Exploring States of Sparse, Large Markov Chains

Tuğrul Dayar[1] and William J. Stewart [2]

Abstract. Dynamic state exploration is an effective technique to explore states of sparse, large Markov chains having highly unbalanced stationary probabilities and to compute related performance measures for them. States of interest are explored starting from a particular state until a predetermined stopping criterion is met. In this paper, the dynamic state exploration idea is extended to the case where more than one starting state is considered simultaneously. This approach easily lends itself to parallel implementation and reduces the number of operations to be carried out for each starting state. Furthermore, it is shown that this approach gives quite an accurate feeling for the nature of the states without having to solve the complete Markov chain. Experiments are carried out on a Markov chain arising from an Asynchronous Transfer Mode (ATM) traffic queue.

1 Introduction

The class of stochastic processes named after A. A. Markov (1856-1922) has been of considerable interest and use to scientists from many disciplines. What is referred to as a Markov process arises in queueing network analysis, large scale economic modeling, reliability analysis, computer system performance evaluation and other areas.

A stochastic process with a discrete state space becomes a *Markov chain* if the next state visited by the system depends only on the current state and not on any previous states. In other words, a Markov chain restricts the time between two consecutive state changes to be *memoryless*. All dependencies on the history of the process are removed. If the instants at which state changes (transitions) occur are finite or countable, then we have a *discrete-time parameter*. Whereas, if state changes may occur at any point in time, we have a *continuous-time parameter*. Moreover, a stochastic process is said to be *time homogeneous* if the system state does not depend on the value of the time parameter. Alternatively, in time homogeneous Markovian systems, the next system state depends only on the current system state, and it is independent of the value of the current time instant. For a continuous-time Markov chain (CTMC), the memoryless property requires that the time between consecutive state changes be exponentially distributed. On the other hand, time between two state changes for a discrete-time Markov chain (DTMC) needs to be geometrically distributed. Throughout this paper we concentrate on time homogeneous stochastic processes that are discrete-state, finite in the state space, discrete- or continuous-time, and Markovian.

In what follows, boldface capital letters denote matrices, boldface lowercase letters denote column vectors, lowercase letters denote scalars, calligraphic letters denote sets.

A finite DTMC over the state space $\mathcal{S}$ may be represented by a *one-step transition probability matrix*, $\boldsymbol{P}$, of order n ($= \#(\mathcal{S})$: number of states in $\mathcal{S}$). The $(i,j)^{th}$ element of $\boldsymbol{P}$, denoted $p_{i,j}$, is the one-step transition probability of going from state i to state j. An analogous definition applies to the case of a CTMC. For a CTMC, one can define $\boldsymbol{Q}$ as the *transition rate matrix* (or the *infinitesimal generator matrix* of the Markov chain). The $(i,j)^{th}$ element of $\boldsymbol{Q}$ is the rate at which the system makes a transition from state i to state j. Each diagonal element of $\boldsymbol{Q}$ is the negated sum of the corresponding off-diagonal elements (i.e., $q_{i,i} = -\sum_{j \neq i} q_{i,j}$).

Any $\boldsymbol{\pi}$ satisfying

$$\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}, \quad \sum_{i \in \mathcal{S}} \pi_i = \|\boldsymbol{\pi}\|_1 = 1 \tag{1.1}$$

[1]Department of Computer Science and Information Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey
[2]Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA

for a DTMC, or

$$0 = \boldsymbol{\pi}\boldsymbol{Q}, \quad \|\boldsymbol{\pi}\|_1 = 1 \tag{1.2}$$

for a CTMC, is called a *stationary probability distribution.* A Markov chain initialized with a stationary distribution follows this distribution at all points in time. In what follows, 0 is a row or column vector of all zeros depending on the context in which it is used.

Equation (1.1) ((1.2)) is an eigensystem and we seek the unit left-hand eigenvector of $\boldsymbol{P}$ ($\boldsymbol{Q}$) corresponding to eigenvalue 1 (0). Another way of posing the problem is to rewrite (1.1) in the form

$$\boldsymbol{\pi}(\boldsymbol{P} - \boldsymbol{I}) = 0, \quad \|\boldsymbol{\pi}\|_1 = 1 \tag{1.3}$$

and view it as a homogeneous system of linear equations with singular coefficient matrix $\boldsymbol{P} - \boldsymbol{I}$ and unknown vector $\boldsymbol{\pi}$. Equation (1.2) is already in form (1.3); it is a homogeneous system of linear equations with singular coefficient matrix $\boldsymbol{Q}$ an unknown vector $\boldsymbol{\pi}$.

