*** RELATÓRIO TÉNICO ***

STATE SPACE EXPLORATION IN MARKOV MODELS

Edmundo de Souza e Silva*
Pedro Mejía Ochoa**

NCE 09/91
Julho/91

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

* Universidade Federal do Rio de Janeiro
** Hondutel-Honduras

STATE SPACE EXPLORATION IN MARKOV MODELS

RESUMO

Análise de desempenho e dependabilidade baseia-se usualmente em modelos Markovianos.

Um dos principais problemas que o analista encontra é a grande cardinalidade do espaço de estados da cadeia de Markov associada ao modelo, o que impede não somente a solução do modelo, mas também a geração da matriz de transição de estados.

Entretanto, em muitos modelos de sistemas reais, a maioria da massa de probalidade está concentrada em um pequeno número de estados em comparação com a totalidade do espaço de estados. Por conseguinte, medidas de desempenhabilidade ("performability") podem ser avaliadas com precisão a partir desses estados "mais prováveis".

Neste artigo, apresentamos um algoritmo de geração dos estados mais prováveis que é mais eficiente que algoritmos anteriormente propostos na literatura.

Abordaremos também o problema de cálculo das medidas de interesse e mostraremos como limites para algumas medidas poden ser eficientemente calculados.

State Space Exploration in Markov Models[1]

Edmundo de Souza e Silva
Federal University of Rio de Janeiro. NCE and CS Dept.
Cx.P. 2324, Rio de Janeiro, Brazil

Pedro Mejía Ochoa[2]
Hondutel

September 1991

[1]This research was partially supported by a grant from CNPq-Brazil and NSF grant INT-8514377.
[2]This work was done while the author was at The Federal University of Rio de Janeiro, COPPE, partially supported by grants from Hondutel-Honduras and CAPES-Brazil

Abstract

Performance and dependability analysis is usually based on Markov models. One of the main problems faced by the analyst is the large state space cardinality of the Markov chain associated with the model, which precludes not only the model solution, but also the generation of the transition rate matrix. However, in many real system models, most of the probability mass is concentrated in a small number of states in comparison with the whole state space. Therefore, performability measures may be accurately evaluated from these "high probable" states. In this paper, we present an algorithm to generate the most probable states that is more efficient than previous algorithms in the literature. We also address the problem of calculating measures of interest and show how bounds on some measures can be efficiently calculated.

1 Introduction.

Performance and dependability analysis of computer and communication systems has been an important tool for designers who wish to understand and predict the behavior of such systems. Performance modelling attempts to capture the effect of contention for resources in the system and queueing networks have been extensively used as a modelling tool (e.g., [32,33,7,15]). In particular, major advances in the area where due to the product form solution which allows performance measures to be obtained without the solution of the underlying Markov chain model. Dependability modelling is concerned with the changes in the structure of the system which may occur due to faults in its components. In this area, Markov chain models are extensively used (e.g., [45]) since, except for a few cases (e.g., [21]), the models do not possess product form solution. More recently, with the advent of degradable computer systems, the combined modelling of performance and dependability, the so called *performability* modelling [36], has received increasing attention (see [14,37]) and references therein). Similarly to dependability modelling, Markov models are the main tool for performability modelling.

The complexity of the current computer systems led to the development of many tools for performance/dependability modelling, e.g. [2,20,4,41,25,6,40] (see also [30] for a survey and further references). Several tools allow the analyst to describe the system in a high level representation and automatically generate the corresponding Markov chain model. Once the transition rate matrix is generated, Markov chain solution techniques are used to obtain the measures of interest.

One of the major problems faced by such tools is the large state space cardinality of the Markov chain associated with the models. For instance, in dependability (performance) models, the number of states grows exponentially with the number of components that can fail (number of resources and customers in the model). So, unless the model has special properties (e.g., product form) the solution is impractical. Large state space cardinality has a major impact not only on the solution techniques but also on model generation, since it may be too time consuming to generate and practically impossible to store a transition rate matrix for a Markovian model with millions or billions of states.

Many techniques have been developed to deal with large state space cardinality, and among those we mention: decomposition, lumping, truncation. Decomposition is a useful technique for analyzing systems consisting of weakly coupled subsystems. In this case, the underlying transition rate matrix is nearly completely decomposable and an approximation solution for the state probabilities can be found based on the solution of each individual subsystem [43,8]. This technique has been found very useful when applied to queueing networks, although it is not as useful for dependability modelling, since the underlying Markov chain is not nearly completely decomposable, in general. However, the basic aggre-

gation/disaggregation technique described in [8] is applicable, regardless of the form of the transition rate matrix. Courtois and Semal [9] are concerned with the calculation of steady state probabilities when only part of the model is generated. They obtain bounds on steady state probabilities of each state in a subset of states, conditioned on the system being in a state of the subset, and show how the approach can be useful for certain queueing models [11]. Muntz *et al* [38] developed a method for bounding steady state availability of repairable computer system models, based on the technique of Courtois and Semal. Their approach uses certain properties of availability models in order to obtain tight bounds. Courtois and Semal [10] obtain bounds on the mean and second moment of the mean time until absorption (MTA) based on the results of [9]. The technique is shown to be cheaper than traditional methods to calculate the MTA.

Lumping is a method that is used to reduce the state space of models. For a given measure of interest, subsets of states may be combined (lumped) into a single state, without affecting the final result. In [31], conditions for lumping states of a Markov model are given in terms of the transition probabilities of the one step transition probability matrix, and in [21] lumping is applied to availability modelling. Recently, Sanders and Meyer [42] proposed a state space reduction method based on lumpability conditions, but which can be applied without the need to construct the transition rate matrix. Instead, submodels of a high level system description (a SAN model or Stochastic Petri net model) are identified and joined, resulting in a much smaller state space than in the original model. In general, however, lumping is restricted to the existence of symmetries in the model.

The truncation of the state space, after generating a number of states, is another method which can be used to cope with large state space cardinality. For availability modelling, a "natural" partition of the state space exists in terms of the number of components failed. For example, in the SAVE [20] tool states can be generated up to a given number of failed components in the model, according to the user specifications. As can be seen in the examples given in [19], the steady state availability converges fast to the final result, after generating a relatively small number of states corresponding to those with a few failed components in the system. An issue is how to bound the final solution and, as mentioned above, accurate bounds on steady state availability can be found [38,39].

The results in [46] can be applied to bound the absolute value of the difference between the expected accumulated reward of a Markov model and a perturbed model with different one step transition probabilities and rewards for the states. Truncation is an example of such perturbation.

In many performance/dependability models, the state probabilities are highly skewed and this property has been used to calculate performability measures. For instance, in availability models, and for highly available systems, it is reasonable to assume that most of the probability mass is concentrated on the states that represent the system with only

a few components failed. This observation motivated the bounding technique of [38], and was used to obtain computational savings when transient performability measures [13,12] are calculated. It also indicates that truncation techniques should produce good results. Other examples of models with highly unbalanced state probabilities are those of communication protocols (e.g., [35]). In general, it may be difficult to find a "natural" state space partition that contains the most probable states. As a consequence, a very important issue is how to generate the subset of those important states in the model. The so called "dynamic state exploration" techniques address this issue.

The basic idea of dynamic state exploration techniques is to develop algorithms to guide the generation of the transition rate matrix (or equivalently the one step transition probability matrix) so that "important" states are generated first. The importance of a state should be given in terms of its contribution to calculate the measure of interest. For instance, for many measures such as the ones obtained in availability modelling, the states with the highest steady state probabilities should be generated first. This contrasts with the depth first or breadth first techniques, commonly used for generating the state space of Markov models.

Several papers address the issue of generating the most probable states [23,47,17,16]. In this work we extend the results of [16] and develop a new dynamic state exploration algorithm which is shown to provide significant computational savings when applied to computer and communication models. We also discuss the use of a different criterion than the one used in [16] to guide the choice of the next state generated. This criterion, may be computational more expensive than the previous one, but can be useful in certain cases. Then, we address the problem of calculating the measures of interest, and propose a way to bound the solution.

In section 2 we discuss some dynamic state generation techniques and present the background material. Section 3 provides a description of our approach. In section 4 we discuss issues related to the criterion used to guide the search procedure and also address the calculation of some measures of interest. Section 5 presents examples to illustrate the application of the results. Our conclusions are presented in section 6.

2 Dynamic State Exploration.

As mentioned in the introduction, in many computer and communication models most of the time is spent in a relatively small number of states, in comparison to the total number of states. As a consequence, the probability mass is highly skewed. For such models, several measures of interest can be calculated from a relatively small number of states. Dynamic state space exploration techniques try to find these most probable states up to a given

tolerance, without the need to generate the whole state space.

The generation procedure is usually based on the "transient" behavior of the system from a given state. By transient we mean that the measure used for ending the search is related to the amount of time the system remains in the subset of the states generated from the given state, before leaving the subset. For instance, the generation stops when the expected amount of time in the subset is greater than an specified value. This procedure is good for transient measures, but it may not be acceptable for steady state measures. Note that the subset of generated states may not be even similar to the subset of the most probable states (up to a given tolerance) in steady state. Clearly, if we do not have any knowledge of the system behavior once in the non-generated states, it is possible that a highly probable, but not yet generated, state exists (for instance, an absorbing state). Nevertheless, this procedure can be used to obtain bounds on steady state measures conditioned that the system is in the subset of generated states, and this is useful in many cases. Depending on the amount of knowledge of the system being modelled, unconditioned steady state measures can also be bounded.

Grassman [23] proposed a dynamic state generation method to be used in conjunction with the randomization technique [3] to avoid the generation of the entire transition rate matrix before the randomization procedure starts. At each step of the randomization procedure (say step k), an "active state set" is updated. The active set contains all states that can be reached from the set of initial states in k steps. New states are generated, one by one, and the probability of being in any of the states after k jumps is also calculated incrementally. This method is useful because it may avoid the generation of the complete transition rate matrix specially when the infinite sum in the randomization procedure is truncated to a small value N. However, the number of states generated at each step increases very fast.

The work of Yang and Kubat [47] is an example of a method that obtains the set of the most probable states in steady state. The algorithm is developed for a very particular system model and improves previous work of Li and Silvester [34] and Chiou and Li [5]. The model is composed of N components that can operate in one of several possible modes. Furthermore, a component i operates in mode m_i ($1 \leq i \leq N$, $1 \leq m_i \leq M_i$, M_i is the number of modes of component i) with probability p_{im_i}, independently of other units. From this particular model, it is easy to obtain the steady state probability for a given state. The form of the solution is used to transform the original problem into a tree search procedure and from this generate the most probable states.

