Performability Analysis Using Semi-Markov Reward Processes

GIANFRANCO CIARDO, MEMBER, IEEE, RAYMOND A. MARIE, BRUNO SERICOLA, AND
KISHOR S. TRIVEDI, SENIOR MEMBER, IEEE

Abstract — With the increasing complexity of multiprocessor and distributed processing systems, the need to develop efficient and accurate modeling methods is evident. Fault tolerance and degradable performance of such systems has given rise to considerable interest in models for the combined evaluation of performance and reliability [1], [2]. Markov or semi-Markov reward models can be used to evaluate the effectiveness of degradable fault-tolerant systems. Beaudry [1] proposed a simple method for computing the distribution of performability in a Markov reward process. We present two extensions of Beaudry's approach. First, we generalize the method to a semi-Markov reward process. Second, we remove the restriction requiring the association of zero reward to absorbing states only. We illustrate the use of the approach with three interesting applications.

Index Terms—Computer performance, computer reliability, graceful degradation, Markov models, performability, reward processes, semi-Markov models.

I. INTRODUCTION

MULTIPROCESSORS and distributed processing systems can provide higher performance and higher reliability/availability over single-processor systems. In order to properly assess the effectiveness of such systems, measures that combine performance and reliability are needed. For this purpose, Meyer [2] developed a conceptual framework of performability. Markov and semi-Markov reward models are used in the performability evaluation of computer or communication systems.

This paper presents a new algorithm for the computation of the distribution of accumulated reward until absorption in a semi-Markov reward process. Failures and repairs of system resources are modeled by a semi-Markov process, called the structure-state process [2]. A performance level, or reward rate, is associated with each state of the structure-state process. The resulting semi-Markov reward process is then able to capture not only failure and repair of system components, but degradable performance as well. Application examples used to illustrate our method are fairly diverse and interesting. The first example is a repairable system with a limited number of repairs [3]. The second example is the M/M/2 queueing system subject to failure and repair. The final example is based on a measurement-based performability model of a large IBM system [4].

In [1], Beaudry proposed an algorithm for computing the distribution of the accumulated reward until absorption in a continuous-time Markov chain. We extend her result in two directions. First, we consider semi-Markov reward processes, thus removing the restriction of exponentially distributed sojourn times. We have observed in [4] that the use of an exponential distribution as an approximation to a non-exponential distribution can cause considerable errors in the results. Second, we allow nonabsorbing states with zero reward rate, which do occur in practical models [4]. In Beaudry's method, these states would have caused a "division by zero" in the computations or would have required approximating their zero reward rate using a small positive quantity instead, but we caution the reader that the numerical errors due to such an approximation can be large (as will be shown in one of the examples of Section VI).

In Section II, the problem of computing the distribution of accumulated reward for a semi-Markov process is introduced; in Section III, an informal approach to the solution of this problem is given; in Section IV, a transformation is defined to obtain a new semi-Markov process whose lifetime is distributed as the accumulated reward of the original process. The efficient construction of the new semi-Markov process is discussed in Section V, while examples of the application of our method to the modeling of computer systems are presented in Section VI. Section VII contains some considerations for further extensions. Proofs of important results are given in the Appendix.

II. THE SEMI-MARKOV REWARD PROCESS

Assume that \{X(t), t \geq 0\} is a right continuous semi-Markov process with state space \( S = \{0, 1, 2, \ldots\} \) and with probability of being eventually absorbed equal to 1. Let \( T_k \) be the time of the \( k \)th transition \( \left(T_0 = 0\right) \) and define \( V_k = T_{k+1} - T_k \), the sojourn time in the \( k \)th visited state \( (k \in \mathbb{N}) \). Let \( X_k = X(T_k) \), the state reached after the \( k \)th transition. Let \( S_k \) denote the set of absorbing states. Define the kernel, the transition probability matrix, and the initial
The stochastic process \( \{X(t), t \geq 0\} \) defined in the previous section is a semi-Markov reward process, since a reward rate is associated with each state of the semi-Markov process. A method of computing the distribution of accumulated reward until absorption when \( \{X(t), t \geq 0\} \) is a Markov chain was given in [1], where transient states were restricted to strictly positive reward rates. More recently, in [9], this restriction is removed. A similar approach can be used in the semi-Markov setting. Beaudry’s idea in [1] is to define a new Markov chain \( \{\tilde{X}(t), x \geq 0\} \) by dividing the transition rates out of state \( i \) by \( r_i \), so that the distribution of time to absorption of the new process is the same as that of \( Y(\infty) \) in the original process. This is true because the transformation takes into account the earned reward in the “time” index of the stochastic process. The sojourn in state \( i \) is slowed down or accelerated according to the value of \( r_i \), being larger or smaller than 1. Then, for state \( i \), sojourn times of length \( \tau \) in \( \{X(t), t \geq 0\} \) and of length \( \tau r_i \) in \( \{\tilde{X}(x), x \geq 0\} \) are equivalent. Another way of thinking about this transformation is a substitution of the index variable, from time to reward: a sojourn of \( x \) time units corresponds to a gain of \( x r_i \) reward units in state \( i \) (and a gain of \( x \) reward units in state \( i \) corresponds to a sojourn of \( x/r_i \) time units). Fig. 2 illustrates this correspondence.

Following the example of Fig. 1, the sojourn time in state up is left unchanged and so is the one in state down (which is irrelevant since state down is absorbing), but the sojourn time in state recover is changed. The transition toward state down now has rate \( \lambda/\gamma \) and a successful recovery takes \( T \gamma \) “time” (or, better, accrues \( T \gamma \) reward).

There is a difficulty with the above approach: the reward rate of a transient state could be zero. No reward is earned during the sojourn in such a state; however, the system will eventually move either to another transient state, possibly with positive reward rate, or to an absorbing state.

Intuitively, zero reward rates correspond to instantaneous transitions in \( \{\tilde{X}(x), x \geq 0\} \), since \( \{\tilde{X}(x), x \geq 0\} \) spends no time \( \{X(t), t \geq 0\} \) gains no reward) in transient states with zero reward rate. Note that it is possible to increase the reward rate in each state by a positive quantity \( b \) (e.g., 1) so that all the reward rates become strictly positive. The follow-
In Fig. 1, state recover becomes vanishing if $\gamma$ is 0. This would amount to saying that no useful work is performed when a recovery is under way, a reasonable assumption.

IV. THE DEFINITION OF THE NEW PROCESS

In this section, a new semi-Markov process $\{\hat{X}(x), x \geq 0\}$ is defined, whose state space is a subset of $S$ and whose time to absorption has the same distribution as $Y(\infty)$. This is done by eliminating the vanishing states and by scaling the sojourn time in each state.