An important class of Markovian systems that come up in various areas of modeling and analysis is the *nearly completely decomposable* (NCD) Markov chains. These are finite stochastic irreducible (meaning each state is reachable from every other state in the chain) matrices. But most importantly, the matrix of transition probabilities can be ordered in such a way that it has a block structure in which the nonzero elements of the off-diagonal blocks are small compared to those of the diagonal blocks ([10]).

$$\boldsymbol{P}_{n\times n} = \begin{array}{c} \begin{array}{cccc} n_1 & n_2 & \cdots & n_N \end{array} \\ \begin{pmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} & \cdots & \boldsymbol{P}_{1,N} \\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} & \cdots & \boldsymbol{P}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{N,1} & \boldsymbol{P}_{N,2} & \cdots & \boldsymbol{P}_{N,N} \end{pmatrix} \end{array} \begin{array}{c} n_1 \\ n_2 \\ \vdots \\ n_N \end{array} \tag{1.4}$$

The subblocks $\boldsymbol{P}_{i,i}$ are square, of order n_i, with $n = \sum_{i=1}^{N} n_i$. Let the stationary distribution of $\boldsymbol{P}$, $\boldsymbol{\pi}$ (i.e., $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$, $\|\boldsymbol{\pi}\|_1 = 1$), be partitioned conformally with $\boldsymbol{P}$ such that $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_N)$. Each $\boldsymbol{\pi}_i$, $i = 1, 2, \ldots, N$, is a row vector having n_i elements. Let $\boldsymbol{P} = diag(\boldsymbol{P}_{1,1}, \boldsymbol{P}_{2,2}, \ldots, \boldsymbol{P}_{N,N}) + \boldsymbol{E}$. The quantity $\|\boldsymbol{E}\|_\infty$ is referred to as the *degree of coupling* and it is taken to be a measure of the decomposability of the matrix (see [6]). If it were zero, then $\boldsymbol{P}$ would be reducible.

Small perturbations in the transition probabilities of NCD Markov chains may lead to considerable changes in the stationary probabilities; therefore, they are known to be ill-conditioned. When the system under consideration is nearly completely decomposable, there are eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods ([12]). Accordingly, different techniques, one of which is iterative aggregation-disaggregation (IAD), have been developed. IAD algorithms do not suffer from these limitations ([8, 19, 1, 18, 17, 3]).

The idea in IAD methods is to observe the system in isolation in each of the diagonal blocks as if the system is completely decomposable (see [16]), and to compute the stationary probability distribution of each diagonal block. However, there are two problems with this approach. First of all, since the diagonal blocks are substochastic, the off-diagonal probability mass must somehow be incorporated into the diagonal blocks. Secondly, the probabilities obtained by this approach are conditional probabilities, and this condition has to be removed by weighing each probability subvector by the probability of being in that group of states. Only if these two problems are overcome, can one form the stationary distribution vector of the Markov chain by weighing the subvectors and pasting them together. The IAD algorithm is a two-step algorithm that handles these two problems in an iterative framework.

In most real-life applications, the Markov chains we come across are sparse and they more or less possess some structure. That is, the ratio of the number of nonzero elements to the total number of elements in the underlying chain is small; moreover, the magnitude and location of these nonzero elements is not random. Depending on the size of the system under investigation, the chain to be solved can be quite large implying an implementation problem on computers even for iterative methods. Indeed, sparse storage schemes may not be adequate to handle some chains (see [20]).