Maxemchuck and Sabnani [35] are interested in obtaining the most probable states for models of communication protocols. They argued that "partial" evaluation is useful due to the highly skewed probabilities of most protocol models. The main assumptions used in [35] are that a protocol is modeled as a collection of finite state machines (FSM) and non-deterministic transitions of each FSM has only two choices called: a "high probability

choice" (with probability $1-p$) and a "low probability choice" (with probability $p << 1$). Furthermore, each change in the global state is the result of a state change in two of the FSM's of the model. From these assumptions, each transition from a global state to another has either probability $1-p \approx 1$ or p or p^2. The search algorithm generates the reachable states from a starting one and organize these states into classes according to the probability of reaching the generated state from the initial one. This is not difficult due to the assumptions above. (The probability of reaching a state in class i is p^i.) One measure that can be computed is the probability of reaching a state that has not been explored after n runs of the protocol, starting from the initial state.

Dimitrijevic and Chen [17,16] developed a dynamic generation method based on a recursive calculation of the expected number of visits to a state between successive visits to a given (initial) state. There are no restrictions with respect to the model as in [35]. We describe this method in more detail since our approach is based on a similar search for the states. This will also serve to introduce some basic notation used throughout the paper.

Let $\mathcal{X} = \{X(t) : t \geq 0\}$ be a homogeneous continuous time Markov process with generator $\mathbf{Q}$ that describes the behavior of the system being modelled, and let $\mathcal{S} = \{a_i : i = 1, \ldots, M\}$ be the finite state space associated with the model. Let q_{ij} be the entry ij of $\mathbf{Q}$. It is possible to think of $\mathcal{X}$ as a discrete time Markov chain $\mathcal{Z} = \{Z_n : n = 0, 1, \ldots\}$, with state space $\mathcal{S}$ and transition matrix $\mathbf{P} = \mathbf{Q}/\Lambda + \mathbf{I}$ ($\Lambda \geq \max q_i = \sum_{j \neq i} q_{ij}$), subordinated to a Poisson process $\mathcal{N} = \{N(t) : t \geq 0\}$. For $t \geq 0$, it is known that $X(t) = Z_{N(t)}$ [29,3]. The transformation of a continuous time Markov chain into a discrete time Markov chain subordinated to a Poisson process is called randomization or uniformization. In what follows, to simplify our description and without loss of generality, we consider the uniformized process $Z_{N(t)}$ and its associated transition matrix $\mathbf{P}$.

Let g_{kl} be the expected number of visits to state l in a path that starts in state a_k and ends in state a_1, the initial state. Let V_l be the expected number of visits to state a_l between two visits to state a_1. By definition, $V_l = g_{1l}$.

At step n ($n = 1, \ldots N$) of the dynamic generation procedure, there are two sets of states: $\mathcal{S}_e^{(n)}$ and $\mathcal{S}_u^{(n)}$. The set $\mathcal{S}_e^{(n)}$ contains n states and is called the *explored set*. This set includes the states (called *explored states*) chosen to be included in the final reduced model. All transition probabilities from each state in $\mathcal{S}_e^{(n)}$ have already been found in step n. The set $\mathcal{S}_u^{(n)}$ is called the *unexplored set* and contains the neighboring states to those in $\mathcal{S}_e^{(n)}$. In step n, the transitions from each state in $\mathcal{S}_u^{(n)}$ have not been found yet. It is assumed that for all states $a_i \in \mathcal{S}_u^{(n)}$ there is a single transition $p_{iu} = 1$ to a fictitious state a_u that represents the "unknown" states in the model. Furthermore, the definition of g_{kl} is modified to include visits in a path that starts in a_k and ends in a_u (i.e., g_{kl} is the expected number of visits to state l in a path that starts in a_k and ends in a_1 or a_u). The definition of V_l is modified accordingly. Figure 1 illustrates the sets of states $\mathcal{S}_e^{(n)}$ and $\mathcal{S}_u^{(n)}$ and their transitions.

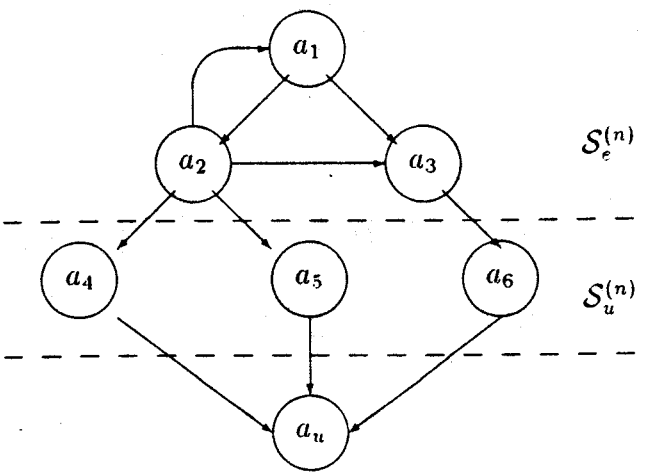

Figure 1: Sets of states in step n.

The generation algorithm is as follows:

1. Initial step: $\mathcal{S}_e^{(1)} = \{a_1\}$, $\mathcal{S}_u^{(1)} = \{\emptyset\}$. $s = a_1$, $n = 1$.

2. $n = n + 1$.
   Find all transitions from s and all its neighboring states. $\mathcal{S}_u^{(n)} = \mathcal{S}_u^{(n-1)} \bigcup \{$ neighbors of s not in $\mathcal{S}_e^{(n-1)} \bigcup \mathcal{S}_u^{(n-1)}\}$.

3. Stop if the stopping criterion is met.

4. Choose a state in $\mathcal{S}_u^{(n)}$, say state a_c, according to some rule. $\mathcal{S}_e^{(n)} = \mathcal{S}_e^{(n-1)} \bigcup \{a_c\}$, $\mathcal{S}_u^{(n)} = \mathcal{S}_u^{(n)} - \{a_c\}$.

5. $s = a_c$, go to 2.

As mentioned previously, the stop criterion is related to the amount of time spent in the generated states. In [16], the stopping criterion is the average amount of time in the generated states before reaching the fictitious state a_u. Since the mean time to unknown at step n ($MTTU^{(n)}$) can be calculated from the visit ratios:

$$MTTU^{(n)} = \frac{\sum_i V_i^{(n)}}{V_u^{(n)}} \tag{1}$$

(where the superscript indicates the step of the algorithm) the rule used in [16] to choose the next state in step 4 above is based on the visit ratios. In other words, assume that $V_i^{(n)}$ is calculated at step n of the algorithm, for all $a_i \in \mathcal{S}_u^{(n)}$. A state $a_c \in \mathcal{S}_u^{(n)}$ is chosen if $V_c^{(n)} = \max\{V_i^{(n)}, a_i \in \mathcal{S}_u^{(n)}\}$.

The key to the approach is to develop an efficient algorithm to compute $V_i^{(n)} = g_{1i}^{(n)}$, $\forall a_i \in \mathcal{S}_u^{(n)}$, for each step of the generation procedure. It is known that the $g_{kl}^{(n)}$ can be calculated from the equation below (e.g., see [26]):

$$g_{kl}^{(n)} = \sum_{j \neq 1} p_{kj} g_{jl}^{(n)} + \delta_{kl} \tag{2}$$

where p_{kj} is the one step transition probability from state a_k to state a_j, and $\delta_{kl} = 1$ if $k = l$ and 0 otherwise. It is easy to see that solving equation (2) at each step of the algorithm would require $O(M^4)$ operations, where M is the number of generated states. In [16], an efficient recursive solution is proposed to calculated $g_{kl}^{(n)}$ $\forall k, l$. It it shown that:

$$g_{kl}^{(n)} = g_{kl}^{(n-1)} + g_{kc}^{(n-1)} \frac{\sum_{j \neq 1} p_{cj} g_{jl}^{(n-1)}}{1 - \sum_{j \neq 1} p_{cj} g_{jc}^{(n-1)}} \tag{3}$$

where a_c is the state chosen in step $(n-1)$. Since $g_{kl} = 0$ $\forall a_k \in \mathcal{S}_u^{(n)}$ (see Figure 1), equation (3) can be evaluated as follows:

- calculate $\sum_{j \neq 1} p_{cj} g_{jc}^{(n-1)}$
- for all $a_l \in \mathcal{S}_e^{(n)} \bigcup \mathcal{S}_u^{(n)}$
  - calculate $\sum_{j \neq 1} p_{cj} g_{jl}^{(n-1)}$
  - for all $a_k \in \mathcal{S}_e^{(n)}$
    - calculate $g_{kl}^{(n)}$

Let $L^{(n)}$ be the cardinality of $\mathcal{S}_u^{(n)}$. (Recall that n is the cardinality of $\mathcal{S}_e^{(n)}$.) The number of operations to calculate the $g_{kl}^{(n)}$ is $O(n(n + L^{(n)}))$. In general, $L^{(n)}$ can be much larger than n. If we assume that each state has, in the average, r output transitions and a fraction p of those ends in non-explored states, then $L^{(n)} = prn$ and the number of operations at step n is $O((1 + pr)n^2)$ and so the total number of operations is $O((1 + pr)N^3)$, where N is the total number of steps of the algorithm. The total storage needed is $O(N(N + L^{(N)})) = O((1 + pr)N^2)$.

In the next sections we present an algorithm which is shown to have less computational requirements than the above procedure. We also show its applicability to models other than communication protocols, such as availability models.

3 An Iterative Approach.

There are two main issues concerning the generation procedure outlined in the previous section: the choice of an appropriate rule in step 4 in order to achieve the goal of generating the most probable states and; the computational requirements needed to apply the rule. In this section we address the second issue.

We assume that the rule used to choose a state at each step of the algorithm is based on the visit ratio of the states, as proposed by [16] and outlined in section 2. The method of [16], summarized in equation (3), is recursive and, at step n, the values of $g_{kl}^{(n)}$ are calculated from the values of $g_{kl}^{(n-1)}$. One main drawback of this technique is that the storage requirements are very high: $O((1+pr)N^2)$. This is a major problem for large values of N (say, tens of thousands). One of the important advantages of iterative solution techniques to obtain the measures of interest once the model is generated is that they preserve the sparseness of the transition rate matrix of the model. Since the recursive method for generating the model requires the storage of a full matrix (all entries are different than zero) with size equivalent to the final (truncated) transition rate matrix, the advantage of preserving the sparseness of matrices usually obtained from real system models is lost. Another drawback of the technique is the potential for numerical problems due to the denominator of equation (3). If the sum in the denominator is close to one, there may be a considerable loss of accuracy in the final result. As indicated by Grassman (e.g., [22]) algorithms that involve subtractions are amenable to round off errors.