Let $S_0$ be the set of transient states with zero reward rates, $S_T$ be the set of transient states with positive reward rates, and define $S_i = S_T \cup S_A = S \setminus S_0$, so that $S = S_0 \cup S_T \cup S_A = S_0 \cup S_i$. If the states are reordered according to these subsets, the one-step transition probability matrix $P$ and the initial probability vector $\pi$ can be partitioned as

$$P = \begin{bmatrix} P^{[00]} & P^{[01]} \\ P^{[10]} & P^{[11]} \end{bmatrix}, \quad \pi = \begin{bmatrix} \pi^{[0]} \\ \pi^{[1]} \end{bmatrix} = \begin{bmatrix} \pi^{[0]} \\ \pi^{[1]} \\ \pi^{[1]} \end{bmatrix}.$$

The quantity

$$M = \begin{bmatrix} M_{i,j} \end{bmatrix} = (I - P^{[00]})^{-1} P^{[01]}$$

will be often used in the following derivations. $(I - P^{[00]})^{-1}$ is the fundamental matrix of the discrete-time Markov chain $\{X_k, k \in \mathbb{N}\}$ when all the states in $S_T$ are considered absorbing. Hence, the entries of $M$ satisfy [13]

$$M_{i,j} = \sum_{k=1}^{\infty} \Pr\{X_k = j, \forall n < k, X_n \in S_0 | X_0 = i\},$$

that is, $M_{i,j}$ is the probability that the first state visited in $S_i$ will be $j$, after an arbitrary number of transitions within $S_0$, given that the process $\{X_k, k \in \mathbb{N}\}$ is in state $i \in S_0$.

The state space of the new process $\hat{X}(x), x \geq 0$ is $\hat{S} = S_i$. Let $\hat{T}_i$ be the "time" of the $i$th transition for $\hat{X}(x), x \geq 0$ and let $\hat{T}_0 = 0, \hat{T}_i = \hat{T}_{i+1} - \hat{T}_i$, and $\hat{X}_i = \hat{X}(\hat{T}_i)$. The kernel

$$\hat{Q}(x) = [\hat{Q}_{i,j}(x)] = \Pr\{\hat{X}_{k+1} = j, \forall n \leq k, \hat{X}_k \leq x | \hat{X}_k = i\}$$

is given by

$$\hat{Q}_{i,j}(x) = \begin{cases} Q_{i,j}(x) \left(1 \over r_j\right) + \sum_{l \in S_0} Q_{l,i} \left(x \over r_l\right) M_{l,j} & \text{if } i \in S_T \\ Q_{i,j}(x) & \text{if } i \in S_A \end{cases}.$$

If state $i \in S_T$ does not reach any state in $S_0$, this transformation preserves the type of the $Q_{i,j}(\cdot)$ distribution, it merely stretches or compresses its shape on the horizontal axis, from "time" to "reward." If states in $S_0$ are indeed reached,
\( \hat{Q}_i(j) \) is a mixture distribution, possibly completely different from \( Q_i(j) \), which is just a component in the mixture \( \{Q_i(j)\} \) could even be identically equal 0.

The matrix
\[
\hat{P} = \hat{Q}(\alpha) = \begin{bmatrix} \hat{P}_{i,j} \\ \end{bmatrix} = \begin{bmatrix} \hat{P}^{(1)}(\alpha) \\ \hat{P}^{(4)}(\alpha) \\ \end{bmatrix}
\]
is related to \( P \) by the following relation:
\[
\hat{P} = P(\alpha) + P(\alpha)M = \begin{bmatrix} \hat{P}^{(1)}(\alpha) \\ \hat{P}^{(4)}(\alpha) \\ \end{bmatrix} + \begin{bmatrix} P(\alpha)^{10}P(\alpha) \\ P(\alpha)^{100}P(\alpha) \\ \end{bmatrix} + \cdots \quad (1)
\]
\( \hat{P} \) defines the DTMC obtained from \( \{X_k, k \in \mathbb{S}\} \) when only transitions into states of \( S \) are considered:
\[
\forall i \in \mathbb{S}, j \in \mathbb{S},
\hat{P}_{i,j} = \sum_{k=1}^{\infty} \Pr \{ X_k = j, \forall n 0 < n < k | X_n = \mathbb{S}_0 | X_0 = i \}.
\]
That is, a single step from \( i \in \mathbb{S} \) to \( j \in \mathbb{S} \) in \( \{X_k, k \in \mathbb{S}\} \) can be seen as a sequence of steps in \( \{X_k, k \in \mathbb{S}\} \), starting from \( i \), ending in \( j \), and such that the visited states, apart from the first and the last, belong to \( \mathbb{S}_0 \).

The initial probability vector
\[
\tilde{\pi} = \begin{bmatrix} \tilde{\pi}_i \\ \end{bmatrix} = \begin{bmatrix} \Pr \{ \hat{X}(0) = i \} \\ \end{bmatrix}
\]
of the transformed process is given by
\[
\tilde{\pi} = \pi^{(1)} + \pi^{(4)}M = \begin{bmatrix} \tilde{\pi}^{(1)} \\ \tilde{\pi}^{(4)} \\ \end{bmatrix}
\]
\[
= \begin{bmatrix} \pi^{(1)} + \pi^{(4)}M^{(1)} \\ \pi^{(4)} + \pi^{(4)}M^{(4)} \\ \end{bmatrix}. \quad (2)
\]
That is, the probability that the initial state for \( \{X_k, k \in \mathbb{S}\} \) is \( i \in \mathbb{S}_1 \) is computed as the sum of the probability that the initial state for \( \{X_k, k \in \mathbb{S}\} \) is \( i \) plus the probability that the initial state for \( \{X_k, k \in \mathbb{S}\} \) belongs to \( \mathbb{S}_0 \) but the first state visited in \( \mathbb{S}_1 \), after an arbitrary number of transitions in \( \mathbb{S}_0 \), is \( i \).

Define \( N(n) \) and \( \hat{N}(n) \) as the number of states in \( \mathbb{S}_T \) visited in the first \( n \) steps by \( \{X_k, k \in \mathbb{S}\} \) and \( \{X_k, k \in \mathbb{S}\} \), respectively, including the initial state. Using the definition \( I(n) = 1 \) if event \( \alpha \) is true, \( I(n) = 0 \) otherwise:
\[
N(n) = \sum_{k=0}^{n} I(X_k \in \mathbb{S}_T), \quad \hat{N}(n) = \sum_{k=0}^{n} I(X_k \in \mathbb{S}_T).
\]
\( N(\infty) \) and \( \hat{N}(\infty) \) are the number of states in \( \mathbb{S}_T \) visited before absorption by the two processes, respectively, including the initial state. We then obtain the following results:
\[
\Pr \{ N(\infty) = 0 | \forall n \geq 1 \} = \prod_{j=1}^{\mathbb{S}_T} \Pr \{ X_j \in \mathbb{S}_A \} = 1 - \prod_{j=1}^{\mathbb{S}_T} \tilde{\pi}_j = \tilde{\pi}_1^{\mathbb{S}_T - 1}
\]
\[
\Pr \{ N(\infty) = n | \forall n \geq 1 \} = \tilde{\pi}_1^{\mathbb{S}_T - 1} \hat{P}^{(1)}(\alpha)^{n-1} \tilde{\pi}_1^{(1)}
\]
where \( \mathbb{I} \) is the column vector with all entries equal to 1. The following results are proved in [12]:
\[
\forall i \in \mathbb{S}, \forall n \geq 0, \quad \Pr \{ N(\infty) = n | X_0 = i \} = \Pr \{ \hat{N}(\infty) = n | X_0 = i \}
\]
Thus, the system stays in state $up$ for a number of periods, until it jumps to state $down$, where it gets absorbed. The length of each period is exponentially distributed with parameter $\lambda$ (the semi-Markov process is independent, since the distribution of the period does not depend on whether the destination state is $up$ or $down$) and the number of periods up to absorption has a geometric distribution with parameter $(1 - c)\lambda$. The sum of the lengths of all the periods is exponentially distributed with parameter $\lambda(1 - c)$, so the process is equivalent, as far as absorption time is concerned, to a Markov chain with state space $\{up, down\}$ where the transition from state $up$ to state $down$ has rate $(1 - c)\lambda$ and where state $down$ is absorbing (see Fig. 3). Then

$$Pr \{ Y(\infty) \leq x \} = Pr \{ \hat{X}(x) = down \} = 1 - e^{-\lambda(1 - c)x}.$$  

V. ELIMINATION OF THE VANISHING STATES AND COMPUTATION OF $C(x)$

The first part of this section discusses the cost of performing the elimination of the vanishing states when the state space $S$ is finite. Various methods to compute the functions $\hat{F}(x)$, needed to obtain $C(x)$ using Theorem 1, are discussed at the end of the section.

