frac{(\lambda t s_m)^k}{k!} P_{B_m}^k\right]\left[\sum_{n\geq k} e^{-\lambda t(1-s_m)}\frac{(\lambda t(1-s_m))^{n-k}}{(n-k)!} b_{B_m}^{(m-1)}(n-k,n-k)\right] \\
&= \alpha_{B_m}\left[\sum_{k\geq 0} e^{-\lambda t s_m}\frac{(\lambda t s_m)^k}{k!} P_{B_m}^k\right]\left[\sum_{n\geq 0} e^{-\lambda t(1-s_m)}\frac{(\lambda t(1-s_m))^{n}}{n!} b_{B_m}^{(m-1)}(n,n)\right] \\
&= \alpha_{B_m} e^{\lambda t s_m (P_{B_m} - I_{B_m})}\left[\sum_{n\geq 0} e^{-\lambda t(1-s_m)}\frac{(\lambda t(1-s_m))^{n}}{n!} b_{B_m}^{(m-1)}(n,n)\right].
\end{aligned}$$

Since $P = \dfrac{A}{\lambda} + I$, then $A_{B_m} = \lambda(P_{B_m} - I_{B_m})$. So

$$\mathbb{P}\{Y_t > s\} = \alpha_{B_m} e^{A_{B_m} t s_m}\left[\sum_{n\geq 0} e^{-\lambda t(1-s_m)}\frac{(\lambda t(1-s_m))^{n}}{n!} b_{B_m}^{(m-1)}(n,n)\right].$$

The proof of corollary 2.4 is then completed. □

Appendix D

To prove theorem 2.5 and theorem 2.6, we first need two technical lemmas which can be proved by recurrence. To simplify notations, we denote the fraction $\frac{r_l - r_j}{r_l - r_{j-1}}$ by $\alpha_{l,j}$. Note that $\frac{r_j - r_{j-1}}{r_l - r_{j-1}} = 1 - \alpha_{l,j}$ and $0 \leq \alpha_{l,j} < 1$ if $l \geq j$.

Lemma 5.1 *For all $1 \leq l \leq m$ and $k \geq 1$, we have :*

$$b_{B_l}^{(1)}(k) \leq b_{B_l}^{(1)}(k-1).$$

Proof. We prove this lemma by recurrence on l. For $l = 1$, we easily obtain $b_{B_1}^{(1)}(k) \leq b_{B_1}^{(1)}(k-1)$ for all $k \in \mathbb{N}^*$ by considering the following system

$$\begin{cases} b_{B_1}^{(1)}(1) \leq b_{B_1}^{(1)}(0) = \mathbf{1}_{B_1} \\ \\ b_{B_1}^{(1)}(k) = P_{B_1} b_{B_1}^{(1)}(k-1) \\ \\ P_{B_1}\mathbf{1}_{B_1} \leq \mathbf{1}_{B_1}. \end{cases}$$

We suppose now that the property is true up to $l-1$. To prove the property for index l, we proceed by recurrence on k. The initial condition for $k = 1$ is verified since $b_{B_l}^{(1)}(1) \leq b_{B_l}^{(1)}(0) = \mathbf{1}_{B_l}$. The property for $k-1$ is equivalent to

$$b_{B_l}^{(1)}(k-1) \leq b_{B_l}^{(1)}(k-2) \tag{3}$$

The lemma 2.2 can be written, for $j = 1$, as

$$\begin{aligned} b_{B_l}^{(1)}(k) &= \alpha_{l,1} b_{B_l}^{(1)}(k-1) + (1-\alpha_{l,1}) \sum_{i=1}^{l} P_{B_l B_i} b_{B_i}^{(1)}(k-1) \\ &= \alpha_{l,1} b_{B_l}^{(1)}(k-1) + (1-\alpha_{l,1}) \left[\sum_{i=1}^{l-1} P_{B_l B_i} b_{B_i}^{(1)}(k-1) + P_{B_l B_l} b_{B_l}^{(1)}(k-1) \right]. \end{aligned}$$

According to the recurrence hypothesis on index $l-1$, we have

$$\forall\, 1 \leq i \leq l-1, \quad b_{B_i}^{(1)}(k-1) \leq b_{B_i}^{(1)}(k-2).$$

Since the P matrix is positive and $\alpha_{l,1} \in [0,1[$, we obtain

$$(1-\alpha_{l,1}) \sum_{i=1}^{l-1} P_{B_l B_i} b_{B_i}^{(1)}(k-1) \leq (1-\alpha_{l,1}) \sum_{i=1}^{l-1} P_{B_l B_i} b_{B_i}^{(1)}(k-2).$$

The recurrence hypothesis (3) gives

$$\begin{aligned} P_{B_l B_l} b_{B_l}^{(1)}(k-1) &\leq P_{B_l B_i} b_{B_l}^{(1)}(k-2) \text{ and} \\ \\ \alpha_{l,1} b_{B_l}^{(1)}(k-1) &\leq \alpha_{l,1} b_{B_l}^{(1)}(k-2). \end{aligned}$$