One way of getting around such a problem is to try to locate any inherent structure in the chain for which there are existing memory efficient solution techniques. For instance, if the Markov chain is nearly completely decomposable, then the conditional stationary probability vector of each NCD block and consequently the stationary probability vector of the entire chain may be bounded with the bounded aggregation method discussed in [2] and [13]. On the other hand, if there are lumpable (or quasi-lumpable, [5]) sets of states in the Markov chain, one can cut down the amount of computation drastically by lumping such states and solving for a smaller Markov chain (see [7]). Note that, in both cases, one essentially solves for smaller Markov chains, and then weighs, or as in the second case, extends these individual solutions to the complete Markov chain. In order to benefit from these approaches, one first has to make sure that the Markov chain satisfies either one of the characteristics. If the given problem does not conform to these structural properties, bounded aggregation and lumping will not be of much help.

Another way of undertaking the problem is to explore only the states that will be used in the computation of the desired performances. Techniques of this type have been addressed in [4], [9], [14], and [15]. The property of sparse Markov chains that makes such a notion feasible is that each state has a few neighboring states (states within one-step transition). Secondly, in many cases, real systems have most of the probability mass concentrated on a subset of states with the result that the system very rarely reaches the other states; that is, these are systems having *highly unbalanced stationary probabilities* [11]. The idea then is to start at a particular state and explore states in the desired direction. The search direction may be the most probable states, the least probable states, or another set of states identified in advance or for which a different criterion has to be satisfied at each iteration of the exploration procedure. However, we must stress that, so far bounding techniques have been succesfully applied only to performance measures that depend on the most probable states or on the conditional stationary vector of a NCD block.

Under the light of the above discussion, our aim is to come up with a computationally efficient algorithm that locates states of interest rapidly. This is achieved by considering more than one starting state simutaneously and exploring (hopefully) fewer states from each one of these starting states. In other words, we wish to get a feeling for the overall behavior of the system by conducting a few local searches in a parallel manner. Depending on the stopping criterion chosen, the number of states explored from each starting state may vary. After states of interest are located, bounds on performance measures may be obtained either exactly through Courtois and Semal's technique discussed in [2] and [13], or approximately through the help of the ideas presented in [14], [15], and [9].

In the next section, we describe how information from multiple starting states may be utilized to locate states of interest rapidly. In the third section, we point out two different approaches that may be used to bound performance measures of interest. In the fourth section, we describe the problem experimented with and discuss the results of these experiments. Finally, we draw some conclusions in the fifth section.

2 State Space Exploration in Markov Chains

Most real-life systems modeled in the form of Markov chains yield stochastic matrices that have few outgoing transitions from each state. However, it is not correct to assume beforehand that only a few states will accumulate most of the probability mass. In fact, a system such as the multi-media traffic queue discussed in §4 may need many states to concentrate even 90% of the probability mass. Therefore, exploration of states, most probable ones for instance, starting from a single state as suggested in [4], [14], and [15], may be time consuming, sometimes even not viable.

Since tha aim of the exploration process is to locate states of interest as quickly as possible, one can start from different states and explore groups of states in the desired direction simultaneously. In this technique, it is quite probable (almost certain) that some explored states will occur in more than one set of explored states. Normally, one should explore a larger number of states from each starting state for a small number of starting states; whereas, if many starting states are considered, a sufficient number of steps from each one needs to be taken to locate those states of interest accurately. The exploration algorithm inspired by the work in [14] is given below. Note that the same algorithm is executed for each starting state.

Define:

s_f	: starting state at the current iteration
s_c	: state chosen at the current iteration
$\mathcal{E}^{(k)}$	: set of explored states at iteration k
$\mathcal{U}^{(k)}$	: set of unexplored states at iteration k
$\mathcal{T}$	: neighboring states of s_c
k	: *cardinality* of $\mathcal{E}^{(k)}$ (i.e., number of elements in set $\mathcal{E}^{(k)}$)
$u^{(k)}$	: cardinality of $\mathcal{U}^{(k)}$
$m^{(k)}$	: sum of k and $u^{(k)}$ (introduced for notational convenience)

Algorithm:

1. Choose a starting state, s_f
   $k = 0;\ \ s_c = s_f;\ \ \mathcal{E}^{(0)} = \{s_c\};\ \ \mathcal{U}^{(0)} = \emptyset$