In what follows, we propose an iterative technique to calculate directly the visit ratios ($V_l^{(n)} = g_{1l}^{(n)}$) and which takes advantage of sparse matrices. The idea is based on the observation that, if the solution for $\mathbf{V}^{(n-1)} =< V_1^{(n-1)}, \ldots, V_{n-1}^{(n-1)} >$ is available at step $n-1$ then, when a new state is added at step n, the value for $V_i^{(n)}$ should not differ much from the previous value $V_i^{(n-1)}$ $(1 \leq i \leq n-1)$.

At step n, all the transition probabilities from the states in $\mathcal{S}_e^{(n)}$ are known. The one step transition probability matrix $\mathbf{P}^{(n)}$ is given by (see also Figure 1):

$$\mathbf{P}^{(n)} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n} & p_{1,n+1} & \cdots & p_{1,n+L^{(n)}} & 0 \\ p_{21} & p_{22} & \cdots & p_{2,n} & p_{2,n+1} & \cdots & p_{2,n+L^{(n)}} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} & p_{n,n+1} & \cdots & p_{n,n+L^{(n)}} & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad (4)$$

where we organize the states so that the first n are in $\mathcal{S}_e^{(n)}$, the subsequent $L^{(n)}$ are in $\mathcal{S}_u^{(n)}$ and the last one is the fictitious state a_u. In order to calculate the visit ratios of each state between visits to state a_1 or a_u, it is useful to observe that $\mathbf{P}^{(n)}$ represents a Markov chain with one absorbing state, and this process has equivalent behavior to the one defined by matrix $\mathbf{\Pi}^{(n)}$ in (5) where, whenever the process reaches the absorbing state a_u, it is restarted from the initial state. Furthermore, in $\mathbf{\Pi}^{(n)}$, the states $a_i \in \mathcal{S}_u^{(n)} \bigcup \{a_u\}$ are aggregated to a single state a_f:

$$\mathbf{\Pi}^{(n)} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n} & p_{1,f} \\ p_{21} & p_{22} & \cdots & p_{2,n} & p_{2,f} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} & p_{n,f} \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{5}$$

The values of p_{if} in (5) are the sum of the transition probabilities from state a_i to states in $\mathcal{S}_u^{(n)}$. $p_{if} = \sum_{j=n+1}^{n+L^{(n)}} p_{ij}$. The irreducible ergodic Markov chain given by $\mathbf{\Pi}^{(n)}$ describes the behavior of the original absorbing process given by $\mathbf{P}^{(n)}$, over an infinite number of runs that start from the initial state. Thus $V_i^{(n)} = V_i'^{(n)}$, where the "prime" indicates the quantity related to process $\mathbf{\Pi}^{(n)}$. If $\pi^{(n)} =< \pi_1, \ldots, \pi_f >$ is the solution of $\pi^{(n)} = \pi^{(n)}\mathbf{\Pi}^{(n)}$, then

$$V_i'^{(n)} = \frac{\pi_i^{(n)}}{\pi_1^{(n)}} \tag{6}$$

The visit ratio of states $a_i \in \mathcal{S}_u^{(n)}$ needed to choose the next state to be included in $\mathcal{S}_e^{(n+1)}$ are also easily calculated from $\pi^{(n)}$ as:

$$V_l'^{(n)} = \frac{\sum_{j=1}^{n} p_{jl}\pi_j}{\pi_1} \qquad l \in \mathcal{S}_u^{(n)} \tag{7}$$

Finally, the $MTTU^{(n)}$, needed for stopping the algorithm, can be calculated as the sum of the expected number of visits to each state in $\mathcal{S}_e^{(n)}$ between two visits to state a_f:

$$MTTU^{(n)} = \sum_{i=1}^{n} \frac{\pi_i^{(n)}}{\pi_f} = \frac{1 - \pi_f^{(n)}}{\pi_f^{(n)}} \tag{8}$$

$$= \frac{1}{\pi_f} - 1 \tag{9}$$

Since the Markov chain determined by $\mathbf{\Pi}^{(n)}$ is ergodic, the solution $\pi^{(n)} = \pi^{(n)}\mathbf{\Pi}^{(n)}$ can be calculated by iterative techniques. For those techniques, an initial distribution for the $\pi^{(n)}$ is needed. Let $\pi^{(n)}(0)$ be this initial distribution at step n. At a new step of the algorithm a new state is added to the explored set of states, but the transition probabilities among the states in the previous explored set remain the same. Intuitively, if the probability mass is

skewed, after a few steps the value of $\pi_i^{(n-1)}$ $(1 \leq i \leq n-1)$ should not differ much from the values of $\pi_i^{(n)}$. Therefore, the set of $\{\pi_i^{(n-1)}\}$'s is a good starting point for the iterative algorithm at step n and only a few iterations should be required to find the solution $\pi^{(n)}$. As we will show below, significant computational savings can be obtained with the use of iterative techniques.

In summary, we set

$$\pi^{(n)}(0) = < \pi_1^{(n-1)}, \ldots, \pi_{n-1}^{(n-1)}, \pi_c^{(n-1)}, \pi_f^{(n-1)} - \pi_c^{(n-1)} > \tag{10}$$

where $\pi_c^{(n-1)} = V_c'^{(n-1)} \pi_1^{(n-1)}$ and a_c identifies the state chosen to be included in $\mathcal{S}_e^{(n)}$, and $V_c'^{(n-1)}$ is obtained from equation (7).

Computational Requirements.

In order to calculate the computational requirements of the searching algorithm using iterative techniques, we assume, as before, that r is the average number of transitions out of a state and p is the fraction of those transitions that go to states in $\mathcal{S}_u^{(n)}$. At step n, matrix $\mathbf{\Pi}^{(n)}$ has dimension $(n+1) \times (n+1)$. Therefore, the number of multiplications performed to solve $\pi^{(n)} = \pi^{(n)} \mathbf{\Pi}^{(n)}$ is $I^{(n)}(1-p)r(n+1)$ where $I^{(n)}$ is the number of iterations at step n. Assuming $I^{(n)}$ is approximately constant for all n and equal to I, the total number of operations needed for the search algorithm over N steps is $O(I(1-p)rN^2)$.

The storage requirements for the iterative algorithm is rN since all transitions out of the states in $\mathcal{S}_e^{(n)}$ need to be stored at each step (and so we have rN transitions in the last step). Therefore, iterative generation algorithms preserve the sparseness of the transition rate matrix of system models. Note also that the storage requirements are independent of I.

Comparing iterative approaches with the recursive approach of [16] we see that:

(a) number of operations:
iterative/recursive $= [I(1-p)rN^2]/[N^2(N+L^{(N)})] = [I(1-p)r][N(1+pr)]$. Since $r << N$, if I is small compared to N, then large computational savings are obtained.

(b) storage requirements:
iterative/recursive $= [rN]/[N(N+L^{(N)})] = [r]/[N(1+pr)]$ Since $r << N$ the storage requirements of iterative techniques are much lower than the ones for the recursive technique, and it is independent of I, the number of iterations at each step.

The Choice of an Iterative Technique.

The number of iterations for an iterative technique has an impact on the number of operations needed for the generation procedure. (But no impact on the storage requirements.)

Therefore, the choice of a method that requires few iterations is important. We have experimented with different iterative techniques and choose the SOR (Successive Over Relaxation) method (e.g., see [18]). In the method, an element $\pi_i(m)$ at step m of the iteration is calculated as:

$$\pi_i(m) = (1-w)\pi_i(m-1) + \frac{w}{1-p_{ii}}\left[\sum_{j<i} p_{ji}\pi_j(m) + \sum_{j>i} p_{ji}\pi_j(m-1)\right] \tag{11}$$

where w is the so called relaxation parameter. The choice of w is crucial for improving the convergence of the method. The optimum value of w can be estimated from the computed values of $\pi(m)$ during the iteration (see [48,44]).

The convergence of this method is fast. However, there are cases where it does not converge. When $w \geq 2$, the method diverges, and so we use a variant of the Power method which has less strict convergence conditions than those for the traditional one [1]. Let

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \tag{12}$$

The m^{th} iteration is given by:

$$\pi^2(m) = \pi(m-1)\mathbf{P}_{22} + \pi^1(0)P_{12} \tag{13}$$

where $\pi(m) =< \pi^1(0), \pi^2(m) >$ and converges to the solution of $\pi = \pi\mathbf{P}$. (The entries of π do not add to one, but they can be normalized later.)

In the many examples we run, the convergence of the method was fast and in the order of 3 to 8 iterations per each step of the generation procedure. In the next section we discuss other rules to apply in the selection of a state in each step of the generation.

4 Additional Considerations.

In the previous section we show that iterative techniques perform better than the recursive technique of [16] to implement the search rule based on visit ratios. In this section we address two issues. One is related to the search rule as implemented by the algorithm of section 2. The other discusses the calculation of measures of interest, once the generation procedure is over.

4.1 The search Rule.

The stopping criterion used in step 3 of the algorithm described in section 2 is the average amount of time spent in the generated states before exit (MTTU). Therefore, the search rule should try to maximize the MTTU at each step of the algorithm. The MTTU can be calculated from the visit ratios using equation (1). However, choosing a state a_i from those in $\mathcal{S}_u^{(n)}$ so that it has the maximum value of $V_i^{(n)}$, as implemented by the algorithm of section 2. may not be the best choice for increasing the MTTU of the new set $\mathcal{S}_e^{(n+1)}$, as it can be seen by the following example.

Assume that at step n of the search, $\mathcal{S}_e^{(n)} = \{a_i\}$ and $\mathcal{S}_u^{(n)} = \{a_2, a_3\}$ as shown in Figure 2. From the figure it is easy to see that $V_2^{(n)} = 0.4$, $V_3^{(n)} = 0.6$ and thus, according to the

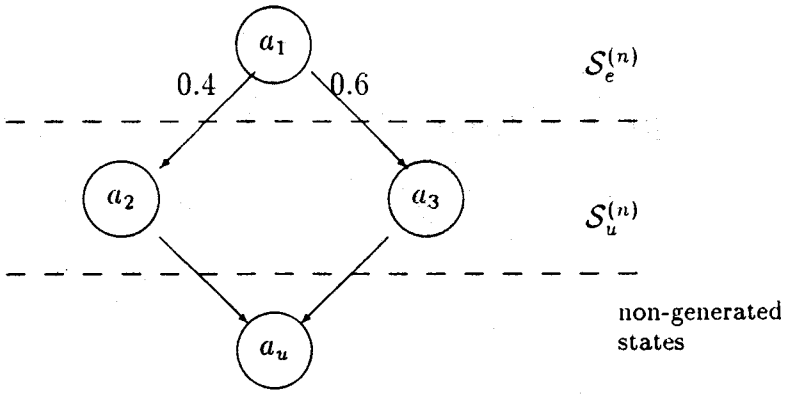

Figure 2: An example of the search.

rule which chooses the state with the maximum value of the visit ratio, state a_3 is chosen. Now assume that state a_2 (a_3) has a transition back to itself with probability 0.9 (0.1). If we take into account these transitions when we calculate the visit ratios then $V_2^{(n)} = 4$ and $V_3^{(n)} = 2/3$, and state a_2 would be chosen. From equation (8), clearly this choice maximizes the MTTU. This problem arises because, at each step of the search, the choice is based on the visit ratios of the unexplored states and we do not know the output transitions from these states. This problem can be overcome if we modify step 2 of the search algorithm of section 2 as indicated below:

Step 2: n=n+1.
Find all transitions from s and all its neighboring states.
Let $\mathcal{S}_w^{(n)} = \{$ neighbors of s not in $\mathcal{S}_e^{(n-1)} \bigcup \mathcal{S}_u^{(n-1)}\}$. (Those are the new states found.)
Find all output transitions from states in $\mathcal{S}_w^{(n)}$.

