Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A NOTE ON CALCULATING TRANSIENT DISTRIBUTIONS OF CUMULATIVE REWARD

E. de Souza e Silva
H. R. Gail

November 1993
CSD-930038

A Note on Calculating Transient Distributions of Cumulative Reward

Edmundo de Souza e Silva[1]
UCLA Computer Science Department
Los Angeles, CA 90024

H. Richard Gail
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

In a recent report, the authors developed an algorithm based on probabilistic arguments to calculate numerically the transient distribution of cumulative reward. In this short note it is shown that the computational requirements of the approach can be further reduced without altering the basic steps of the algorithm. The computational savings are obtained by grouping terms of the previous recursion.

[1]This work was done while E. de Souza e Silva was on leave from the Federal University of Rio de Janeiro partially supported by grants from CNPq(Brazil) and NSF CCR-9215064.

1 Introduction

In a recent report [3], the authors developed an algorithm to calculate numerically the distribution of cumulative reward over a finite observation period. The algorithm is based on the general methodology of [2], and only probabilistic arguments were used in its development. It was shown in [3] that the computational requirements of the algorithm are considerably less than those of the method of [2]. In fact, the number of operations required by the algorithm of [3] is $O(mdMN^3)$, and the storage requirements are $O(MN^2)$. Here m is a parameter which is smaller than the total number of rewards, d is the average number of nonzero entries in the columns of the uniformized one step state transition probability matrix $\mathbf{P}$, M is the total number of states, and N is the truncation point for the infinite series of the main equation (40) in [3].

In this short note, we show that the computational complexity of the algorithm can be reduced to $O(mdMN^2)$ operations with storage requirements of $O(MN)$, simply by grouping terms in the main recursion of [3]. The basic development remains the same. In the next section we show how this reduction can be accomplished. In order to avoid repetition, we assume that the reader is familiar with [3] and the notation used there.

2 The Grouping

The main recursion for calculating the distribution of cumulative reward over a finite observation period is derived in Lemma 3 of [3]. This lemma describes the calculation of the quantity $\Omega_s[g, i, n, l]$, which is used in the final algorithm. The recursion is as follows.

Let $s \in \mathcal{S}$, $i = 1, \ldots, m$, $g = 0, \ldots, n+1$. For $n \geq 1$, $l \geq 0$ and for $n = 0$, $l > 0$

$$\Omega_s[g, i, n, l] = \begin{cases} \left(\{\boldsymbol{\Omega}[g, i, n-1, l-1] + \omega_i \boldsymbol{\Omega}[g, i, n-1, l]\}\, \mathbf{P}[:s] - \Omega_s[g, i, n, l-1]\right)/\omega_{i,c(s)} & c(s) \neq i \\ \{\boldsymbol{\Omega}[g-1, i, n-1, l-1] + \omega_i \boldsymbol{\Omega}[g-1, i, n-1, l]\}\, \mathbf{P}[:s] & c(s) = i \end{cases} \tag{1}$$

where $\omega_{i,j} = r_i - r_j$ for $i \neq j$, $\omega_i = r_i - r$, and $\mathbf{P}[:s]$ is the sth column of $\mathbf{P}$.

The initial conditions are (for $n = 0$, $l = 0$)

$$\Omega_s[g, i, 0, 0] = \begin{cases} \pi_s(0)/\omega_{i,c(s)} & c(s) \neq i,\ g = 0 \\ 0 & c(s) \neq i,\ g = 1 \\ 0 & c(s) = i,\ g = 0 \\ \pi_s(0) & c(s) = i,\ g = 1 \end{cases} \tag{2}$$

where $\pi_s(0)$ is the initial probability of state s.

We will show that many of the terms Ω_s in the above recursion can be grouped to calculate the measure of interest, $P[\text{ACR}(t) > r]$. For this, we define the following.