In the GSPN, the presence of a recurrent subset of vanishing states implies that the process becomes stochastically discontinuous. By contrast, during the transformation of the process $\{X(t), t \geq 0\}$ into $\{\hat{X}(x), x \geq 0\}$, this situation cannot arise, since recurrent subsets are neither present initially nor generated during the construction (slowing down or accelerating the sojourn time in a state does not change the relative transition probabilities from a state to the other states). The absence of recurrent subset of vanishing states guarantees the existence of the inverse $(I - P^{[00]})^{-1}$. Define $G$ as the subgraph obtained from the state transition diagram when only the vanishing states are considered. $G$ can contain self-loops and transient loops.

A self-loop is a transition from a vanishing state to itself ($P_{ii} \neq 0$ for some $i \in S_0$) and it can be simply ignored: $P_{ii}$ is set to 0 and the nonzero entries in row $i$ for $P^{[00]}$ and $P^{[01]}$ are normalized (the new value of $P_{ij}$ is $P_{ij}/(1 - P_{ii})$, for $j \in S$, $j \neq i$). Since state $i$ is vanishing, this modification does not change the final result (only the final destination is relevant, not the number of jumps into state $i$ before finally leaving it). The diagonal entries of $P^{[00]}$ from now on can be assumed to be zero.

A transient loop is a set of mutually reachable transient states, a strongly connected component in $G$. The presence of transient loops is not a problem in principle, but it can negatively affect the performance of the algorithm for the elimination of the vanishing states. Let $n_0$ and $n_1$ be the number of states in $S_0$ and $S_1$, respectively, let $x^{[ij]}$ be the number of nonzero entries in $P^{[ij]}$, $i, j \in \{0, 1\}$, and assume that a sparse storage technique is used (for simplicity of notation assume $x^{[ij]} > x^{[i'j']} > 0$ for both $n_0$ and $n_1$).

If no transient loop is present, a partial ordering is implicitly defined on the states of $S_0$ (nodes of $G$): from it, a compatible total ordering can be found in time $O(x^{[00]})$, by means of topological sort [14], so that, if state $i$ follows state $j$, then $j$ cannot be reached from $i$ in $G$.

If the total ordering is applied to $S_0$, $P^{[00]}$ becomes strictly upper triangular:

$$(p^{[00]})^n_0 = 0 \text{ and } (I - P^{[00]})^{-1} = \sum_{k=0}^{n_0-1} (P^{[00]})^k$$

hence $(I - P^{[00]})^{-1}$ is upper triangular with diagonal entries equal to 1. Although $(I - P^{[00]})^{-1}$ is never explicitly computed in practice, the triangular nature of $P^{[00]}$ allows great efficiency; the required steps are listed in the following, together with the asymptotic execution time requirements:

- solve for $M$ in $(I + P^{[00]})M = P^{[01]} = n_1O(x^{[00]})$
- compute $W$ using $W = P^{[00]}M = n_1O(x^{[00]})$
- compute $\hat{P} = P^{[11]} + W = O(n_0n_1 + x^{[11]})$

The total time requirement to compute $\hat{P}$ is then (given the assumption $x^{[10]} \geq n_1$ and $x^{[11]} \geq n_1$)

$$O((x^{[00]} + x^{[10]})n_1).$$  

If transient loops are present, the worst case bound on the time requirement is higher, an LU factorization is involved, and the obtained matrices $L$ and $U$ can have a large fill-in, possibly $O(n_0^3)$:

- factorize $P^{[00]}$ into $L$ and $U = O(n_0^3)$
- solve for $M$ in $(I + P^{[00]})M = P^{[01]}$ using $L$ and $U = n_1O(n_0^2)$
- compute $W$ using $W = P^{[00]}M = n_1O(x^{[00]})$
- compute $\hat{P}$ using $\hat{P} = P^{[11]} + W = O(n_0n_1 + x^{[11]})$

So the total time requirement is

$$O(n_0^3 + (x^{[10]} + n_0^2)n_1).$$  

Comparing (3) to (4), the term $O(x^{[00]}n_1)$ is substituted by the term $O(n_0^3 + n_0^2n_1)$, the two terms are equivalent only if $x^{[00]} = O(n_0^3)$ and if $n_0 = O(n_1)$.

Finally, if transient loops are present, but their size can be considered "small" with respect to $n_0$, the topological sort idea can still be used to improve the performance. A partial ordering is now implicitly defined among the transient loops (strongly connected components in $G$), not among the states (nodes in $G$), if each state not in a transient loop is considered as a transient loop in itself. The strongly connected components can be found using a modification of the depth-first search algorithm [15], so the number of operations to find a total ordering on the states of $S_0$ is still $O(x^{[00]})$ as before. The total ordering defined this way is such that, if state $i$ follows state $j$, then either state $j$ cannot be reached from state $i$ in $G$, or states $i$ and $j$ are mutually reachable in $G$ (they are in the same strongly connected component).

Assume for simplicity that there are exactly $n_0/k$ transient
loops of size \( k \) each; matrix \( P^{(0)} \) can then be put in upper block triangular form, where each block \( P^{(0)}_{i,j} \), \( i \leq j \) is of size \( k \times k \). Also, denote with \( M_{i,j} \) and \( F^{(0)}_{i,j} \) the \( k \times 1 \) blocks forming \( M \) and \( P^{(0)} \), respectively, (a small abuse of notation, since we already used the same symbols to indicate elements of a matrix, not blocks). Each diagonal block must then be \( LU \) factorized, introducing fill-in only in the block itself, then a block-oriented back-substitution procedure can be used to compute \( M, k \) rows at a time, without any fill-in [16, p. 161]. The required steps are:

- for \( i = 1 \) upto \( n_0/k \) do
  - factorize \( P^{(0)}_{i,i} \) into \( L_i \) and \( U_i \)
- for \( i = n_0/k \) downto 1 do
  - for \( j = 1 \) upto \( n_1 \) do
    - compute \( A \) using \( A = \sum_{d=1}^k P^{(0)}_{i,d} M_{i,j} \)
    - solve for \( M_{i,j} \) in \((I - P^{(0)}_{i,i}) M_{i,j} = P^{(0)}_{i,j} - A \) using \( L_i \) and \( U_i \)
- compute \( W \) using \( W = P^{(0)} M \)
- compute \( \bar{P} \) using \( \bar{P} = P^{(1)} + W \)

The expression \( O(\alpha) \) depends on the number of nonzero entries in each \( P^{(0)}_{i,i} \), but, summing over all the values of \( i \) and \( j \) it gives \( O(\alpha \times n) \), to be summed over all \( n_1 \) values of \( j \), so the total time requirement is

\[
O(k^2 n_0 + k n_0 n_1 + (x^{(0)} + x^{(10)})) \quad (5)
\]

If \( k = O(n_0) \), there is a small (constant) number of large loops: expressions (5) and (3) coincide. If \( k = O(1) \), there is a large number of small (constant size) loops: expressions (5) and (3) coincide. Finally, if \( k \) is a sublinear function of \( n_0 \) for example \( k = O(\sqrt{n_0}) \), the dominant term in (5) will be determined by the relative value of \( n_0, n_1, x^{(0)}, \) and \( x^{(10)} \), but the running time of the algorithm will be in any case improved by exploiting this ordering, especially considering that the determination of the ordering itself is an inexpensive operation in comparison to the other steps.