Find all output transitions from states in $\mathcal{S}_u^{(n-1)}$ to states in $\mathcal{S}_w^{(n)}$ and update the transitions to state a_u accordingly.
$\mathcal{S}_u^{(n)} = \mathcal{S}_u^{(n-1)} \bigcup \mathcal{S}_w^{(n)}$

This new algorithm may require more computational effort than the previous one, both when we use the recursive or the iterative technique as well. First, consider the recursive technique. At step n, $g_{kl}^{(n)}$, $\forall a_k \in \mathcal{S}_u^{(n)}$ is not necessarily equal to zero and, therefore, the number of operations at step n is $O((n + L^{(n)})^2)$, and the storage requirements are also $O((n + L^{(n)})^2)$. We assume, as before, that the cardinality of $\mathcal{S}_u^{(n)}$, $L^{(n)}$, is prn. Since we are generating the output transitions not only from states in $\mathcal{S}_e^{(n)}$ but from states in $\mathcal{S}_u^{(n)}$ as well, then our stop criterion should consider the states in $\mathcal{S}_e^{(n)} \bigcup \mathcal{S}_u^{(n)}$. Assume that we stop the generation procedure at step M when the cardinality of $\mathcal{S}_e^{(M)} \bigcup \mathcal{S}_u^{(M)}$ is N^*. Therefore, $M = N^*/(1 + pr)$. The total number of operations is $O((N^*)^3/(1 + pr))$. The storage requirements is $O((N^*)^2)$.

Now we consider the iterative technique. The matrix $\mathbf{\Pi}^{(n)}$ in (5) has now $n + L^{(n)} + 1$ entries. Thus, the number of operations at step n is $O(Ir(n + L^{(n)})) = O(Ir(1 + pr)n)$ and the total number of operations is $O(Ir(N^*)^2/(1 + pr))$. The total storage requirements are $O(rN^*)$.

From the above, we note that: (a) The iterative technique, as before, outperforms the recursive technique for this modified version of the algorithm. (b) The rule for choosing a state guarantees that the MTTU from states in $\mathcal{S}_e^{(n)}$ is maximized among the possible choices. However, this does not guarantee that the mean time to exit from states in $\mathcal{S}_e^{(n)} \bigcup \mathcal{S}_u^{(n)}$ is maximized. The computational requirements of this last algorithm can be larger or smaller than the previous one, depending on the value of N^* and the value of pr. (c) We can reduce the storage requirements of this last algorithm with respect to the previous one, by increasing the number of operations. This can be done if the rule in step 4 of the original algorithm is modified as follows:

- for each state $a_i \in \mathcal{S}_u^{(n)}$:
  - generate the transitions out of a_i.
  - calculate the MTTU considering the states in $\mathcal{S}_e^{(n)} \bigcup \{a_i\}$.
- choose the state that gives the largest value from all values of the MTTU calculated.

The storage requirements for the recursive and iterative techniques are $O((1 + pr)M^2)$ and $O(rM)$ respectively. Since our search rule is anticipated to produce better results than the previous one, we expect that $M < N$. The computational requirements are $O((1 + pr)^2 M^4)$ and $O(I(1 - p)pr^2 M^3)$ for the recursive and iterative algorithms, respectively.

4.2 Measures of Interest.

In the previous section we are concerned with the generation of states such that, for a given initial state, the mean time to reach the non-generated states is greater than a given value. With the procedure outlined above, we intend to generate the states that concentrate most of the probability mass. Once the states are generated, it remains to calculate measures of interest to the user such as performance, dependability or, more generally, performability measures, and to bound the final solution. (For a definition of many performability measures and solution techniques, we refer the reader to [14].) We first consider transient measures. Steady state measures are considered later.

Transient Measures.

One important measure to be obtained is the random variable $L(t)$ that is equal to the time when the system reaches a non-generated state during an observation period $(0, t)$. The distribution of this random variable can be easily calculated using the randomization technique [12,21]. Note that the MTTU obtained with the generation procedure is the expected lifetime as $t \rightarrow \infty$.

We now consider the calculation of performability measures during an observation period $(0, t)$. We assign a reward rate r_i to each state of the model. This reward rate may be equal, for instance, to some measure of the performance of the system at that state. Assume that the rewards are bounded $r_{lb} \leq r_i \leq r_{up}$, for all states including the ones not generated. This assumption is realistic for many models, since we usually know some of its characteristics even without generating the states. If we aggregate the non-generated states into a single absorbing state, it is easy to see that the distribution of the cumulative reward over $(0, t)$, $CR(t)$, is bounded by:

$$P^{lb}[CR(t) > y] \leq P[CR(t) > y] \leq P^{ub}[CR(t) > y] \tag{14}$$

where the superscripts lb and ub indicate that we assign a reward r_{lb} and r_{ub} to the absorbing state a_u, respectively. Clearly, the quality of the bounds depends on the probability of reaching a_u over $(0, t)$ and the absolute value of the difference between r_{lb} and r_{ub}. For dependability models, many measures such as cumulative operational time and interval availability can be evaluated and bounded in this way since the rewards assigned to the states are either 0 or 1.

Steady State Measures.

Steady state measures are usually obtained from the steady state probabilities of each individual state in the model. The values of the state probabilities can be approximated by the ones calculated from the solution of the transition probability matrix obtained from

the generation algorithm, i.e., by solving $\pi^{(N)} = \pi^{(N)}\Pi^{(N)}$, where $\Pi^{(N)}$ is given in (5). Unfortunately, if we do not have any knowledge of the behavior of the model once in the non-generated states, it is impossible to bound accurately individual states probabilities. For instance, there may exist a highly probable state among the non-generated ones (e.g., an absorbing state in the worst case) and so, the values of $\pi^{(N)}$ obtained from $\Pi^{(N)}$ may be a poor approximation of the steady state values. Nevertheless, the steady state probabilities *conditioned* that the system remains in the subset of the generated states can be bounded by using the approach developed by Courtois and Semal [9,11]. If we have some knowledge of the model once in the non-generated states (e.g., if we know a bound for the transition rates from the non-generated states, etc.), then bounds for unconditional measures may be computable from the generated states. For instance, we refer to the bounds on availability obtained in [38,39]. Below we address some of the issues related to bounding steady state measures.

Suppose that we are interested in calculating the cumulative reward $\mathcal{R}$ averaged over the observation period $(0,t)$ as $t \to \infty$. If we know the steady state probabilities $\pi =< \pi_1, \ldots, \pi_C >$ of the complete model, then $\mathcal{R} = \sum_{i=1}^{C} \pi_i r_i$. Now we assume that the state space is partitioned into two subsets $\mathcal{G}$ which contains the states generated by some procedure and $\mathcal{N}$ which contains the non-generated states. Clearly:

$$\mathcal{R} = P(\mathcal{G})\mathcal{R}_{\mathcal{G}} + P(\mathcal{N})\mathcal{R}_{\mathcal{N}} \tag{15}$$

where $P(\mathcal{G})$ $(P(\mathcal{N}))$ is the probability that the system is in a state of $\mathcal{G}$ $(\mathcal{N})$ and $\mathcal{R}_{\mathcal{G}}$ $(\mathcal{R}_{\mathcal{N}})$ is the value of $\mathcal{R}$ conditioned that the system is in a state of $\mathcal{G}$ $(\mathcal{N})$. If the rewards are bounded, $r_{lb} \leq r_i \leq r_{ub}$, then $r_{lb} \leq \mathcal{R}_{\mathcal{N}} \leq r_{ub}$, $r_{lb} \leq \mathcal{R}_{\mathcal{G}} \leq r_{ub}$, and we have:

$$r_{lb} + |P(\mathcal{G})|_{lb}[|\mathcal{R}_{\mathcal{G}}|_{lb} - r_{lb}] \leq \mathcal{R} \leq r_{ub} - |P(\mathcal{G})|_{lb}[r_{ub} - |\mathcal{R}_{\mathcal{G}}|_{ub}] \tag{16}$$

where the subscript lb (ub) indicates a lower (upper) bound on the term.

Equation (16) is borrowed from [38] and indicates that bounds on the expected reward $\mathcal{R}$ can be obtained from the lower and upper bounds on the conditioned expected reward $\mathcal{R}_{\mathcal{G}}$ and a lower bound on $P(\mathcal{G})$. A lower bound on $P(\mathcal{G})$ requires some knowledge of the system in the unknown states such as bounds on transition rates, etc. Bounds on the state probabilities conditioned that the system is in the subset $\mathcal{G}$ (and, from those, bounds on $\mathcal{R}_{\mathcal{G}}$) can be obtained from the results of Courtois and Semal [9] as follows. Consider the matrix $\Pi^{(N)}$ of (5) where the last row and column is removed. (The resulting matrix $\Pi'^{(N)}$ is not stochastic.) For notational convenience, we remove the superscript (N). Let $\mathbf{v}_i$ the i^{th} normalized row of the inverse $(\mathbf{I} - \Pi')^{-1}$. It can be shown that the conditional steady state probability vector is a linear combination of the $\mathbf{v}_i$'s. Therefore, bounds on the conditioned probabilities can be derived from the vectors $\mathbf{v}_i$'s. Unfortunately, it may be impractical to obtain the inverse above using the so called direct methods if Π' is large, since we do not take advantage of the sparseness of the matrix. Iterative methods, on the other hand, are

appropriate to large and sparse matrices. In order to use an iterative method, we modify $\mathbf{\Pi}$ such that $p_{fi} = 1$ for a given i, $1 \leq i \leq N$, and let $\mathbf{\Pi}'_i$ be the resulting matrix and $\mathbf{v}_i$ the solution of $\mathbf{v}_i = \mathbf{v}_i\mathbf{\Pi}'_i$. The conditioned steady state probability vector is a linear combination of the $\mathbf{v}_i$'s and bounds on these conditioned probabilities can be derived from the $\mathbf{v}_i$'s. Clearly, the use of an iterative technique requires the solution of a large number of matrices, i.e., the solution of each $\mathbf{\Pi}'_i$. However, we can overcome this problem by adapting the approach developed in [38]. Bellow we outline this new procedure for obtaining bounds on conditioned steady state probabilities which has cheaper computational requirements than those in [9]. These bounds, however, may not be as tight as those obtained in [9], though the difference is negligible for skewed models.