Definition 1 *For $n \geq 0$, $i = 1, \ldots, m$, $s \in \mathcal{S}$, $u \geq 0$, let*

$$\Upsilon_s[i, n, u] = \sum_{g=0}^{n+1} \Omega_s[g, i, n, g + u - (n+1)]. \tag{3}$$

Note that $\Upsilon_s[i, n, u] = 0$ for $n < 0$ and for $u < 0$, in the latter case because $g \leq n + 1$. We now use equation (1) and Definition 1 to obtain a recursion for the vector

$$\boldsymbol{\Upsilon}[i, n, u] = \langle \Upsilon_{s_1}[i, n, u], \ldots, \Upsilon_{s_M}[i, n, u] \rangle. \tag{4}$$

Lemma 1 *Let $s \in \mathcal{S}$, $i = 1, \ldots, m$. For $n \geq 1$, $u \geq 0$ and for $n = 0$, $u > 1$*

$$\Upsilon_s[i, n, u] = \\ \begin{cases} \left(\{\boldsymbol{\Upsilon}[i, n-1, u-2] + \omega_i \boldsymbol{\Upsilon}[i, n-1, u-1]\} \mathbf{P}[: s] - \Upsilon_s[i, n, u-1] \right) / \omega_{i, c(s)} & c(s) \neq i \\ \{\boldsymbol{\Upsilon}[i, n-1, u-1] + \omega_i \boldsymbol{\Upsilon}[i, n-1, u]\} \mathbf{P}[: s] & c(s) = i \end{cases} \tag{5}$$

The initial conditions are (for $n = 0$, $u = 0, 1$)

$$\Upsilon_s[i, 0, u] = \begin{cases} 0 & c(s) \neq i, \ u = 0 \\ \pi_s(0)/\omega_{i, c(s)} & c(s) \neq i, \ u = 1 \\ \pi_s(0) & c(s) = i, \ u = 0 \\ 0 & c(s) = i, \ u = 1 \end{cases} \tag{6}$$

Proof: We first consider the case $c(s) \neq i$. It has already been observed in [3] that in this case $\Omega_s[n+1, i, n, l] = 0$, and so the expression for Υ_s in (3) involves only the terms from $g = 0$ up to $g = n$ for any $u \geq 0$. Setting $l = g + u - n - 1$ in (1) and adding the resulting equations for $g = 0, \ldots, n$, we have by applying Definition 1 that

$$\Upsilon_s[i, n, u] = \\ \Big(\Big\{ \textstyle\sum_{g=0}^{n} \boldsymbol{\Omega}[g, i, n-1, g+u-n-2] + \omega_i \sum_{g=0}^{n} \boldsymbol{\Omega}[g, i, n-1, g+u-n-1] \Big\} \mathbf{P}[: s] \\ - \textstyle\sum_{g=0}^{n} \Omega_s[g, i, n, g+u-n-2] \Big) / \omega_{i, c(s)} \tag{7}$$

Noting that $g + u - n - 2 = g + (u-2) - n$, we recognize the first sum on the right hand side of (7) as $\boldsymbol{\Upsilon}[i, n-1, u-2]$ from Definition 1. Similarly, the second sum is $\boldsymbol{\Upsilon}[i, n-1, u-1]$.

Finally, writing $g+u-n-2=g+(u-1)-(n+1)$ and again using $\Omega_s[n+1,i,n,l]=0$, we see that the third sum is $\Upsilon_s[i,n,u-1]$.

For $c(s)=i$, we observe that $\Omega_s[0,i,n,l]=0$ for arbitrary l, and thus the expression for Υ_s only involves the terms from $g=1$ up to $g=n+1$ in this case. Setting $l=g+u-n-1$ in (1) and adding the resulting equations for $g=1,\ldots,n+1$, we have as before that

$$\Upsilon_s[i,n,u]=$$
$$\left\{\textstyle\sum_{g=1}^{n+1}\boldsymbol{\Omega}[g-1,i,n-1,g+u-n-2]+\omega_i\sum_{g=1}^{n+1}\boldsymbol{\Omega}[g-1,i,n-1,g+u-n-1]\right\}\mathbf{P}[:s]$$

or

$$\Upsilon_s[i,n,u]=\left\{\sum_{g=0}^{n}\boldsymbol{\Omega}[g,i,n-1,g+(u-1)-n]+\omega_i\sum_{g=0}^{n}\boldsymbol{\Omega}[g,i,n-1,g+u-n]\right\}\mathbf{P}[:s].$$

Applying Definition 1, we obtain the second equation in (1).

