Comments

Comment on "Performability Analysis: A New Algorithm"

Victor Suñé,
Juan A. Carrasco, *Senior Member, IEEE*,
Hédi Nabli, and Bruno Sericola

Abstract—The paper "Performability Analysis: A New Algorithm" describes an algorithm for computing the complementary distribution of the accumulated reward over an interval of time in a homogeneous Markov process. In this comment, we show that in two particular cases, one of which is quite frequent, small modifications of the algorithm may reduce significantly its storage complexity.

Index Terms—Fault tolerance, repairable systems, Markov processes, performability, performance, reliability, uniformization.

1 Introduction

In [1], an algorithm is described for computing the complementary distribution $\mathbb{P}\{Y_t > s\}$ of the accumulated reward over the interval of time $[0,t]$ in a homogeneous Markov process with finite state space and time-independent reward rates associated with states. A salient feature of the algorithm is that it only deals with nonnegative numbers bounded by 1, ensuring numerical stability. Using the notation in [1], $\mathbb{P}\{Y_t > s\}$ is computed with absolute error bounded from above by $\varepsilon/2$ using

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{N} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \sum_{k=0}^{n} \sum_{j=1}^{m} \binom{n}{k} s_j^k (1-s_j)^{n-k} b^{(j)}(n,k) \mathbf{1}_{\{r_{j-1}t \le s < r_j t\}}, \quad (1)$$

where λ is the uniformization rate, N is a nonnegative truncation parameter defined as

$$N = \min\Big\{ n \in \mathbb{N} \;\Big|\; \sum_{j=0}^{n} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^j}{j!} \ge 1 - \frac{\varepsilon}{2} \Big\},$$

$m+1$ is the number of different reward rates $0 = r_0 < r_1 < \ldots r_m$, $s_j = (s - r_{j-1}t)/((r_j - r_{j-1})t)$, and each scalar $b^{(j)}(n,k)$ can be obtained from vectors $b_{U_j}(n,k)$ and $b_{D_j}(n,k)$. Vectors $b_{U_1}(n,0)$, $b_{U_j}(0,0)$, $j > 1$, $b_{D_j}(0,0)$, $j < m$, and $b_{D_m}(n,n)$ are known constant vectors. The remaining vectors are proposed to be computed following a given ordering. In the particular but frequent case $r_{m-1}t < s < r_m t$, $\mathbb{P}\{Y_t > s\}$ is obtained with absolute error bounded from above by ε using

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{C} \sum_{n=k}^{N} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \binom{n}{k} s_m^{n-k} (1-s_m)^k b^{(m)}(n, n-k), \quad (2)$$

where C is a second nonnegative truncation parameter defined as

$$C = \min\Big\{ c \in \mathbb{N} \;\Big|\; \sum_{h=0}^{c} \mathrm{e}^{-\lambda t(1-s_m)} \frac{(\lambda t(1-s_m))^h}{h!} \ge 1 - \frac{\varepsilon}{2} \Big\},$$

which is never larger and often much smaller than N. Ignoring the storage of the transition probability matrix of the uniformized Markov chain, the proposed ordering for the computation of the collection of vectors $b_{U_j}(n,k)$, $b_{D_j}(n,k)$ results in a storage complexity of the algorithm that is stated to be $O(mNM)$ in the case $s \le r_{m-1}t$, where M is the number of states, and $O([(m-1)C+N]M)$ in the case $r_{m-1}t < s < r_m t$.