Consider the transition rate matrix $\mathbf{Q}_1$ of Figure 3 that represents the generated states of the model. The first state in $\mathbf{Q}_1$ is the initial state for the generating procedure. We assume

$$\mathbf{Q}_1 = \begin{bmatrix} \mathbf{Q}_{00} & \mathbf{Q}_{01} & \mathbf{Q}_{02} \\ \mathbf{Q}_{10} & \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{20} & \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}$$

Figure 3: matrix $\mathbf{Q}_1$.

that the last state in $\mathbf{Q}_1$ (a_f) is an exact aggregation of the non-generated states and so, the transition rates out of this state are known. (Note that matrix $\mathbf{Q}_1$ is a transition rate matrix "equivalent" to $\mathbf{\Pi}$, except for the rates out of a_f that do not necessarily go to the first state.) Therefore, the exact conditioned steady state probabilities of the generated states can be obtained from the solution of $\mathbf{Q}_1$. Let $\mathcal{Q}$ be the set of states of matrix $\mathbf{Q}_1$. $\mathcal{Q}_0 = \{a_1\}$, $\mathcal{Q}_1 = \mathcal{Q} - \{\{a_1\} \cup \{a_f\}\}$ and $\mathcal{Q}_2 = \{a_f\}$. In Figure 3 the submatrix $\mathbf{Q}_{ii}$ corresponds to the set $\mathcal{Q}_i$.

We follow the idea developed in [38] and from $\mathbf{Q}_1$, we construct a matrix $\mathbf{Q}_2$ shown in Figure 4, where states in $\mathcal{Q}_1$ are replicated. As indicated in [38], the steady state probabilities

$$\mathbf{Q}_2 = \begin{bmatrix} \mathbf{Q}_{00} & \mathbf{Q}_{01} & 0 & \mathbf{Q}_{02} \\ \mathbf{Q}_{10} & \mathbf{Q}_{11} & 0 & \mathbf{Q}_{12} \\ \mathbf{Q}_{10} & 0 & \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{20} & 0 & \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}$$

Figure 4: matrix $\mathbf{Q}_2$.

of $\mathbf{Q}_1$ can be obtained from those of $\mathbf{Q}_2$, i.e., if $\pi^1\mathbf{Q}_1 = 0$, $\pi^1 =< \pi^1_0, \pi^1_1, \pi^1_2 >$ and $\pi^2\mathbf{Q}_2 = 0$, $\pi^2 =< \pi^2_0, \pi^2_{11}, \pi^2_{12}, \pi^2_2 >$ then $\pi^1_0 = \pi^2_0$, $\pi^1_1 = \pi^2_{11} + \pi^2_{12}$, $\pi^1_2 = \pi^2_2$.

We now organize the replicated states in subsets $\mathcal{F}_k$, $1 \leq k \leq K$, $K \leq N$, such that a state a_i is in $\mathcal{F}_k$ iff the minimum number of steps from a_i to the initial state $a_1 \in \mathcal{Q}_0$ is k. Reorganizing the states is not a problem. A minimum path algorithm (e.g., Dijkstra) can be used. We assume that all the generated states, i.e., the states in subset $\mathcal{Q}_0 \bigcup \mathcal{Q}_1$ communicate. If this is not true at the end of the generation procedure, the states that have transitions only to state a_f are excluded from the set of generated states. As a consequence of this reorganization, the output transition of states in $\mathcal{F}_k$ ($k > 1$) can go only to states in $\mathcal{F}_{k-1}$, or to states in $\mathcal{F}_l$ for $l > k$, or to state a_f. Furthermore, there is always one transition from a state in $\mathcal{F}_k$ to a state in $\mathcal{F}_{k-1}$. Figure 5 shows the new matrix $\mathbf{Q}'_2$ after reorganizing the states in matrix $\mathbf{Q}_2$. We note that the steady state probabilities of $\mathbf{Q}_2$ are identical to the steady state probabilities of $\mathbf{Q}'_2$ after proper matching of states.

$$\mathbf{Q}'_2 = \begin{bmatrix} \mathbf{Q}_{00} & \mathbf{Q}_{01} & 0 & 0 & \cdots & 0 & 0 & \mathbf{Q}_{02} \\ \mathbf{Q}_{10} & \mathbf{Q}_{11} & 0 & 0 & \cdots & 0 & 0 & \mathbf{Q}_{12} \\ \mathcal{F}_{10} & 0 & \mathcal{F}_{11} & \mathcal{F}_{12} & \cdots & \mathcal{F}_{1,K-1} & \mathcal{F}_{1K} & \mathcal{F}_{12} \\ 0 & 0 & \mathcal{F}_{21} & \mathcal{F}_{22} & \cdots & \mathcal{F}_{2,K-1} & \mathcal{F}_{2K} & \mathcal{F}_{22} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mathcal{F}_{K-1,K-1} & \mathcal{F}_{K-1,K} & \mathcal{F}_{K-1,2} \\ 0 & 0 & 0 & 0 & \cdots & \mathcal{F}_{K,K-1} & \mathcal{F}_{K,K} & \mathcal{F}_{K,2} \\ \mathbf{Q}_{20} & \mathbf{Q}_{21} & \mathbf{Q}'_{21} & \mathbf{Q}'_{22} & \cdots & \mathbf{Q}'_{2,K-1} & \mathbf{Q}'_{2K} & \mathbf{Q}_{22} \end{bmatrix}$$

Figure 5: matrix $\mathbf{Q}'_2$.

If submatrices $Q_{2,0}, Q_{21}, Q'_{21}, \ldots, Q'_{2,K-1}$ were null, then we could immediately apply the results of [38]. However, this is not true in our case. We proceed by constructing, from matrix $\mathbf{Q}'_2$, matrix $\mathbf{Q}_3$, where the states in subsets $\mathcal{F}_k$ $1 \leq k \leq K$ are exactly aggregated into a single state f_k, assuming that we can perform such aggregation. Later we show that exact aggregation is not necessary to obtain the bounds on conditioned probabilities. The process defined by $\mathbf{Q}_3$ is shown in Figure 6. In Figure 6 subset $\mathcal{G}'$ is the subset of generated states from the initial state a_1 and does not include state a_1. We cannot perform exact aggregation, since we do not have any knowledge of the complete transition rate matrix of the model. Therefore, in Figure 6, the unknowns are the output rates from states f_k, $1 \leq k \leq K$, and from state a_f. Furthermore, the exact value of the reward rate to be associated with state f_k, $1 \leq k \leq K$, is not known, which precludes the calculation of the exact value for $\mathcal{R}_{\mathcal{G}}$. However, Theorem 1 below indicates a way to obtain a lower bound for $\mathcal{R}_{\mathcal{G}}$. An upper bound on $\mathcal{R}_{\mathcal{G}}$ can be found in the same way.

Theorem 1 *A lower bound on $\mathcal{R}_{\mathcal{G}}$ can be obtained by solving a transition rate matrix $\mathbf{Q}_4$ obtained from $\mathbf{Q}_3$ as follows:*

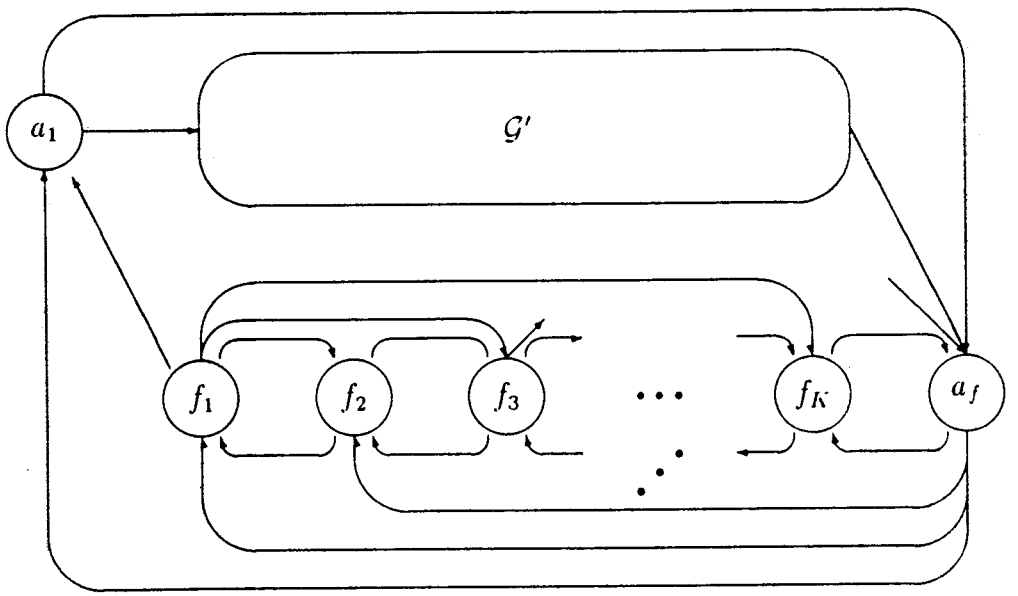

Figure 6: Process defined by matrix $\mathbf{Q}_3$.

- *Remove all transitions from state s_f and replace them by a single transition to state f_K.*
- *Replace the unknown transition values from state f_k to f_l $l > k$ (for all $1 \leq k \leq K$) (or to state a_f) by the maximum sum of transition rates from a state in $\mathcal{F}_k$ to states in $\mathcal{F}_l$ (or to state a_f).*
- *Replace the unknown transition values from f_k to f_{k-1} (for all $1 \leq k \leq K$) (or to state a_1) by the minimum value of the sum of all transition rates from a state in $\mathcal{F}_k$ to states in $\mathcal{F}_{k-1}$ (or to a_1).*
- *Assign a reward r_{lb} to each state f_k for all $1 \leq k \leq K$.*

Proof: the proof is given in the appendix.

Since we know the values of all output transition rates from the states in $\mathcal{F}_k$ for all k, obtaining the maximum and minimum values as indicated by Theorem 1 is not a problem. Furthermore, by the way we assign states to subsets $\mathcal{F}_k$, there is always a transition from a state in $\mathcal{F}_k$ to a state in $\mathcal{F}_{k-1}$ (or to a_1) and thus the minimum is guaranteed to be greater than zero. The rate assigned to the transitions from a_f to f_K is irrelevant for the final calculation of the conditional probabilities of the remaining states.