The initial conditions can be easily obtained from (2) and (3). □

It is useful to note from Definition 1 that $\Upsilon_s[i,0,u]=\Omega_s[0,i,0,u-1]+\Omega_s[1,i,0,u]$, and so $\Upsilon_s[i,0,u]=0$ for $c(s)=i$ and $u>1$ since $\Omega_s[g,c(s),0,l]=0$ for $l>0$ and arbitrary g. Furthermore, $\Upsilon_s[i,n,0]=\Omega_s[n+1,i,n,0]$, and so $\Upsilon_s[i,n,0]=0$ for $c(s)\neq i$ (see the proof of Lemma 1).

Theorem 1 of [3] gives the main equation to calculate the distribution of cumulative reward averaged over t as

$$P[\mathrm{ACR}(t)>r]=\sum_{n=0}^{\infty}e^{-\Lambda t}\frac{(\Lambda t)^n}{n!}\sum_{i=1}^{m}\sum_{g=1}^{n+1}\|\boldsymbol{\Omega}[g,i,n,g-1]\|. \tag{8}$$

Since $\sum_{g=1}^{n+1}\|\boldsymbol{\Omega}[g,i,n,g-1]\|=\|\boldsymbol{\Upsilon}[i,n,n]\|$, the equation above can be rewritten to yield the following result.

Theorem 1 *The distribution of the total reward accumulated during* $(0,t)$ *averaged over t is given by*

$$P[\mathrm{ACR}(t)>r]=\sum_{n=0}^{\infty}e^{-\Lambda t}\frac{(\Lambda t)^n}{n!}\sum_{i=1}^{m}\|\boldsymbol{\Upsilon}[i,n,n]\|. \tag{9}$$

Figure 1 illustrates the recursion for $\boldsymbol{\Upsilon}$. In this figure the dots represent a vector $\boldsymbol{\Upsilon}[i,n,u]$ for a given i, and the arrows indicate the values needed to calculate this vector. The dots on the diagonal line represent the values that are needed to compute $P[\mathrm{ACR}(t)>r]$ using Theorem 1.

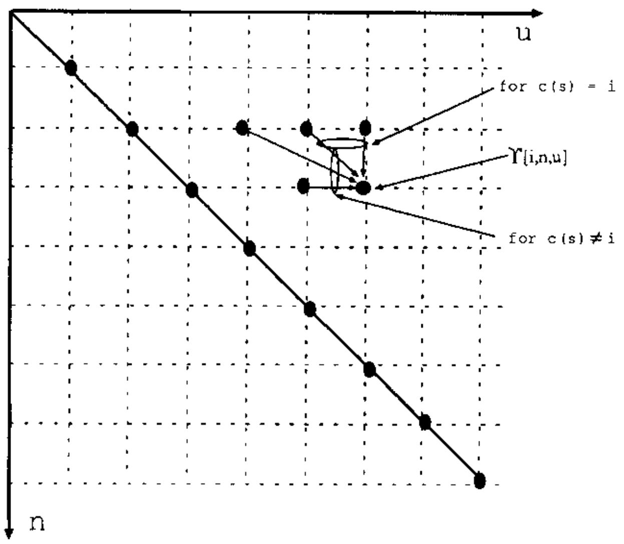

Figure 1: Recursion for Υ.