So far the discussion has been on the computation of \( \bar{P} \). It should be clear how the same bounds hold when considering the computation of \( \bar{Q} \), although, in the analogous expression \( \bar{Q}(x) = Q^{(11)}(x) + Q^{(10)}(x) \) (where \( Q^{(11)} \) and \( Q^{(10)} \) have the obvious meaning) in general involves sums of real functions, not of real numbers. The bounds are nonetheless valid even in this case, for example the reader could imagine that they represent the time to output \( \bar{Q} \) in symbolic sparse format.

The elimination of vanishing states helps the subsequent numerical solution both by reducing the overall size of the problem (especially when \( n_0 \) is large in comparison to \( n_1 \) and the fill-in introduced in \( \bar{P} \) is small) and by avoiding the introduction of distributions having extremely small averages, to approximate vanishing states. As it will be pointed out in the section on the examples, the existence of "fast" states puts an excessive burden on the numerical solution. Furthermore, the incorporation of the reward information into the rate allows us to reduce the problem of computing the accumulated reward to that of computing the time to absorption. Once matrix \( \bar{Q} \) has been obtained, the problem of computing \( C(x) \) (by computing \( F_j(x) \) first) for the relevant value(s) of \( x \) or, even better, as a function of \( x \), remains. Unfortunately this problem is generally harder than the elimination of the vanishing states itself. Viable methods include numerical integration of a system of coupled Volterra equations [17], use of Laplace–Stieltjes transforms, and, if all \( Q_{ji}(\cdot) \) are phase-type distributions, transformation of the semi-Markov process into a CTMC to be solved using appropriate numerical methods. Regardless of the method employed, additional advantages can be obtained by lumping all the absorbing states together, an operation always advisable

\[
\begin{align*}
\frac{n_0}{k} & \ldots \\
O(k^2) & \\
n_0 & \ldots \\
O(\alpha) & \\
n_1 & \ldots \\
O(k^2) & \\
n_0 & O(x^{(10)}) \\
O(n_0 n_1 + x^{(11)}) & \\
\end{align*}
\]

unless detailed information on the particular failure state (reason or "mode" of the failure) is required. In the following, it is assumed that this lumping is performed and the absorbing macro-state is \( L \).

For simplicity, we will drop the "\( \cdot \)" notation, since the problem is now the computation of the time to absorption for a semi-Markov process and the fact that this process was obtained from yet another process is irrelevant. The goal is the computation of the time-dependent state probability for state \( L \) (or the first-passage time distribution to state \( L \)), conditioned on the initial state \( i \), for each state \( i \in S \setminus S_A \) such that \( \pi_i > 0 \).

Conceptually only \( O(n_1) \) distribution functions are needed, or even less; for example, only one function is needed if the initial state is certain, that is, \( \exists i \in S \setminus S_A \) such that \( \pi_i = 1 \). Unfortunately, in general, additional information must be computed to obtain the required distribution(s), so that the solution can be very expensive in terms of execution time, memory requirements, or both.

If each distribution is approximated using a phase-type expansion, a Markov chain is obtained. A survey of numerical methods to solve for the time-dependent state probabilities is presented in [18]. There is a tradeoff between the number of stages used in the expansions and how well the distributions are fitted, but the first and main tradeoff is between size and simplicity: the state space is increased to allow a (simpler) Markovian analysis.

A direct approach for the transient analysis of the semi-Markov process involves solving the system of integral equations

\[
\forall i \in S, \forall j \in S, \quad \Pi_{i,j}(x_i) = \Pi_{i,-j}(1 - H_i(x_j)) + \sum_{h \in S} \int_0^x \Pi_{i,j}(x - u) dQ_{i,h}(u)
\]

where

\[
\Pi(x) = [\Pi_{i,j}(x)] = [\Pr \{ X(x) = j \mid X_0 = i \}]
\]

is the matrix expressing the time-dependent probability of
being in state $j$ conditioned on $i$ being the initial state and

$$H(x) = \left[ H_i(x) = \Pr \{ V_k \leq x \mid X_k = i \} \right] = \sum_{j \in S} Q_{i,j}(x)$$

is the vector of the holding time distributions in each state, independently of the destination state. The conditional distribution of time to absorption is then given by

$$F_j(x) = \left[ \sum_{i \in S} Q_{i,j}(x) \right]$$

Finally, numerical methods for the above integration may be found in [17]. Laplace-Stieltjes transforms ($\mathcal{LST}$) can also be employed. Define the LST of $Q_{i,j}(x)$ by

$$Q_{i,j}(s) = \int_0^\infty e^{-sx} dQ_{i,j}(x).$$

Partition $Q(x)$ as

$$Q(x) = \begin{cases} U, & \text{to state } U, \\ \text{to state } D, & (1 \leq i \leq k) \end{cases}$$

and from state $D, \text{ to state } U,+,$

$$Q_{i,j}(s) = \begin{cases} \text{distribution of } i\text{'th up period} \end{cases}$$

$$Q_{i,j}(s) = \begin{cases} \text{distribution of } i\text{'th repair period}. \end{cases}$$

If the reward rate in state $U$, is $r > 0$ and in state $D$, is $0 \leq i \leq k$, then $D, \text{ to state } U,+,$

$$Q_{i,j}(s) = \begin{cases} \text{vanishing states.} \end{cases}$$

The distribution of the accumulated reward up to absorption is

$$C_U(x) = \left( \hat{\phi}_1 * \cdots * \hat{\phi}_k \right)(x)$$

where $*$ denotes the convolution operator. The expression for the LST transform is simpler:

$$\tilde{C}_U(s) = \int_0^\infty e^{-sx} dC_U(x) = \sum_{i=1}^k \hat{\phi}_i \left( \frac{s}{r_i} \right)$$

Let $\tilde{Q}(s)$ be the matrix of the LST of $Q(x)$ and partition it accordingly:

$$\tilde{Q}(s) = \begin{bmatrix} \tilde{Q}_{TTI}(s) & \tilde{Q}_{TLI}(s) \\ 0 & \tilde{Q}_{L,L}(s) \end{bmatrix}$$

The vector $\tilde{F}(s)$, whose elements are the LST of $F_i(x)$, for $i \in S \setminus S_A$, can be computed as the solution of (see [19])

$$\left( I - \tilde{Q}_{TTI}(s) \right) \tilde{F}(s) = \tilde{Q}_{TLI}(s)$$

as expected, $\tilde{F}(s)$ is independent of $\tilde{Q}_{L,L}(s)$. In order to obtain $F(t)$, we use a two-phase approach. First we solve the matrix equation (6) in $s$-domain and subsequently use numerical inversion of the vector $\tilde{F}(s)$ into $F(t)$; see [20] for a discussion on LST inversion.