2. $k = k + 1$
   $\mathcal{T} = \{s_j \mid \Pr(s_c \rightarrow s_j) \neq 0,\ \ s_j \neq s_c\}$
   $\mathcal{U}^{(k)} = (\mathcal{U}^{(k-1)} \bigcup \mathcal{T}) \backslash \mathcal{E}^{(k-1)}$
   $u^{(k)} = \#(\mathcal{U}^{(k)})$
   $m^{(k)} = k + u^{(k)}$

3. For convenience, map $\mathcal{E}^{(k)}$ to $\{1, 2, \ldots . k\}$ (make sure s_f is mapped to 1)
   Also map $\mathcal{U}^{(k)}$ to $\{k+1, k+2, \ldots, m^{(k)}\}$
   Form the $(m^{(k)} + 1) \times (m^{(k)} + 1)$ stochastic matrix

$$\boldsymbol{P}^{(k)} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,k} & p_{1,k+1} & \cdots & p_{1,m^{(k)}} & p_{1,m^{(k+1)}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{k,1} & \cdots & p_{k,k} & p_{k,k+1} & \cdots & p_{k,m^{(k)}} & p_{k,m^{(k+1)}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{m^{(k)},1} & \cdots & p_{m^{(k)},k} & p_{m^{(k)},k+1} & \cdots & p_{m^{(k)},m^{(k)}} & p_{m^{(k)},m^{(k)}+1} \\ 1 & \cdots & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{2.1}$$

where

$$p_{i,m^{(k)}+1} = 1 - \sum_{j=1}^{m^{(k)}} p_{i,j}, \quad 1 \leq i \leq m^{(k)} \tag{2.2}$$

4. Solve $\boldsymbol{\pi}^{(k)}\boldsymbol{P}^{(k)} = \boldsymbol{\pi}^{(k)}$ for $\boldsymbol{\pi}^{(k)}$ subject to $\|\boldsymbol{\pi}^{(k)}\|_1 = 1$
5. Stop if the stopping criterion is met
6. Choose a new state, s_c, according to some rule from $\mathcal{U}^{(k)}$
   Update the starting state, s_f, and reorder $\mathcal{E}^{(k)}$,$\mathcal{U}^{(k)}$ (optional)
   $\mathcal{E}^{(k)} = \mathcal{E}^{(k)} \bigcup \{s_c\}$
   $\mathcal{U}^{(k)} = \mathcal{U}^{(k)} \backslash \{s_c\}$
   Goto 2

Here is how the algorithm works for a single starting state. First a starting state, s_f, is chosen (see step 1 of the algorithm) either randomly by the computer or by the user who may have knowledge about the nature of the states. Initially, the starting state is the only explored state and its neighbors (states within a single transition excluding itself) form the set of unexplored states. Then we form a matrix, $\boldsymbol{P}^{(k)}$, which has entries for the one-step transition probabilities among all the explored and unexplored states. Since this matrix is not stochastic, we add one more column that sums the rows of the matrix to 1. Finally, we include one last row with all 0's except for a 1 at the column position corresponding to the starting state (see step 3 of the algorithm). Note that this matrix is different than the matrix considered in [14], firstly because, it is a stochastic matrix. Secondly all transitions among the explored and unexplored states have to be known. On the other hand, there is no need to know the transitions from the unexplored states to states other than the explored and unexplored states. Therefore, all neighbors of the unexplored states need not be explored in this algorithm. The reason behind introducing a 1 to the position of the starting state in the last row of $\boldsymbol{P}^{(k)}$ is to restart the original process from the starting state whenever the process makes a transition to one of the unknown states (states yet have to be discovered and about which nothing is known). In essence, we aggregate all unknown states to a single state and place this unknown state at the end of the matrix in each dimension. The stopping criterion needed in step 5 may either be a fixed number of states to be explored or more naturally, a threshold imposed on the mean time between two visits to unknown states (MTTU). Clearly, MTTU can be computed as the sum of the mean number of visits to explored states between two visits to the aggregated unknown state. As shown in [4], [14], and [15], an approximation to the amount of time spent in the set of explored states at iteration k, in our case, may be given by

$$\text{MTTU}^{(k)} = \frac{\sum_{i=1}^{k} \pi_i^{(k)}}{\pi_{m^{(k)}+1}^{(k)}}, \tag{2.3}$$