3 Computational Complexity

The main computational effort in evaluating equation (9) is the computation of $\Upsilon_s[i,n,u]$ for $i = 1,\ldots,m$, $n = 0,\ldots,N$, $u = 0,\ldots,N$. Here N is the truncation point of the infinite series in (9) that produces the desired error tolerance. For a given value of i, the recursion should be carried out from $n = 0$ to N and, for each n, from $u = 0$ to N. To obtain each value of Υ_s requires a simple vector by vector multiplication, where one of the vectors is a column of $\mathbf{P}$. The matrix $\mathbf{P}$ is usually sparse, and we assume that on the average there are d nonzero entries in a column of $\mathbf{P}$. Then $O(dMn)$ multiplications are needed to obtain $\boldsymbol{\Upsilon}$ for a given value of n and i, and thus a total of $O(dmMN^2)$ multiplications are required for all values of n and i where $m < K+1$. Note that, as stated in [3], the value of m is equal to or near 1 when we evaluate the tail of the distribution.

Similar to the computation of $\boldsymbol{\Omega}$, the computation of $\boldsymbol{\Upsilon}$ can be done independently for each i. Furthermore, for a given n, only the values calculated for $n-1$ are needed. As a consequence, a total of N vectors of dimension M need to be stored, *independent* of the number of rewards of the model. In fact, this is the same storage required by the algorithm for the case of two rewards previously developed by the authors in [1] when recursion by column is used.

The method in [4], which combines uniformization and Laplace transform techniques, is stated to have $O(d(K+1)MN^2)$ number of operations and $O((K+1)MN)$ storage requirements, which is not independent of the number of rewards. Thus the algorithm developed in this note compares favorably with the method of [4].

4 Remarks

As noted in [3], it is convenient to scale the rewards and define a scaling factor for Ω_s. We can apply the same scaling for Υ_s as follows.

Definition 2 *Let*

$$\Upsilon^*_s[i,n,u] = \omega^u \Upsilon_s[i,n,u] \tag{10}$$

and

$$\omega^*_{i,c(s)} = \frac{\omega}{\omega_{i,c(s)}}, \tag{11}$$

where $\omega = \min_{i,j}\{\omega_{i,j}\}$, $1 \le i \le m$, $1 \le j \le K+1$, $i \ne j$.

Then we can replace equation (43) of [3] by

$$\Upsilon^*_s[i,n,u] = \begin{cases} \omega^*_{i,c(s)}\left(\{\omega\boldsymbol{\Upsilon}^*[i,n-1,u-2] + \omega^*_i\boldsymbol{\Upsilon}^*[i,n-1,u-1]\}\,\mathbf{P}[:s] - \Upsilon^*_s[i,n,u-1]\right) & i \ne c(s) \\ \{\omega\boldsymbol{\Upsilon}^*[i,n-1,u-1] + \omega^*_i\boldsymbol{\Upsilon}^*[i,n-1,u]\}\,\mathbf{P}[:s] & i = c(s) \end{cases} \tag{12}$$

where recall from [3] that $\omega^*_i = \omega_i/\omega_1$. Finally, we define for $n = 0, \ldots, N$

$$\Theta(n) = \sum_{i=1}^{m} \|\boldsymbol{\Upsilon}^*[i,n,n]\|,$$

and, as in [3], we have $P[\mathrm{ACR}(t) > r|n] = \omega^{-n}\Theta(n)$.

References

[1] E. de Souza e Silva and H.R. Gail. Calculating cumulative operational time distributions of repairable computer systems. *IEEE Trans. on Computers*, C-35(4):322–332, 1986.

[2] E. de Souza e Silva and H.R. Gail. Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM*, 36(1):171–193, 1989.

[3] E. de Souza e Silva and H.R. Gail. Calculating transient distributions of cumulative reward. Technical report, UCLA CSD-930033, UCLA Computer Science Department, Los Angeles, CA 90024, September 1993.

[4] L. Donatiello and V. Grassi. On evaluating the cumulative performance distribution of fault-tolerant computer systems. *IEEE Trans. on Computers*, 40(11):1301–1307, 1991.