VI. EXAMPLES

Limited number of repairs [3]: Consider a system experiencing failures. When a failure occurs, the system is repaired, but only up to $k - 1$ times. At the $k$th failure, the system will not be repaired any more; it will remain in the failed state (Fig. 4). The state space is $S = \{ U_1, D_1, U_2, \ldots, U_k, D_k \}$, there is only one absorbing state, $S_A = \{ D_k \}$, and $\pi = [1, 0, 0, \ldots, 0]$. Transitions are from state $U_i$ to state $D_i$ (1 \leq i \leq k) and from state $D_i$ to state $U_{i+1}$ (1 \leq i \leq k-1).

$$Q_{U_i,D_i}(t) = \phi_i(t) \quad (\text{distribution of } i\text{'th up period})$$

$$Q_{D_i,U_{i+1}}(t) = \rho_i(t) \quad (\text{distribution of } i\text{'th repair period})$$

If the reward rate in state $U_i$ is $r_i > 0$ and in state $D_i$ is $0 \leq i \leq k$, then $D_i, \text{ to state } U_{i+1}$ are vanishing states. Applying the transformations, $S = \{ U_1, U_2, \ldots, U_k, D_k \}$ and

$$\tilde{Q}_{U_i,D_i}(x) = \hat{\phi}_i \left( \frac{x}{r_i} \right) \quad (1 \leq i \leq k - 1)$$

$$\tilde{Q}_{U_i,D_i}(x) = \hat{\phi}_k \left( \frac{x}{r_k} \right).$$

The distribution of the accumulated reward up to absorption is

$$C_{U_i}(x) = \left( \hat{\phi}_1 * \cdots * \hat{\phi}_k \right)(x)$$

where $*$ denotes the convolution operator. The expression for the LST transform is simpler:

$$\tilde{C}_{U_i}(s) = \int_0^\infty e^{-sx} dC_{U_i}(x) = \sum_{i=1}^k \hat{\phi}_i \left( \frac{s}{r_i} \right)$$

Note that if $r_i = 1$ for all the nonabsorbing states, then $C_{U_i}(x)$ is simply the distribution of total uptime until failure. Interesting cases are:

- $Q_{U_i,D_i}(\cdot) \sim \text{CONST}(T)$, for $1 \leq i \leq k$. If $r_i = 1$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{CONST}(kT)$; if $r_i = k + 1 - i$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{CONST}(\frac{T}{2})$. Note the independence of the result from the distributions $Q_{D_i,U_{i+1}}(\cdot)$. Independence from the sojourn time distribution of the states with zero reward is a general property of this method, further simplifying the analysis.

- $Q_{U_i,D_i}(\cdot) \sim \text{EXPO}(\lambda_i)$, for $1 \leq i \leq k$. If $r_i = 1$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{HYPO}(\frac{\lambda_i}{r_i})$. Assuming that all the fractions $\lambda_i/r_i$ are different.

- $Q_{U_i,D_i}(\cdot) \sim \text{NORMAL}(\mu, \sigma^2)$, for $1 \leq i \leq k$. If $r_i = 1$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{NORMAL}(\mu, k\sigma^2)$; if $r_i = k + 1 - i$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{NORMAL}(\mu, \frac{k(k+1)}{2}\sigma^2)$.
case of a uniprocessor system, which can be either up or down.

- $Q_{U_i,D_i}(\cdot) \sim \text{EXPO}(\lambda)$ and $T_i = 1$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{ERL}(\lambda, k)$. This could correspond to the simple case of a uniprocessor system, which can either be up or down.

- $Q_{U_i,D_i}(\cdot) \sim \text{EXPO}(k + 1 - i)\lambda$ and $T_i = k + 1 - i$, for $1 \leq i \leq k$, then $C_{U_i}(\cdot) \sim \text{ERL}(\lambda, k)$, as before. Interestingly enough, since this could correspond, for example, to a $k$-processor system without repair ($\rho_i$ is related to the time to reconfigure the system after the $i$th failure, not to the time to repair it).

**Cumulative computation of an $M/M/2/K$ system before failure:** Consider a queueing system with Poisson arrivals, two servers, exponentially distributed service times, and finite capacity $K$. Let $1/\lambda$ and $1/\mu$ be the average interarrival and service times, respectively. Lifetime and repair time of each server are independent and exponentially distributed with average $1/\phi$ and $1/\rho$, respectively. For example, the reader might imagine a system performing a certain set of tasks using two processors. Tasks are sorted according to their priority and low priority ones may be discarded if there is no waiting room available. When only one processor is available, tasks are more likely to be discarded (especially if with lower priority), but the system is still able to survive. If the second processor fails while the first is being repaired, no task can be performed and the system is considered down. The measure of interest is the cumulative computation performed up to the first failure of the system.

The state of the system is denoted by a pair $(i,j)$ where $i = 0, 1, 2$ is the number of nonfailed servers, and $j$ is the number of jobs in the system ($j$ is absent if $i = 0$). The system is initially in state $(2,0)$ with probability 1. The cumulative computation for this system is represented by $Y(\infty)$ for the CTMC in Fig. 5 when the reward rates are defined as

- $r_{(2,0)} = 0$, $r_{(2,1)} = \mu$, $\forall j \geq 2$, $r_{(2,j)} = 2\mu$,
- $r_{(1,0)} = 0$, $\forall j \geq 1$, $r_{(1,j)} = \mu$,
- $r_{(0)} = 0$.

The SHARPE package [21] was used to analyze $C(x)$ for the system, both with an "approximate" approach (the reward for states $(2, 0)$ and $(1, 0)$ is defined as 0.000001) and with the "exact" method proposed in this paper (states $(2,0)$ and $(1,0)$ are eliminated because they are vanishing). The transformed CTMC is in Fig. 6. The initial probability vector for the transformed CTMC, given the choice $\pi_{(2,0)} = 1$, is

$$\tilde{\pi}_{(2,1)} = \frac{\lambda(\phi + \lambda + \rho)}{\rho \lambda + (2\phi + \lambda)(\phi + \lambda)}.$$  

$$\tilde{\pi}_{(1,1)} = \frac{2\lambda \phi}{\rho \lambda + (2\phi + \lambda)(\phi + \lambda)}.$$  

$$\tilde{\pi}_{(0)} = \frac{2\phi^2}{\rho \lambda + (2\phi + \lambda)(\phi + \lambda)}.$$  

The values assigned to the parameters are $K = 10$, $\mu = 1.0$, $\lambda = 0.6$, $\phi = 0.00001$, and $\rho = 0.005$. The distribution function $C(x)$ is plotted in Fig. 7. The difference between the exact and the approximate values of $C(x)$ is plotted in Fig. 8 (note the logarithmic scale on the $x$ axis). As expected, the approximate solution is slightly more optimistic than the exact one (shown in figure), since it assumes that some computation is performed in the two states that would otherwise be vanishing. Unfortunately, the presence of extremely small reward rates (six orders of magnitude smaller than the reward rates for the other states) causes numerical problems in SHARPE. The values of $C(x)$ obtained with the
CIARDO

while the state was fail, an off-line system: the holding time distributions for the transient states are obtained using the exact method. When the system enters state \( y \) to state 0 means that an observation period ended recovery can be attempted on-line. If they increasing the state space, but obtaining a CTMC as a result.

An observation of the transition and reward rates suggests that this model has two types of vanishing states. The reward rates for states \( h \) and \( s \) are nearly 0, they would be specified as 0 if SHARPE had a provision for zero reward transient states (the system gains no reward in them). The remaining states have a nonnegligible reward, but the holding time in state 0 is negligible in comparison to the one in the other states. State 0 is indeed introduced as a device to represent the interruption of observation epochs; it simply provides a probabilistic switch from the states having a transition to it \((1, 2, 4, 6, 7)\) to the states reached from it \((1, 2, 4, \ldots, 7)\). State 0 is vanishing because the original process does not spend time in it (state 0 is really an abstraction), not because its reward rate is zero, but the final effect is the same: ideally the process does not gain any reward in states 0, \( h \), or \( s \).