In this comment, we start by noting that, using (1), Equation (2) holds also when $s = r_{m-1}t$, and that, therefore, computing vectors $b_{U_j}(n,k)$ and $b_{D_j}(n,k)$ following the ordering given in [1], the storage complexity of the algorithm is $O(mNM)$ in the case $s < r_{m-1}t$ and $O([(m-1)C+N]M)$ in the case $r_{m-1}t \le s < r_m t$. In addition, in this second case, which is quite frequent, small modifications of the algorithm make its storage complexity $O(mCM)$ for $C > 1$ and $O(M)$ for $C = 1$. The modifications involve a reformulation of (2) and modifications of the collections of vectors $b_{U_j}(n,k)$, $b_{D_j}(n,k)$ that are computed and of the order in which they are computed. The reformulation of (2) is

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{N} \sum_{k=0}^{\min\{n,C\}} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \binom{n}{k} s_m^{n-k} (1-s_m)^k b^{(m)}(n, n-k). \quad (3)$$

The collection of vectors that are computed and the order in which they are computed are shown in Fig. 1. The upper part of cell (j,n,k) represents vector $b_{U_j}(n,k)$, the lower part of cell (j,n,k) represents vector $b_{D_j}(n,k)$, 0_{D_j} and 1_{D_j} represent constant vectors with elements equal to respectively 0 and 1, and computed vectors are indicated as "( )". The storage complexity results from the facts that, using (3), $\mathbb{P}\{Y_t > s\}$ can be computed obtaining $b^{(m)}(n, n-k)$ in groups including the scalars corresponding to the same n value, sorted by increasing n, and that, using the recursive expressions given in Theorem 2.1 of [1], noting that $b_{U_m}(n,k)$, $n > 0$, $k > 0$ does not depend on $b_{U_m}(n, k-1)$, each vector can be computed using a subset of vectors including known constant vectors, the previously computed vector and, for $n = i > 1$, computed vectors of the previous row $(n = i-1)$ with the same j value.

As said previously, in the case $s < r_{m-1}t$, the storage complexity of the algorithm is $O(mNM)$. However, there exists a second particular case in which the storage complexity can also be improved. That particular case is $0 \le s < r_1 t$. In that case, it is possible [2] to define a truncation parameter

$$C' = \min\Big\{ c \in \mathbb{N} \;\Big|\; \sum_{h=0}^{c} \mathrm{e}^{-\lambda t s_1} \frac{(\lambda t s_1)^h}{h!} \ge 1 - \frac{\varepsilon}{2} \Big\},$$

- *V. Suñé is with Departament d'Enginyeria Electrònica,Universitat Politècnica de Catalunya, EPSC, Avda. Canal Olímpic 15, plta. 1, 08860 Castelldefels, Spain. E-mail: sunye@eel.upc.edu.*
- *J.A. Carrasco is with Departament d'Enginyeria Electrònica,Universitat Politècnica de Catalunya, ETSEIB, Avda. Diagonal 647, plta. 9, 08028 Barcelona, Spain. E-mail: carrasco@eel.upc.edu.*
- *H. Nabli is with Faculté des Sciences de Sfax, Route de Soukra Km 3, BP 1171, Sfax 3000, Tunisia. E-mail: Hedi.Nabli@fsm.rnu.tn.*
- *B. Sericola is with INRIA, Rennes-Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes cedex, France. E-mail: Bruno.Sericola@inria.fr.*

Manuscript received 10 Apr. 2007; accepted 5 Jan. 2009; published online 23 July 2009.
Recommended for acceptance by Y. Mazin.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2007-04-0124.
Digital Object Identifier no. 10.1109/TC.2009.114.

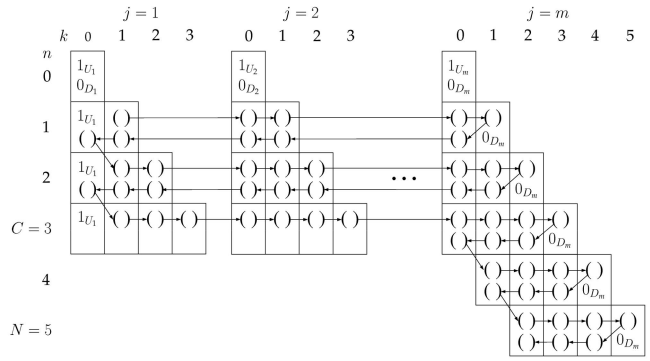

Fig. 1. Collection of computed vectors $b_{U_j}(n,k)$, $b_{D_j}(n,k)$ and order in which they are computed in the modified algorithm for the case $r_{m-1}t \le s < r_m t$, $C > 0$.