In order to simplify further the solution, we can exactly aggregate the states f_k, independently of the other states. This is true since the only return to this set of states is from state a_f to f_K. Because the submatrix formed by the states f_k is upper Hessemberg, the conditioned steady state probabilities of being in states f_k can be easily obtained and from those, the transition rates of the aggregate state representing the states f_k are calculated. After this aggregation, we remove state a_f such that all transitions to this state go to the new aggregated state. Note that the final solution is not affected when a_f is removed. Let us call this new matrix $\mathbf{Q}_5$ and from this we obtain the randomized matrix $\mathbf{P}_5$. The final probability matrix $\mathbf{P}_5$ used to obtain the value of $\mathcal{R}_\mathcal{G}$ is given in Figure 7. Note that $\mathbf{P}_5$ is identical to $\mathbf{\Pi}$, except by the value of p_{f1} which is obtained after aggregating the states f_k in matrix $\mathbf{Q}_4$. The desired lower bound is obtained by assigning a reward r_{lb} to state a_f in Figure 7 and an upper bound is obtained by assigning a reward r_{ub} to a_f.

$$\mathbf{P}_5 = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n} & p_{1,f} \\ p_{21} & p_{22} & \cdots & p_{2,n} & p_{2,f} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} & p_{n,f} \\ p_{f1} & 0 & \cdots & 0 & 1-p_{f1} \end{bmatrix}$$

Figure 7: Matrix $\mathbf{P}_5$.

5 Examples.

In this section, we present examples to illustrate some of the issues concerning the generation procedure.

The first is a "toy" example which shows the difference between the search rule based on visit ratios and the one based on MTTU as outline in section 4.1. Consider a model with six states that has the transition probabilities given by the table bellow. The steady state probabilities are also given in the table.

state	state	trans. prob.	steady state prob.
a_1	a_2	0.60	$\pi_1 = 0.2017$
a_1	a_3	0.40	
a_2	a_4	0.95	$\pi_2 = 0.1210$
a_2	a_5	0.05	
a_3	a_5	0.40	$\pi_3 = 0.3067$
a_3	a_1	0.60	

state	state	trans. prob.	steady state prob.
a_4	a_5	0.95	$\pi_4 = 0.1150$
a_4	a_6	0.05	
a_5	a_6	0.05	$\pi_5 = 0.2380$
a_5	a_3	0.95	
a_6	a_1	1.00	$\pi_6 = 0.0176$

Starting from state a_1, if we use the search rule based on visit ratios, then the states are generated in the order $a_2 \rightarrow a_4 \rightarrow a_5 \rightarrow a_3 \rightarrow a_6$, which is not the desired order of decreasing steady state probabilities. However, using the rule based on the MTTU, the most probable states are generated first (i.e., $a_3 \rightarrow a_5 \rightarrow a_2 \rightarrow a_4 \rightarrow a_6$). For large models, we found that both rules perform well in general, though the number of states M found by using the MTTU rule is always smaller that N, the number found using the visit ratio rule, and when the same stopping criterion is used. This indicates that the MTTU perform better as we anticipate.

We now consider a very large model of a communication protocol: the *Abacadabra protocol* [27] which is defined by ISO to compare different formal description techniques, and has similar characteristics to many of the standard protocols.

There are two layers in the protocol: the bottom layer offers a connectionless transmission service which is used by the subsequent layer that offers a connection oriented full duplex service to the user. The formal specification of this protocol is lengthy and we omit it from the text for conciseness. The structure of the specification is shown in Figure 8. The number

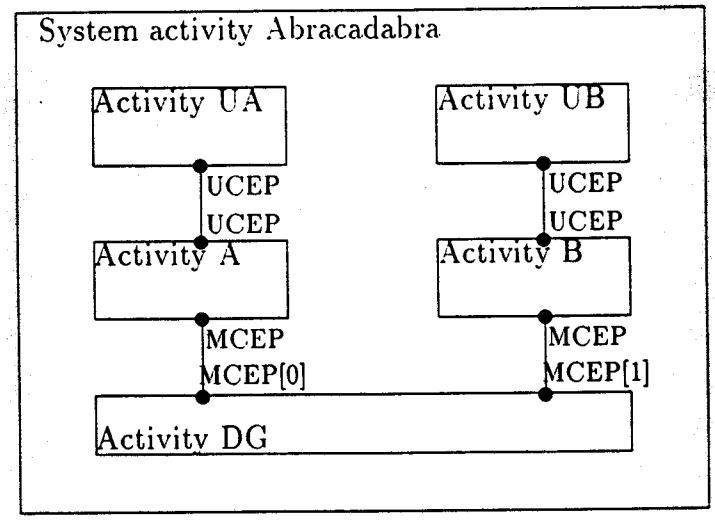

Figure 8: The specification structure of the *Abacadabra* protocol.

of states in this protocol is very large and there are approximately over 10^{12} possible states in the model. We used the Estelle [28] formal description language to specify the protocol and from the description, the states of the Markov model of the protocol are generated by using the iterative procedure outlined in section 3. If we specify a value of 10^5 units of time to the MTTU, 257 states are generated. By verifying the meaning of those states, we can see that they correspond to the opening and shutdown of a connection without error, and subsequent data transfer. The table below shows the data throughput values for different number of states in the generation procedure. It can be observed that the throughput converges fast.

Number of states	Throughput ($\times 10^{-3}$)
100	8.20
120	40.87
140	40.93
160	41.76
180	41.79

When we use the modified algorithm outlined in section 4.1, the number of generated states (M) drops to 187 with the same value of the MTTU as above. This corroborates our conjecture that $M < N$ in section 4.1. From the results of section 4.2 we can also obtain bounds for the throughput values above, *conditioned* that the protocol is in the generated scenario (open/shutdown and data transfer). We are in the process of automating the bounds calculation, and so we do not have the results for this example yet. However, we obtain bounds for the example below which has a simpler structure.

The last example is an availability model taken from the save manual [19]. The example is a model of a fault-tolerant database system with the following components: a front-end, a database and two sets of modules, each formed by two processors connected to a memory unit by a switch. Unlike the example in [19], we assume that the components can fail even when the system is down. The failure rate for the front-end, the database, the switches and memory units is 1/2400, and the processors fail with a rate of 1/120. All components are repaired (with a rate of 1) by a single repairman who gives the highest (preemptive) priority to the front-end and the database, followed by the switches and memory units and the processors have the smallest priority. If a processor fails, it may affect the database with probability c.

In this example, the "natural" order of generating the most probable states is by increasing number of failures, since we expect that states with relatively low number of components failed are the most probable. However, this may not correspond to the "most probable states" order. When $c = 0.99$, the most probable states after the state with no failures are the two which represent a failure of one processor in a module (one state for each module), followed by the state which represents two processors failed, one in each module. Therefore, a state with two components failed concentrates more probability mass than the other states

with only one component failed. When $c = 0.90$, the "natural" order performs even worse. Since we increase the probability of having a database failure when a processor fails, more states with i components failed concentrates more probability mass than those with j components failed, for $j < i$. In order to obtain better bounds for the measures of interest, when we truncate the state space, it is important that we generate the most probable states.

When we specify a value of $MTTU = 10^5$ for stopping the generation of states, 31 states are obtained when $c = 0.99$ and 42 when $c = 0.90$. The entire state space for this example has 576 states. We first calculate a transient measure, say the probability that the system (with $c = 0.90$) remains operational for a percentage y of the observation period t equal to 360 units of time ($P[O(t) > y]$). Using the method of [13] and equation (14) to calculate the bounds, we obtain:

y	$P^{lb}[O(t) > y]$	$P^{ub}[O(t) > y]$
0.98	0.986	0.989
0.99	0.880	0.888

We now consider the calculation of steady state availability Av. This is possible, since we have knowledge of the behavior of the system once in the non-generated states. If the states are generated in increasing order of number of failed components, then the results in [38] can be used to calculate bounds on Av. However, as we mention above, this order is not the best order to obtain the most probable states. As a consequence, the "state duplication procedure" outlined in [38] is not directly applicable to obtain bounds on conditional availability and, from those, bounds on Av. This can be seen by observing Figure 6. In that Figure, there may be several transition rates from state a_f to the aggregate states f_k representing the duplicated states, unlike the single transition from a_f to f_K, required for the procedure in [38]. (Note that if the states are generated in order of increasing number of failures, this problem does not arise, since there would be only a single transition from a_f to f_K.) However, we can invoke Theorem 1 to obtain bounds on conditional availability. From equation (16 we also need a lower bound on $P(\mathcal{G})$, the probability that the system is in a state of those generated, in order to calculate Av. The "maximum holding time" Lemma in [38] provides this lower bound. Applying these results, we obtain the following bounds on Av:

c	number of states generated	lower bound	upper bound	exact value (576 states)
0.99	31	0.998823	0.998840	0.998834
0.90	42	0.995861	0.995878	0.995873

6 Conclusions.

We have developed an approach to generate efficiently a subset of the state space of a Markov model with the aim to obtain the most probable states in the model. The approach is based on the work of Dimitrijevic and Chen [16]. The iterative method we propose is shown to be significantly more efficient than previous methods, both in terms of storage and number of operations. In this new method, the sparseness of the transition rate matrix of the Markov model is preserved, which is important to solve large models. The examples show that the method works satisfactorily for models with skewed probabilities.

We have addressed the problem of bounding reward measures based on the states generated. Even if there is no knowledge about the non-generated states, bounds on transient reward measures can be easily obtained. For steady state, we have used the state replication technique in [38] and showed how to obtain (for general models) bounds on reward measures conditioned that the system is in the subset of the generated states. Unlike Courtois and Semal bounds [9], ours can be calculated while preserving the sparseness of the generated transition rate matrix. Bounds on conditional state probabilities are important to evaluate performability measures conditioned that the system is in the generated, most probable, scenario. They can also be used to evaluate unconditioned measures, when we have more knowledge about the behavior of the model in the non-generated states.

Several extensions are possible to increase the efficiency of the algorithm. In one of them, sets of states are chosen instead of one at a time. In another, sets of generated states are aggregated and the aggregated process is used for the choice of the next state, instead of using the entire unaggregated set of generated states. We are currently investigating these and other approaches.

We have implemented our method in a tool and used it to generate states of communication protocols described in Estelle. We hope the method will be useful in other specification languages, such as Stochastic Petri-net. The method to bound conditioned measures is in the process of being automated.

Acknowledgments

We wish to thank R.R. Muntz and J.C. Guedes for reading and comment on early versions of this manuscript. We also thank H.R. Gail and R.R. Muntz for enlightening discussions which led to the state reorganization procedure in section 4.2.

APPENDIX

A Proof of Theorem 1.

Before proving Theorem 1, we need the following Lemmas.