Using the notation previously introduced, \( \hat{S} = \{1, 2, 3, 4, 5, 6, 7, c, e, d, m, o\} \). Since the original process is independent, also the new process is independent, and, in addition, the holding time distributions for its states are obtained by stretching the corresponding original distributions on the horizontal axis (no mixture distribution is generated):

\[
\forall i \in \hat{S} \setminus \{0\}, \quad \hat{H}_i(x) = H_i\left(\frac{x}{r_y}\right) \quad \text{and} \quad \hat{H}_0(x) = H_0(x).
\]

The transition probability matrix \( \hat{P} \) (Fig. 11) completes the specification of the new process. The matrix \( \hat{P} \) was obtained applying (1) with the help of the MATLAB package [22].

The analysis of the new process is also performed using SHARPE, expanding the stage-type distributions, as was done for the original process, but now matrix \( \hat{P} \) contains nonzero diagonal entries and this leads, in the expanded CTMC, to arcs having coinciding source and destination. When this happens, the correct CTMC is obtained by simply ignoring these arcs (the holding time in the state or stage is increased).

The system was modeled with the original description method using several sets of data, differing in the choice of \( H_0, r_y, \) and \( r_x \), to observe how the results were affected (all the other specifications were as in Fig. 9). The choices were \( H_0 \sim \text{EXP}(10^n) \) and \( r_y = r_x = 10^{-n} \), for \( n = 1, 2, 4, 6, \) and fitted to the stage-type distributions shown in Fig. 9 (the holding time is independent of the next state reached):

\[
\forall i \in S, \quad H_i(t) = \mathbb{P}\{V_k \leq t | X_k = i\}.
\]

This example was also analyzed using SHARPE. Each stage-type distribution was expanded, increasing the state space, and the holding time distributions for the transient states are

\[
\hat{H}_i(x) = H_i\left(\frac{x}{r_y}\right) \quad \text{and} \quad \hat{H}_0(x) = H_0(x).
\]

\[
\mathbb{P}\{V_k \leq t | X_k = i\}.
\]

The transition probability matrix \( P \) resulting from the observations is given in Fig. 10. Note that state 0 is the only absorbing state. Assume \( \pi_0 = 1 \). The following reward rates for the states are defined:

\[
r_0 = \cdots = r_2 = 1; \quad r_c = 0.9946; \quad r_6 = 0.2736;
\]

\[
r_d = 0.5708; \quad r_m = 0.2777; \quad r_h = r_y = 0.0001; \quad r_o = 0.
\]

The state space is then \( \{0, 1, 2, 3, 4, 5, 6, 7, \{\}, \{c\}, \{e\}, \{d\}, \{m\}, \{o\}\} \). Since the original process is independent, also the new process is independent, and, in addition, the holding time distributions for its states are obtained by stretching the corresponding original distributions on the horizontal axis (no mixture distribution is generated):

\[
\forall i \in \hat{S} \setminus \{0\}, \quad \hat{H}_i(x) = H_i\left(\frac{x}{r_y}\right) \quad \text{and} \quad \hat{H}_0(x) = H_0(x).
\]

\[
\mathbb{P}\{V_k \leq t | X_k = i\}.
\]

The transition probability matrix \( \hat{P} \) (Fig. 11) completes the specification of the new process. The matrix \( \hat{P} \) was obtained applying (1) with the help of the MATLAB package [22].

The analysis of the new process is also performed using SHARPE, expanding the stage-type distributions, as was done for the original process, but now matrix \( \hat{P} \) contains nonzero diagonal entries and this leads, in the expanded CTMC, to arcs having coinciding source and destination. When this happens, the correct CTMC is obtained by simply ignoring these arcs (the holding time in the state or stage is increased).

The system was modeled with the original description method using several sets of data, differing in the choice of \( H_0, r_y, \) and \( r_x \), to observe how the results were affected (all the other specifications were as in Fig. 9). The choices were \( H_0 \sim \text{EXP}(10^n) \) and \( r_y = r_x = 10^{-n} \), for \( n = 1, 2, 4, 6, \) and fitted to the stage-type distributions shown in Fig. 9 (the holding time is independent of the next state reached):

\[
\forall i \in S, \quad H_i(t) = \mathbb{P}\{V_k \leq t | X_k = i\}.
\]

The transition probability matrix \( P \) resulting from the observations is given in Fig. 10. Note that state 0 is the only absorbing state. Assume \( \pi_0 = 1 \). The following reward rates for the states are defined:

\[
r_0 = \cdots = r_2 = 1; \quad r_c = 0.9946; \quad r_6 = 0.2736;
\]

\[
r_d = 0.5708; \quad r_m = 0.2777; \quad r_h = r_y = 0.0001; \quad r_o = 0.
\]

This example was also analyzed using SHARPE. Each stage-type distribution was expanded, increasing the state space, but obtaining a CTMC as a result.

An observation of the transition and reward rates suggests that this model has two types of vanishing states. The reward rates for states \( h \) and \( s \) are nearly 0, they would be specified as 0 if SHARPE had a provision for zero reward transient states (the system gains no reward in them). The remaining states have a nonnegligible reward, but the holding time in state 0 is negligible in comparison to the one in the other states. State 0 is indeed introduced as a device to represent the interruption of observation epochs; it simply provides a probabilistic switch from the states having a transition to it \((1, \cdots, 7, h, s)\) to the states reached from it \((1, 2, 4, \cdots, 7)\). State 0 is vanishing because the original process does not spend time in it (state 0 is really an abstraction), not because its reward rate is zero, but the final effect is the same: ideally the process does not gain any reward in states 0, \( h \), or \( s \).

Using the notation previously introduced, \( \hat{S} = \{1, 2, 3, 4, 5, 6, 7, c, e, d, m, o\} \). Since the original process is independent, also the new process is independent, and, in addition, the holding time distributions for its states are obtained by stretching the corresponding original distributions on the horizontal axis (no mixture distribution is generated):

\[
\forall i \in \hat{S} \setminus \{0\}, \quad \hat{H}_i(x) = H_i\left(\frac{x}{r_y}\right) \quad \text{and} \quad \hat{H}_0(x) = H_0(x).
\]

The transition probability matrix \( \hat{P} \) (Fig. 11) completes the specification of the new process. The matrix \( \hat{P} \) was obtained applying (1) with the help of the MATLAB package [22].

The analysis of the new process is also performed using SHARPE, expanding the stage-type distributions, as was done for the original process, but now matrix \( \hat{P} \) contains nonzero diagonal entries and this leads, in the expanded CTMC, to arcs having coinciding source and destination. When this happens, the correct CTMC is obtained by simply ignoring these arcs (the holding time in the state or stage is increased).

The system was modeled with the original description method using several sets of data, differing in the choice of \( H_0, r_y, \) and \( r_x \), to observe how the results were affected (all the other specifications were as in Fig. 9). The choices were \( H_0 \sim \text{EXP}(10^n) \) and \( r_y = r_x = 10^{-n} \), for \( n = 1, 2, 4, 6, \) and

approximate method appear to be correct only up to \( x = 10^2 \), then they become unstable, getting larger than the one obtained using the exact method.