It follows that

$$\begin{aligned} b_{B_l}^{(1)}(k) &\leq \alpha_{l,1} b_{B_l}^{(1)}(k-2) + (1-\alpha_{l,1}) \sum_{i=1}^{l} P_{B_l B_i} b_{B_i}^{(1)}(k-2) \\ &= b_{B_l}^{(1)}(k-1). \end{aligned}$$

The proof of the lemma is completed. □

Remark: According to lemma 2.2, we observe that the coefficient $b_{B_l}^{(j)}(n,k)$ is a convex combination of the two vectors $b_{B_l}^{(j)}(n,k-1)$ and $\sum_{i=j}^{l} P_{B_lB_i} b_{B_i}^{(j)}(n-1,k-1)$. So, we obtain the following equivalence

$$b_{B_l}^{(j)}(n,k) \leq b_{B_l}^{(j)}(n,k-1) \Longleftrightarrow \sum_{i=j}^{l} P_{B_lB_i} b_{B_i}^{(j)}(n-1,k-1) \leq b_{B_l}^{(j)}(n,k-1) \tag{4}$$

This remark above will be exploited in the following lemma.

Lemma 5.2 *For $n \geq 1$, $2 \leq j \leq m$ and $j \leq l \leq m$, we have*

$$b_{B_l}^{(j-1)}(n,n) \leq b_{B_l}^{(j-1)}(n,n-1) \Longrightarrow b_{B_l}^{(j)}(n,1) \leq b_{B_l}^{(j)}(n,0)$$

Proof. Since we have

$$b_{B_l}^{(j)}(n,0) = b_{B_l}^{(j-1)}(n,n) = \alpha_{l,j-1} b_{B_l}^{(j-1)}(n,n-1) + (1-\alpha_{l,j-1}) \sum_{i=j-1}^{l} P_{B_lB_i} b_{B_i}^{(j-1)}(n-1,n-1),$$

then

$$\sum_{i=j}^{l} P_{B_lB_i} b_{B_i}^{(j)}(n-1,0) \leq b_{B_l}^{(j)}(n,0)$$

$$\Updownarrow$$

$$\sum_{i=j}^{l} P_{B_lB_i} b_{B_i}^{(j)}(n-1,0) \leq \alpha_{l,j-1} b_{B_l}^{(j-1)}(n,n-1) + (1-\alpha_{l,j-1}) \sum_{i=j-1}^{l} P_{B_lB_i} b_{B_i}^{(j-1)}(n-1,n-1)$$

$$\Updownarrow$$

$$0_{B_l} \leq \alpha_{l,j-1} \left[b_{B_l}^{(j-1)}(n,n-1) - \sum_{i=j-1}^{l} P_{B_lB_i} b_{B_i}^{(j-1)}(n-1,n-1) \right] + P_{B_lB_{j-1}} b_{B_{j-1}}^{(j-1)}(n-1,n-1)$$

So, if $b_{B_l}^{(j-1)}(n,n) \leq b_{B_l}^{(j-1)}(n,n-1)$ then, according to (4), the vector between hooks is greater than 0_{B_l}. This is equivalent to $\sum_{i=j}^{l} P_{B_lB_i} b_{B_i}^{(j)}(n-1,0) \leq b_{B_l}^{(j)}(n,0)$. By taking into account the equivalence (4), we get $b_{B_l}^{(j)}(n,1) \leq b_{B_l}^{(j)}(n,0)$. □

Proof of theorem 2.5:

We prove theorem 2.5 by recurrence on the indices j, n and k. For $1 \leq j \leq m$ fixed, we consider the following property $\Re(j)$

$$\Re(j) : \left[\forall\, n \geq 1,\ \forall\, 1 \leq k \leq n \text{ and } \forall\, j \leq l \leq m,\ \ b_{B_l}^{(j)}(n,k) \leq b_{B_l}^{(j)}(n,k-1) \right]$$

For $j = 1$, according to lemma 5.1, the vectors $b_{B_l}^{(1)}(n,k)$ are independent of n, so $\Re(1)$ is verified. We suppose that $\Re(j-1)$ is true. In order to prove the property $\Re(j)$, we proceed by recurrence on n. For $n = 1$, the recurrence hypothesis $\Re(j-1)$ leads to $b_{B_l}^{(j-1)}(1,1) \leq b_{B_l}^{(j-1)}(1,0)$, which gives according to lemma 5.2

$$b_{B_l}^{(j)}(1,1) \leq b_{B_l}^{(j)}(1,0).$$

We suppose now that the property holds for $n-1$. This means that the property $\Re(j,n-1)$ is true, where

$$\Re(j,n-1) \text{ is : } \left[\forall\, 1 \leq k \leq n-1 \text{ et } \forall\, j \leq l \leq m,\ \ b_{B_l}^{(j)}(n-1,k) \leq b_{B_l}^{(j)}(n-1,k-1)\right]$$