Lemma 1 *A lower bound on $\mathcal{R}_{\mathcal{G}}$ can be calculated from the solution of $\pi^{(k)}\mathbf{Q}_3^{(k)} = 0$, where $\mathbf{Q}_3^{(k)}$ is identical to $\mathbf{Q}_3$, except that all the transition rates out of state a_f go to f_k, and k is one of the possible values between 1 and K.*

Proof: Consider matrix $\mathbf{Q}_2'$ in Figure 5. $\mathcal{R}_{\mathcal{G}}$ is given by:

$$\mathcal{R}_{\mathcal{G}} = \sum_{i=1}^{N} r_i[\beta_i^2 + \beta_{h(i)}^2] \tag{17}$$

where N is the number of generated states, β_i^2 is equal to the i^{th} element of of π^2 solution of $\pi^2\mathbf{Q}_2' = 0$, normalized so that the sum of all the states of $\mathbf{Q}_2'$ except state a_f is one, $h(i)$ is a function that maps a state $a_i \in \mathcal{G}'$ to an equivalent (replicated) state in the set $\{\mathcal{F}_1 \bigcup \ldots \bigcup \mathcal{F}_K\}$. Let us assume that the states in each set $\mathcal{F}_k$ $(1 \leq k \leq K)$ are exactly aggregated and we obtain the process of Figure 6. Obviously, a lower bound $\mathcal{R}_{\mathcal{G}}^1$ on $\mathcal{R}_{\mathcal{G}}$ can be found by assigning a reward rate r_{lb} to each state f_k $(1 \leq k \leq K)$.

$$\mathcal{R}_{\mathcal{G}}^1 = \sum_{i=1}^{N} r_i\beta_i^3 + r_{lb}\sum_{k=1}^{K}\beta_{N+k}^3 \leq \mathcal{R}_{\mathcal{G}} \tag{18}$$

where β_i^3 is equal to the i^{th} element of of π^3 solution of $\pi^3\mathbf{Q}_3 = 0$ normalized as above. We cannot find exactly the state probability vectors π^2 or π^3, since we do not know the values of $\mathbf{Q}_{20}, \mathbf{Q}_{21}, \mathbf{Q}_{21}', \ldots, \mathbf{Q}_{2K}'\mathbf{Q}_{22}$ in Figure 5 or, equivalently, the output rates from state a_f in Figure 6. However,

$$\begin{aligned}\mathcal{R}_{\mathcal{G}}^1 &= \sum_{i=1}^{N} r_i\beta_i^3 + r_{lb}[1 - \sum_i \beta_i^3] \\ &= r_{lb} + \sum_{i=1}^{N}\beta_i^3[r_i - r_{lb}]\end{aligned} \tag{19}$$

A lower bound $\mathcal{R}_{\mathcal{G}}^2 \leq \mathcal{R}_{\mathcal{G}}^1$ can be found if we obtain a lower bound on β_i^3, i.e., the normalized steady state probabilities of the process of Figure 6. Let $\pi^{(k)}$ be the solution of $\pi^{(k)}\mathbf{Q}_3^{(k)} = 0$. Let $\beta^{(k)}$ be the normalized $\pi^{(k)}$ as above. From the results of Courtois and Semal [9] we

know that β^3 is a linear combination of the $\beta^{(k)}$'s $(1 \leq k \leq K)$. Since the only entry in set $\mathcal{G} = \{a_1\} \bigcup \mathcal{G}'$ is through state a_1, the steady state probabilities of being in any state in set $\mathcal{G}$, conditioned on the system being in $\mathcal{G}$ is not altered when we change the transition from a_f to f_k to another state f_l or to a_1. This implies that $< \beta_1^{(l)}, \ldots, \beta_N^{(l)} >= C(k) < \beta_1^{(k)}, \ldots, \beta_N^{(k)} >$ for any $l \neq k$, where $C(k)$ is value that depends on k. Therefore, one of the K matrices $\mathbf{Q}^{(k)}$ will give the desired lower bound.

Lemma 2 *Consider a process represented by Figure 9 with a corresponding transition probability matrix* $\mathbf{P}$ *given in Figure 10. In this process,* $p_{m,k} \geq 0$ *for* $k \geq m-1$, $p_{m,m-i} = 0$ *for* $i > 1$ *and for all values of* $m \leq n-1$; $p_{n,l} \neq 0$, $p_{n,k} = 0$, $k \neq l$ *and* $k \neq n$. *Let* π_0 *be the first entry of* π, *the solution of* $\pi = \pi\mathbf{P}$. *If we vary* l, $1 \leq l \leq n$, *the minimum value of* π_0 *occurs when* $l = n-1$.

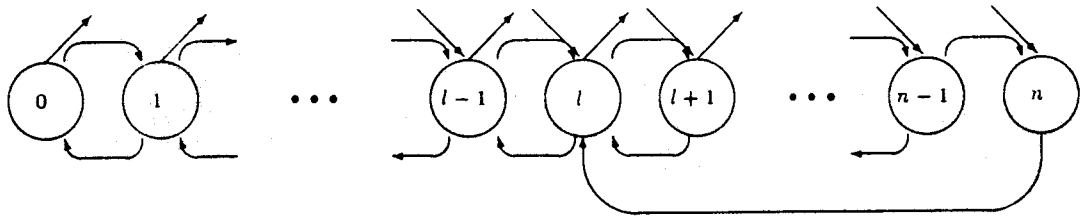

Figure 9: Process of Lemma 2.

$$\mathbf{P} = \begin{bmatrix}
p_{00} & p_{01} & p_{02} & \cdots & p_{0,l-1} & p_{0l} & p_{0,l+1} & \cdots & p_{0,n-1} & p_{0,n} \\
p_{10} & p_{11} & p_{12} & \cdots & p_{1,l-1} & p_{1l} & p_{1,l+1} & \cdots & p_{1,n-1} & p_{1,n} \\
0 & p_{21} & p_{22} & \cdots & p_{2,l-1} & p_{2l} & p_{2,l+1} & \cdots & p_{2,n-1} & p_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & p_{l-1,l-1} & p_{l-1,l} & p_{l-1,l+1} & \cdots & p_{l-1,n-1} & p_{l-1,n} \\
0 & 0 & 0 & \cdots & p_{l,l-1} & p_{ll} & p_{l,l+1} & \cdots & p_{l,n-1} & p_{l,n} \\
0 & 0 & 0 & \cdots & 0 & p_{l+1,l} & p_{l+1,l+1} & \cdots & p_{l+1,n-1} & p_{l+1,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & p_{n-1,n-1} & p_{n-1,n} \\
0 & 0 & 0 & \cdots & 0 & p_{n,l} & 0 & \cdots & 0 & p_{n,n}
\end{bmatrix}$$

Figure 10: matrix of Lemma 2.

Proof: Let $l = n-1$. Then the matrix $\mathbf{P}$ is upper Hessemberg and the solution satisfies:

$$\pi_m^{(n-1)} p_{m,m-1} = \sum_{i=0}^{m-1} \pi_i^{(n-1)} \sum_{j=m}^{n} p_{ij} \tag{20}$$

where the superscript indicates that $p_{n,n-1} \neq 0$. From (20) it is easy to see that $\pi_m^{(n-1)}$ can be obtained by first calculating recursively the unnormalized values $\gamma_m^{(n-1)}$ from the values of $\gamma_0^{(n-1)}, \ldots \gamma_{m-1}^{(n-1)}$, assuming $\gamma_0^{(n-1)} = 1$. The final normalized value of $\pi_m^{(n-1)}$ is calculated by dividing $\gamma_m^{(n-1)}$ by the normalization constant $G^{(n-1)} = \sum_{i=0}^{n} \gamma_i^{(n-1)}$.

Let $p_{n,l} \neq 0$. For $m \leq l$, equation (20) is satisfied:

$$\pi_m^{(l)} p_{m,m-1} = \sum_{i=0}^{m-1} \pi_i^{(l)} \sum_{j=m}^{n} p_{ij} \tag{21}$$

Therefore, if $\gamma_0^{(l)} = 1$, $\gamma_m^{(n-1)} = \gamma_m^{(l)}$ $m \leq l$. For $m = l + 1$:

$$\begin{aligned} \gamma_{l+1}^{(l)} p_{l+1,l} &= \sum_{i=0}^{l} \gamma_i^{(l)} \sum_{j=l+1}^{n} p_{ij} - \gamma_n^{(l)} p_{n,l} \\ &= \gamma_{l+1}^{(n-1)} p_{l+1,l} - \gamma_n^{(l)} p_{n,l} \end{aligned} \tag{22}$$

Since $\gamma_n^{(l)} > 0$, $\gamma_{l+1}^{(l)} < \gamma_{l+1}^{(n-1)}$.

Now assume $\gamma_m^{(l)} < \gamma_m^{(n-1)}$ for a given value of m, $l+1 < m < n$, i.e., $\gamma_m^{(l)} = \gamma_m^{(n-1)} - c(m)$, where $c(m)$ is a positive value dependent on m.

$$\begin{aligned} \gamma_{m+1}^{(l)} p_{m+1,m} &= \sum_{i=0}^{m} \gamma_i^{(l)} \sum_{j=m+1}^{n} p_{ij} - \gamma_n^{(l)} p_{n,l} \\ &= \sum_{i=0}^{l} \gamma_i^{(l)} \sum_{j=m+1}^{n} p_{ij} + \sum_{i=l+1}^{m} \gamma_i^{(l)} \sum_{j=m+1}^{n} p_{ij} - \gamma_n^{(l)} p_{n,l} \\ &= \sum_{i=0}^{l} \gamma_i^{(n-1)} \sum_{j=m+1}^{n} p_{ij} + \sum_{i=l+1}^{m} [\gamma_i^{(n-1)} - c(i)] \sum_{j=m+1}^{n} p_{ij} - \gamma_n^{(l)} p_{n,l} \\ &= \sum_{i=0}^{m} \gamma_i^{(n-1)} \sum_{j=m+1}^{n} p_{ij} - [\sum_{i=l+1}^{m} c(i) \sum_{j=m+1}^{n} p_{ij}] - \gamma_n^{(l)} p_{n,l} \\ &< \gamma_{m+1}^{(n-1)} p_{m+1,m} \end{aligned} \tag{23}$$

Similarly to above, for $m = n$ $\gamma_n^{(l)} p_{nl} < \gamma_n^{(n-1)} p_{nl}$ In summary:

$$\begin{aligned} \gamma_m^{(l)} &= \gamma_m^{(n-1)} \quad m \leq l \\ \gamma_m^{(l)} &< \gamma_m^{(n-1)} \quad l < m \leq n \end{aligned} \tag{24}$$

Since $G^{(l)} = \sum_{i=0}^{n} \gamma_m^{(l)}$, $G^{(l)} < G^{(n-1)}$. Therefore, $\pi_0^{(l)} = [\gamma_0^{(l)}]/[G^{(l)}] = [\gamma_0^{(n-1)}]/[G^{(l)}] > [\gamma_0^{(n-1)}]/[G^{(n-1)}] = \gamma_0^{(n-1)}$.