Measurement-based performability of a multiprocessor system: In [4], a performability study of an IBM 3081 system is reported. The analysis involves three aspects: performance, failure, and recovery behavior. The working states are classified into clusters, according to the utilization of resources (CPU, channel, direct access storage). States 1 through 7 correspond to the clusters. When a failure occurs in a working state, it is logged as a channel (c), software (e), or direct access storage (d) error. Furthermore, a failure affecting several components is classified as a multiple failure, or a failure affecting several components is classified as a multiple error. When the system enters state c, e, d, or m, a recovery process tries to restore the state of the system. Hardware (h) and software (s) recovery can be attempted on-line. If they fail, an off-line (o) recovery is necessary and the system is considered down. It is observed that the failure behavior is dependent on the particular working state. An extra state, 0, is added, to signify the periods of no data collection. A transition from state 0 to state x means that the state is x at the beginning of an observation period. A transition from state y to state 0 means that an observation period ended while the state was y. The state space is then \( \{0, 1, 2, 3, 4, 5, 6, 7, c, e, d, m, h, s, o\} \).

The process is assumed to be independent semi-Markovian and the holding time distributions for the transient states are
Let $C^{(n)}(x)$ be the accumulated reward up to absorption computed this way, for $n = 1, 2, 4, 6,$ and $8$.

The system was then modeled with the exact method, where states $0, h$, and $s$ are absent, obtaining a distribution of the accumulated reward consistently more pessimistic than the previous ones. The exact accumulated reward, $C^{(\infty)}(x)$, is shown in Fig. 12.

The absolute differences between $C^{(n)}(x)$ and $C^{(1)}(x)$, $C^{(4)}(x)$, and $C^{(6)}(x)$, respectively, are reported in Fig. 13 (for small values of $x$, from 10 to 100 reward units) and in Fig. 14 (for large values of $x$, up to $3 \cdot 10^5$ reward units). As expected, the differences tend to be smaller (closer to 0) as $n$ increases, but raising $n$ too much ($n \geq 6$) causes numerical problems in SHARPE. Interestingly enough,
C(\(x\)), whose computation is numerically correct, is still consistently apart from \(C(\infty)\). The cause of this behavior is the large number of visits to states 0, \(h\), and \(s\) before absorption: state 0 has an extremely short holding time, but its reward rate is 1; and states \(h\) and \(s\) have an extremely small reward rate, but their holding times are not negligible. To obtain an almost perfect match with \(C(\infty)\) without numerical problems, both the reward rate for state 0 and the sojourn times for states \(h\) and \(s\) had to be reduced. In other words, it may be difficult to obtain exact results using a “limiting” operation that will stress, if not compromise, most numerical methods: it is safer, and generally faster, to explicitly eliminate the vanishing states.

VII. OBSERVATIONS

Cumulative time spent in a subset of states up to absorption: If the reward rates for the transient states are restricted to be either 0 or 1, the reward rates for the absorbing states are 0, and the set \(B \subseteq S \setminus S_a\) is defined as \(B = \{i \in S \setminus S_a \land r_i = 1\}\), then \(C(x)\) represents the probability that no more than \(x\) time units are globally spent in the states of the subset \(B\) before absorption. By changing the set of states having reward rate 1, different transformed processes can be obtained and different aspects of the same system can be analyzed.

Extensions of the class of semi-Markov processes considered: The proposed transformation has been applied to semi-Markov processes with probability of absorption equal 1 and with state space \(S\) partitioned into two subsets, \(S_a\), composed of transient states with nonnegative reward rate, and \(S_{a+}\), composed of absorbing states with zero reward rate. A more general class of semi-Markov processes could be considered, partitioning the state space \(S\) into two subsets, \(S_{a+}\), composed of transient states with positive reward rates, and \(S_{a-}\), composed of states with zero reward rate. States in \(S \setminus S_{a-}\) may be recurrent, transient, or absorbing, as long as the states of \(S_{a-}\) are transient. It is easy to see, though, that this is a trivial generalization, where a recurrent subset of states with zero reward can be substituted for a single absorbing state with null reward rate.

Another class of semi-Markov process that could fit the proposed approach is exemplified by the process \(\{X(t), t \geq 0\}\) having \(S = \emptyset, \forall i \in \emptyset, \forall j \in \emptyset\), \(Q_{ij}(t) = \text{EXP}(2^j)\), if \(j = i + 1\), \(Q_{ii}(t) = 0\) otherwise, \(\pi_0 = 1\), a classical example of nonregular CTMC [23]. The properties of the process obtained applying the transformation will be dependent on the reward rates assigned to each state. If we let \(r_i = 1 \forall i \in \emptyset\), the transformed process \(\{\hat{X}(x), x \geq 0\}\) will be nonregular as well: its lifetime, and \(Y(\infty)\), will be finite with probability 1. If, on the other hand, we let \(r_i = 2^j \forall i \in \emptyset\), the transformed process has a sojourn time \(\sim \text{EXP}(1)\) in each state; hence, it is regular: its lifetime, and \(Y(\infty)\), will be \(\infty\) with probability 1.

Analogously, consider an initial process as before, but with \(Q_{i,i+1}(t) = \text{EXP}(1)\), an example of regular CTMC where all states are transient. If we let \(r_i = 1 \forall i \in \emptyset\), the transformed process is regular and its lifetime, and \(Y(\infty)\), are \(\infty\) with probability 1. If instead we let \(r_i = 2^{-i} \forall i \in \emptyset\), the transformed process has a sojourn time \(\sim \text{EXP}(2^j)\) in state \(i\); hence, it is irregular and its lifetime, and \(Y(\infty)\), will be finite with probability 1.

Absorbing states with positive reward rate: If a positive reward rate is assigned to an absorbing state reachable from the initial state \(i\), then \(E[Y(\infty) | X_0 = i] = +\infty\). Furthermore, if the absorbing states \(S_a\) are partitioned into \(S_{a+}\) and \(S_{a-}\), the former containing states with zero reward rate and the latter containing states with positive reward rate, then the value of

\[
\alpha_i = \text{pr} \{Y(\infty) = +\infty | X_0 = i\} = \text{pr} \{\hat{X}(\infty) \in S_{a+} | X_0 = i\} = \text{pr} \{X(\infty) \in S_{a-} | X_0 = i\}
\]

will influence the subsequent analysis. If \(\alpha_i < 1\), the conditional distribution of \(Y(\infty)\), given that the process is absorbed in a state with zero reward rate and the initial state \(i\), may be computed,

\[
\text{pr} \{Y(\infty) \leq x | X_0 = i, X(\infty) \in S_{a+}\}
\]

as well as its expectation,

\[
E[Y(\infty) | X_0 = i, X(\infty) \in S_{a+}].
\]

Once \(\alpha_i\) has been computed, the problem is reduced to the one previously discussed and it can be solved with the same technique (\(\alpha_i\) is simply the mass at \(\infty\) for a defective distribution).

Negative reward rates: A further complication arises in the presence of negative reward rates, often present, for example, when costs are considered. If some absorbing state has a negative reward rate, then \(S_a\) must be partitioned into \(S_{a+}\), \(S_{a-}\), and \(S_{a-}\), \(\alpha_i, \alpha_i, \alpha_i, \alpha_i\) can be defined as the probabilities of being absorbed in each subset, respectively. If \(\alpha_i > 0\), the conditional distribution of \(Y(\infty)\) can be defined, as done in the previous paragraph, but its computation is complicated by transient states with negative reward rate. If there is a lower bound \(-b\) to the value of these rates, the reward rate of the transient states can be shifted by adding \(b\) to them, so that they become nonnegative. In particular, if \(S\) is finite, \(b\) always exists.