In order to prove $\Re(j,n)$, we proceed by recurrence on k. For $k=1$, the hypothesis $\Re(j-1)$ allows us to write $b_{B_l}^{(j-1)}(n,n) \leq b_{B_l}^{(j-1)}(n,n-1)$, which gives according to lemma 5.2,

$$b_{B_l}^{(j)}(n,1) \leq b_{B_l}^{(j)}(n,0).$$

At present, we suppose the property true for $k-1$. That means that property $\Re(j,n,k-1)$ is true, where

$$\Re(j,n,k-1) \text{ is : } \left[\forall\, j \leq l \leq m,\ \ b_{B_l}^{(j)}(n-1,k) \leq b_{B_l}^{(j)}(n-1,k-1)\right]$$

According to lemma 2.2, $b_{B_l}^{(j)}(n,k)$ and $b_{B_l}^{(j)}(n,k-1)$ can be written as :

$$\begin{aligned} b_{B_l}^{(j)}(n,k) &= \alpha_{l,j} b_{B_l}^{(j)}(n,k-1) + (1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-1,k-1) \\ b_{B_l}^{(j)}(n,k-1) &= \alpha_{l,j} b_{B_l}^{(j)}(n,k-2) + (1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-1,k-2). \end{aligned}$$

Moreover, according to $\Re(j,n-1)$, we have $b_{B_i}^{(j)}(n-1,k-1) \leq b_{B_i}^{(j)}(n-1,k-2)$, and, since the matrix P is positive and $0 \leq \alpha_{l,j} < 1$, we get

$$(1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-1,k-1) \leq (1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-1,k-2).$$

The hypothesis $\Re(j,n,k-1)$ and the previous expressions for $b_{B_l}^{(j)}(n,k)$ and for $b_{B_l}^{(j)}(n,k-1)$ lead to the required inequality

$$b_{B_l}^{(j)}(n,k) \leq b_{B_l}^{(j)}(n,k-1).$$

The proof of theorem 2.5 is then completed. □

Proof of theorem 2.6: For $j=1$, the sequence $(b_{B_l}^{(1)}(n,k))_{n \geq k}$ is independent of n. We suppose now that the property holds for $j-1$. We must prove it for the the index j. The elements $b_{B_l}^{(j)}(n,0)$ and $b_{B_l}^{(j)}(n-1,0)$ are respectively equal to $b_{B_l}^{(j-1)}(n,n)$ and $b_{B_l}^{(j-1)}(n-1,n-1)$. According to theorem 2.5, we have $b_{B_l}^{(j-1)}(n,n) \leq b_{B_l}^{(j-1)}(n,n-1)$, and according to the recurrence hypothesis, we have $b_{B_l}^{(j-1)}(n,n-1) \leq b_{B_l}^{(j-1)}(n-1,n-1)$. So, we get $b_{B_l}^{(j-1)}(n,n) \leq b_{B_l}^{(j-1)}(n-1,n-1)$. It follows that

$$b_{B_l}^{(j)}(n,0) \leq b_{B_l}^{(j)}(n-1,0).$$

This inequality proves the result for $k=0$. We suppose now that the property holds for $k-1$ and we show that $b_{B_l}^{(j)}(n,k) \leq b_{B_l}^{(j)}(n-1,k)$. For this purpose, we write

$$b_{B_l}^{(j)}(n,k) = \alpha_{l,j} b_{B_l}^{(j)}(n,k-1) + (1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-1,k-1)$$

$$\leq \alpha_{l,j} b_{B_l}^{(j)}(n-1,k-1) + (1-\alpha_{l,j}) \sum_{i=j}^{l} P_{B_l B_i} b_{B_i}^{(j)}(n-2,k-1)$$

$$= b_{B_l}^{(j)}(n-1,k).$$

Which completes the proof of theorem 2.6. □

References

[1] J. F. Meyer. – Closed-form solutions of performability. – *IEEE Trans. Computers*, C-31(7):648–657, July 1982.

[2] D. G. Furchtgott. – *Performability models solutions.* – PhD thesis, University of Michigan, January 1984.

[3] J. F. Meyer, D. G. Furchtgott, and L. T. Wu. – Performability evaluation of the sift computer. – *IEEE Transactions on Computers*, C-29(6):501–509, June 1980.

[4] M. D. Beaudry. – Performance-related reliability measures for computing systems. – *IEEE Trans. Computers*, C-27:540–547, June 1978.

[5] G. Ciardo, R. Marie, B. Sericola, and K. S. Trivedi. – Performability analysis using semi-Markov reward processes. – *IEEE Trans. Computers*, C-39:1251–1264, October 1990.

[6] A. Goyal and A. N. Tantawi. – Evaluation of performability for degradable computer systems. – *IEEE Trans. Computers*, C-36(6):738–744, June 1987.

[7] S. M. Ross. – *Stochastic Processes.* – John Wiley and Sons, 1983.

[8] H. Nabli and B. Sericola. – Performability analysis of fault-tolerant computer systems. – *Rapport INRIA*, (2254), May 1994. – To appear in IEEE Trans. Computers.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399