Lemma 3 *The value of k which minimizes simultaneously all the normalized state probabilities* $< \beta_1^{(k)}, \ldots, \beta_N^{(k)} >$ *defined in Lemma 1 is* $k = K$.

Proof: Let us aggregate all states in the set $\mathcal{G}$, of the process given by transition matrix $\mathbf{Q}_3^{(k)}$ defined in Lemma 1. Figure 11 shows the resulting process, and $a_{\mathcal{G}}$ is the aggregated state. Since the process in Figure 11 is identical to the process of Figure 9, we can invoke

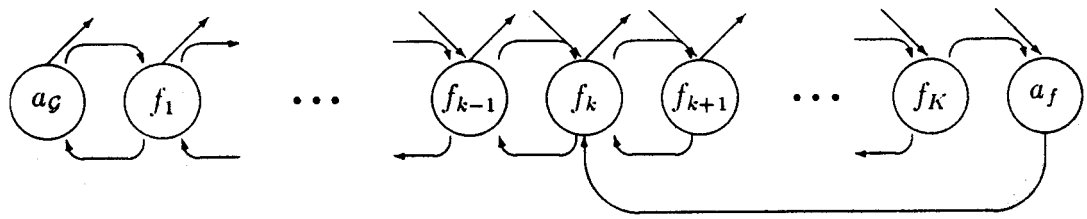

Figure 11: Process defined by matrix $\mathbf{Q}_3^{(k)}$ where states in $\mathcal{G}$ are aggregated.

Lemma 2 which shows that $\pi_{\mathcal{G}}$ is minimized when $k = K$.

In the proof of Lemma 1 we establish that $< \beta_1^{(l)}, \ldots . \beta_N^{(l)} >= C(k) < \beta_1^{(k)}, \ldots . \beta_N^{(k)} >$ for any $l \neq k$. Let α_i be the steady state probability of being in state $a_i \in \mathcal{G}$ conditioned on the system being in $\mathcal{G}$. Then, $< \beta_1^{(k)}, \ldots, \beta_N^{(k)} >= \pi_{\mathcal{G}} < \alpha_1, \ldots, \alpha_N >$. When k varies, $\pi_{\mathcal{G}}$ varies, but the value of each α_i remains constant. From Lemma 2 $\pi_{\mathcal{G}}$ is minimized when $k = K$.

Proof of Theorem 1.

From Lemmas 1 and 3, the $\beta_i^{(k)}$'s are minimized when the single transition out of state a_f goes to state f_K. It remains to find the transition rates among the aggregate states f_k. Consider the process shown in Figure 11 and assume that the single transition from state a_f goes to f_K. From this process, construct a new one where the transitions from state f_m to f_l for $l > m$ and from f_m to a_f are replaced by upper bounds, and each transition from f_m to f_{m-1} for $1 < m \leq K$ and from f_1 to a_1 is replaced by a lower bound. Let $\pi_{\mathcal{G}}$ be the state probability as defined in Lemma 2 and $\pi'_{\mathcal{G}}$ is the corresponding state probability in the modified process. Lemma 2 of [38] shows that $\pi'_{\mathcal{G}} \leq \pi_{\mathcal{G}}$, which proves the theorem if we can find upper and lower bounds on the transition rates. The maximum value of the sum of transition rates from a state in $\mathcal{F}_m$ to states in $\mathcal{F}_l$ (or to a_f) is an upper bound on the transition rate from f_m to f_l $l > m$ (or to a_f). The minimum value of the sum of all transitions from a state in $\mathcal{F}_m$ to a state in $\mathcal{F}_{m-1}$ (or from a state in $\mathcal{F}_1$ to a_1) is a lower bound on the transition rate from f_m to f_{m-1} (or from f_1 to a_1).

References

[1] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation. Numerical Methods.* Prentice-Hall, 1989.

[2] J.A. Carrasco and J. Figueras. METFAC: design and implementation of a software tool for modeling and evaluation of complex fault-tolerant computing systems. In *Proceedings of FTCS-16*, pages 424–429, 1986.

[3] E. Çinlar. *Introduction to Stochastic Processes.* Prentice-Hall, 1975.

[4] G. Chiola. A software package for the analysis of generalized stochastic Petri net models. In *Proceedings of the International Workshop on Timed Petri-nets*, pages 136–143, 1985.

[5] S.N. Chiou and V.O.K. Li. Reliability analysis of a communication network with multi-mode components. *IEEE Journal on Selected Areas in Communications*. SAC-4:1156–1161, 1986.

[6] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proceedings of the Third International Workshop on Petri-nets and Performance Models*, pages 142–151, 1989.

[7] A.E. Conway and N.D. Georganas. *Queueing Networks - Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation.* MIT Press, 1989.

[8] P.J. Courtois. *Decomposability: Queueing and Computer System Applications.* Academic Press, 1977.

[9] P.J. Courtois and P. Semal. Bounds for the positive eigenvectors of nonnegative matrices and for their approximations. *Journal of the ACM*, 31:804– 825, 1984.

[10] P.J. Courtois and P. Semal. Bounds for transient characteristics of large or infinite Markov chains. In *The First International Conference on the Numerical Solution of Markov Chains*, pages 446–468, 1990.

[11] P.J. Courtois and P. Semal. Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models. *IEEE Journal on Selected Areas in Communications*, SAC-4(6):926–937, 1986.

[12] 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.

[13] 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.

[14] E. de Souza e Silva and H.R. Gail. *Performability Analysis of Computer Systems: from Model Specification to Solution.* Technical Report RC 16961, T.J. Watson Research Center, 1991.

[15] E. de Souza e Silva and R.R. Muntz. *Stochastic Analysis of Computer and Communication Systems*, chapter Queueing Networks: Solutions and Applications, pages 319–399. North-Holland, 1990.

[16] D.D. Dimitrijevic and M.-S. Chen. Dynamic state exploration in quantitative protocol analysis. In *Protocol Specification, Testing, and Verification, IX*, pages 327–338, North-Holland, 1990.

[17] D.D. Dimitrijevic and M.-S. Chen. An integrated algorithm for probabilistic protocol verification and evaluation. In *Proceedings of INFOCOM'89*, 1989.

[18] G.H. Golub and C.F. van Loan. *Matrix Computation.* Johns Hopkins University Press, second edition, 1989.

[19] A. Goyal. *System Availability Estimator (SAVE). User's Manual.* IBM, 1988.

[20] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, and K.S. Trivedi. The system availability estimator. In *Proceedings of FTCS-16*, pages 84–89, 1986.

[21] A. Goyal, S.S. Lavenberg, and K.S. Trivedi. Probabilistic modeling of computer system availability. *Annals of Operations Research*, 8:285–306, 1987.

[22] W.K. Grassman. *Rounding Errors in some Recursive Methods used in Computational Probability.* Technical Report 73, Dept. of Operations Research, 1983.

[23] W.K. Grassmann. Finding transient solutions in Markovian event systems through randomization. In *The First International Conference on the Numerical Solution of Markov Chains*, pages 375–395, 1990.

[24] D. Gross and D.R. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):343–361, 1984.

[25] B.R. Haverkort, I.G. Niemegeers, and P. Veldhuyzen van Zanten. DyQNtool - a performability modelling tool based on the dynamic queueing network concept. In *Proceedings of the Fifth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1991.

[26] D.P. Heyman and M.J. Sobel. *Stochastic Models in Operations Research, Volume I.* McGraw-Hall, 1982.

[27] ISO/SC21/WG1/FDT-A. *Guidelines for the Application of Estelle, LOTUS and SDL.* Technical Report Project/97.21.9/Q48.2 CCITT/SG/X/Q7, ISO, 1987.

[28] ISO/TC97/SC21/WG1/DIS9074. *Estelle - A Formal Description Technique Based on an Extended State Transition Model.* ISO, 1987.

[29] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavsk Aktuarietidskrift*, 36:87–91, 1953.

[30] A.M. Johnson Jr. and M. Malek. Survey of software tools for evaluating reliability, availability and serviceability. *ACM Computing Surveys*, 20:227–271, 1988.

[31] J.G. Kemeny and J.L. Snell. *Finite Markov Chains, 2nd Edition.* Springer-Verlag, 1987.

[32] S. S. Lavenberg, editor. *Computer Performance Modeling Handbook.* Academic Press, New York, 1983.

[33] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models.* Prentice Hall, Englewood Cliffs, NJ, 1984.

[34] V.O.K. Li and J.A. Silvester. Performance analysis of networks with unreliable components. *IEEE Trans. on Communications*, COM-32(10):1105–1110, 1984.

[35] N.F. Maxemchuck and K. Sabnani. Probabilistic verification of communication protocols. In *Proceedings of Protocol Specification, Testing, and Verification VII*, pages 307–319, 1987.

[36] J.F. Meyer. On evaluating the performability of degradable computing systems. *IEEE Trans. on Computers*, C-29(8):720–731, 1980.

[37] J.F. Meyer. *Performability: A Retrospective and Some Pointers to the Future.* Technical Report, University of Michigan, 1991.

[38] R.R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. *IEEE Trans. on Computers*, 38(12):1714–1723, 1989.

[39] R.R. Muntz and J.C.S. Lui. Evaluating bounds on steady state availability from Markov models of repairable systems. In *The First International Conference on the Numerical Solution of Markov Chains*, pages 469–489, 1990.

[40] T.W. Page Jr., S.E. Berson, W.C. Cheng, and R.R. Muntz. An object-oriented modeling environment. *ACM Sigplan Notices (Proceedings OOPSLA '89)*, 24(10):287–296, 1989.

[41] W.H. Sanders and J.F. Meyer. METASAN: a performability evaluation tool based on stochastic activity networks. In *Proceedings of the 1986 Fall Joint Computer Conference*, pages 807–816, 1986.

[42] W.H. Sanders and J.F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications*. 9(1):25–36, 1991.

[43] H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[44] W. Stewart and A. Goyal. *Matrix Methods in Large Dependability Models.* Technical Report, T.J. Watson Research Center, 1985.

[45] K.S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications.* Prentice-Hall, 1982.

[46] N. van Dijk. The importance of bias-terms for error bounds and comparison results. In *The First International Conference on the Numerical Solution of Markov Chains*, pages 640–663, 1990.

[47] C.-L. Yang and P. Kubat. Efficient computation of most probable states for communication networks with multimode components. *IEEE Trans. on Communications*, 37(5):535–538, 1989.

[48] D.M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, 1971.