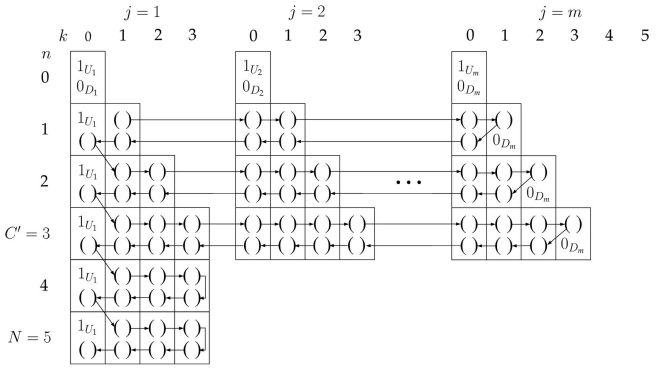

Fig. 2. Collection of computed vectors $b_{U_j}(n,k)$, $b_{D_j}(n,k)$ and order in which they

which is never larger than N, and compute $\mathbb{P}\{Y_t > s\}$ with absolute error bounded from above by ε using [2]

$$\mathbb{P}\{Y_t > s\} = \sum_{k=0}^{C'} \sum_{n=k}^{N} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \binom{n}{k} s_1^k (1-s_1)^{n-k} b^{(1)}(n,k). \quad (4)$$

Small modifications of the algorithm in that second particular case make its storage complexity $O(mC'M)$ for $C' > 1$ and $O(M)$ for $C' = 1$. The modifications involve a reformulation of (4) and modifications of the collections of vectors $b_{U_j}(n,k)$, $b_{D_j}(n,k)$ that are computed and of the order in which they are computed. The reformulation of (4) is

$$\mathbb{P}\{Y_t > s\} = \sum_{n=0}^{N} \sum_{k=\max\{0,n-C'\}}^{n} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \binom{n}{k} s_1^{n-k} (1-s_1)^k \, b^{(1)}(n, n-k). \quad (5)$$

The collection of vectors that are computed and the order in which they are computed are shown in Fig. 2. The storage complexity follows from the facts that, using (5), $\mathbb{P}\{Y_t > s\}$ can be computed obtaining $b^{(1)}(n, n-k)$ in groups including the scalars corresponding to the same n value, sorted by increasing n, and that, using the recursive expressions given in Theorem 2.1 of [1], noting that $b_{D_1}(n,k)$, $n > 0$, $k < n$ does not depend on $b_{D_1}(n, k+1)$, each vector can be computed using a subset of vectors including known constant vectors, the previously computed vector and, for $n = i > 1$, computed vectors of the previous row $(n = i-1)$ with the same j value.

We end the comment by noting that in the cases $r_{m-1}t \le s < r_m t$, $C = 0$, and $0 \le s < r_1 t$, $C' = 0$, the storage complexity of the algorithm is $O(M)$. This results from the facts that, as noted previously, (2) holds also when $s = r_{m-1}t$, that, using (2) (resp., (4)), $\mathbb{P}\{Y_t > s\}$ can be computed obtaining $b^{(m)}(n,n)$ (resp., $b^{(1)}(n,0)$) sorted by increasing n, and that, using the recursive expressions given in Theorem 2.1 of [1], noting that $b_{U_m}(n,k)$, $n > 0$, $k > 0$ does not depend on $b_{U_m}(n, k-1)$ and that $b_{D_1}(n,k)$, $n > 0$, $k < n$ does not depend on $b_{D_1}(n, k+1)$, each vector can be computed using a subset of vectors including known constant vectors and the previously computed vector.

REFERENCES

[1] H. Nabli and B. Sericola, "Performability Analysis: A New Algorithm," *IEEE Trans. on Computers* vol. 45, no. 4, pp. 491-494, 1996.
[2] H. Nabli and B. Sericola, "Performability Analysis of Fault-Tolerant Computer Systems," Technical Report 2254, INRIA, May 1994.