As already mentioned in Section III, though, performing this shift is useful only to compute the expected value of
using the relation
\[ E[ Y(\infty) \mid X_0 = i] = E[ \Theta(\infty) \mid X_0 = i] - b \ \text{MTTF}, \]
but not to compute its distribution.

VIII. CONCLUSION

In this paper, we have presented an algorithm for the computation of the accumulated reward in a semi-Markov reward process. This is an extension of an algorithm proposed by Beaudry. The algorithm is robust, since it allows for zero reward states that are nonabsorbing. The algorithm proceeds by replacing zero reward nonabsorbing states by a probabilistic switch; thus it is related to the elimination of "vanishing" states from the reachability graph of a generalized stochastic Petri net [10] and to the elimination of fast transient states in a decomposition approach to stiff Markov chains [11]. The usefulness of the algorithm is illustrated with several examples.

APPENDIX

Proof of Lemma 1: \forall i \in S_0, \forall n \geq 0,
\[ \Pr \{ Y(\infty) \leq x, N(\infty) = n \mid X_0 = i \} = \sum_{j \in S_0 \cup S_1} P_{i,j} \Pr \{ Y(\infty) \leq x, N(\infty) = n \mid X_0 = j \} \]

\[ \Pr \{ Y(\infty) \leq x, N(\infty) = 1 \mid X_0 = i \} \]

\[ = \sum_{h \geq 1} \Pr \left\{ X_h \in S_A, \forall n \geq h \ X_h \in S_0, V_h \leq \frac{x}{r_i} \mid X_0 = i \right\} \]

\[ = \Pr \left\{ X_1 \in S_A, V_0 \leq \frac{x}{r_i} \mid X_0 = i \right\} \]

\[ + \sum_{h \geq 2} \Pr \left\{ X_h \in S_A, \forall n < h \ X_n \in S_0, V_n \leq \frac{x}{r_i} \mid X_0 = i \right\} \]

\[ = \sum_{j \in S_A} Q_{1,j} \left( \frac{x}{r_i} \right) + \sum_{k \in S_0} \sum_{h \geq 2} \Pr \left\{ X_h \in S_A, \forall n < h \ X_n \in S_0, X_1 = l, V_0 \leq \frac{x}{r_i} \mid X_0 = i \right\} \]

\[ = \sum_{j \in S_A} Q_{1,j} \left( \frac{x}{r_i} \right) + \sum_{i \in S_0} \sum_{h \geq 2} \Pr \left\{ X_h \in S_A, \forall n < h \ X_n \in S_0, X_1 = l, V_0 \leq \frac{x}{r_i} \mid X_0 = i \right\} \]

\[ = \sum_{j \in S_A} Q_{1,j} \left( \frac{x}{r_i} \right) \sum_{h \geq 2} \Pr \left\{ X_h \in S_A, \forall n < h \ X_n \in S_0, X_1 = l \right\} \]

\[ = \sum_{j \in S_A} Q_{1,j} \left( \frac{x}{r_i} \right) \sum_{h \geq 1} \Pr \left\{ X_h \in S_A, \forall n < h \ X_n \in S_0, X_0 = l \right\} \]

\[ = \sum_{j \in S_A} Q_{1,j} \left( \frac{x}{r_i} \right) \sum_{j \in S_A} \sum_{i \in S_0} Q_{1,i} \left( \frac{x}{r_i} \right) M_{i,j} = \sum_{j \in S_A} \hat{Q}_{1,j}(x). \]

Suppose now the lemma is true for 1, 2, \ldots, n - 1 and let n > 1.
\[ \Pr \{ \hat{X}(x) \in S_A, \hat{N}(\infty) = n \mid \hat{X}_0 = i \} \]

\[ = \sum_{j \in S_0} \int_0^x \Pr \{ \hat{X}(x-u) \in S_A, \hat{N}(\infty) = n-1 \mid \hat{X}_0 = j \} d\hat{Q}_{1,j}(u) \]
Proof of Lemma 3: Distinguish three cases:

If $i \in S_T$, 

$$C_i(x) = \sum_{n \geq 0} \mathbb{P}\{Y(\infty) \leq x, N(\infty) = n | X_0 = i\} = \sum_{n \geq 0} \mathbb{P}\{\hat{X}(x) \in S_A, \hat{N}(\infty) = n | \hat{X}_0 = i\}$$

using Lemma 2

$$= \hat{F}_i(x).$$

If $i \in S_A$, $C_i(x) = \hat{F}(x) = 1$.

If $i \in S_0$, 

$$C_i(x) = \sum_{j \in S_0} M_{i,j} C_j(x)$$

using Lemma 1

$$= \sum_{j \in S_0} M_{i,j} \hat{F}_j(x).$$

REFERENCES


Gianfranco Ciardo (S’87–M’88) received the Doctor degree in computer science from the University of Torino, Torino, Italy, in 1982 and the Ph.D. degree in computer science from Duke University in 1989.

From 1982 to 1984 he worked on database interfaces as an Assistant Researcher at CSELT, Torino, Italy. In 1984, he began graduate studies at the Department of Computer Science Duke University, Durham, NC. He was awarded an IBM Graduate Fellowship for the 1986–1987 academic year. He joined Software Productivity Consortium, Herndon, VA in 1988. His current interests are logic and stochastic modeling, performance evaluation, and reliability of complex hardware/software systems, with an emphasis on the theory and applications of stochastic Petri nets. He is a codesigner and developer of the SPNP package.

Dr. Ciardo is a member of the Association for Computing Machinery and the IEEE Computer Society.

Raymond A. Marie received the Doctorat d’Ingénieur and the Doctorat d’Etat es-Sciences Mathématiques from the University of Rennes, France, in 1973 and 1978, respectively.

Since 1977, he has been a Research Manager of an INRIA project. He spent the 1981–1982 academic year as a Visiting Associate Professor at North Carolina State University, Raleigh. Currently, he is a Professor at the Computer Science Department of the University of Rennes. His active research interests include performance evaluation of computer systems and reliability computation of complex systems.

Bruno Sericola was born in Marange Silvange, Moselle, France, on May 25, 1959. He received the Doctorat d’Université in computer science from the University of Rennes, France, in June 1988. He is currently working on an ESPRIT project at the University of Rennes. His active research interests include computer system performance modeling, fault-tolerant computing system, applied stochastic processes, and numerical analysis.

Kishor S. Trivedi (M’86–SM’87) received the B. Tech. degree from the Indian Institute of Technology, Bombay, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana–Champaign.

He is the author of a widely used text, Probability and Statistics with Reliability, Queuing and Computer Science Applications, (Englewood Cliffs, NJ: Prentice-Hall). Both the text, and his related research activities, have been focused on computing system reliability and performance evaluation. Presently, he is a Professor of Computer Science and Electrical Engineering, Duke University, Durham, NC. He has served as a Principal Investigator on various AFOSR, ARO, Burroughs, IBM, NASA, NIH, NSF, ONR, and SPC funded projects and as a consultant to the industry and research laboratories.

Dr. Trivedi was an Editor of the IEEE TRANSACTIONS ON COMPUTERS from 1980 to 1987. He was an editor of the Journal of Parallel and Distributed Computing. He is a codesigner of HARPS, SAVE, SHARPE, and SPNP packages.