where the stationary vector elements are available at the end of step 4. However, rather than imposing a threshold on $\text{MTTU}^{(k)}$, we choose to use $\sum_{i=1}^{k} \pi_i^{(k)}$ as the stopping criterion. The reason for this choice is that there is a probability, namely $\sum_{i=k+1}^{m^{(k)}} \pi^{(k)}$, which is not accounted for in $\text{MTTU}^{(k)}$. Secondly, we do not know how large a value is to be chosen as a threshold for MTTU at the outset. Clearly, a threshold value greater than or equal to $\sum_{i=1}^{k} \pi_i^{(k)}$ implies a value that is greater than or equal to $(\sum_{i=1}^{k} \pi_i^{(k)})/(1-\sum_{i=1}^{k} \pi_i^{(k)})$ for $\text{MTTU}^{(k)}$. However, the converse statement

is not true. A state, s_c, is chosen from the set of unexplored states if the stopping criterion has not been met. If the most probable states are of interest, at step 6, we choose $s_c = s_j$ so that

$$\pi_j^{(k)} = \max_{k+1 \le i \le m^{(k)}} \pi_i^{(k)}. \tag{2.4}$$

Thereafter, the chosen state becomes one of the explored states. It is removed from the set of unexplored states and we continue with the next iteration. At each iteration, neighbors of the recently chosen state that do not exist in the sets of explored and unxplored states also join the set of unexplored states. Clearly, if all states of the Markov model were to be explored, we would have the exact solution vector at step 4 and $\sum_{i=1}^{k} \pi_i^{(k)} = 1$ (i.e., MTTU would be infinitely large). Hence, the algorithm always terminates with a properly chosen threshold. If least probable states are to be explored, we replace max by min.

The mapping of states to integers as described in step 3 is introduced to enhance clarity and the understanding of the algorithm. From an implementation point of view, this mapping is unnecessary. With proper indexing, there is essentially no need to reconstruct the arrays storing the sets of explored and unexplored states, and for that matter, the stochastic matrix right from the beginning at each iteration of the algorithm. One can continue from where it was left off at the last iteration and simply build on to the existing data structures. Moreover, the algorithm can also be applied to the continuous-time Markov chain if the matrix in step 3 is replaced accordingly. The fact that the transition rate matrix has negative diagonal entries is irrelevant, since transitions from a state onto itself are not considered in the search process. In this case, the second statement in step 2 should be rewritten as

$$\mathcal{T} = \{s_j \mid \text{Rate}(s_c \rightarrow s_j) \neq 0,\ s_j \neq s_c\} \tag{2.5}$$

One improvement we find useul is to sort the solution vector obtained in step 4 of the algorithm and to reform the sets of explored and unexplored states accordingly. If most probable states are sought, then set $s_f = s_l$ so that

$$\pi_l^{(k)} = \max_{1 \le i \le m^{(k)}} \pi_i^{(k)} \tag{2.6}$$

and

$$\begin{aligned} \mathcal{E}^{(k)} &= \{\text{the } k \text{ states with the highest probabilities in } \boldsymbol{\pi}^{(k)}\ \} \\ \mathcal{U}^{(k)} &= \{\text{the neighbors of the states in } \mathcal{E}^{(k)}\ \} \end{aligned}$$

at step 6 of the algorithm. If least probable states are to be explored, choose the lowest probabilities when forming $\mathcal{E}^{(k)}$. Note that this is a costly operation, and it basically requires the matrix at step 3 to be reorganized. The reason for this reorganization is that an unexplored state may have become an explored state, and we need to include its neighbors in the set of unexplored states as well. Similarly, an explored state may become an unexplored state, and all or some of its neighbors may need to be removed from the set of unexplored states. Even the state itself may need to be removed if it is not a neighbor of any explored state. However, this operation need not be performed at each iteration, but just frequently enough so that misjudgements made earlier may be corrected. Otherwise, once a state becomes an explored state, it will remain as one until the algorithm terminates. An alternative is to change only the starting state to the state with the highest (lowest) probability in the set of explored states. In this case, a reorganization as described above will not be necessary.

If we neglect this last improvement that requires reorganization of the explored and unexplored states, then, with a careful implementation, each iteration may buid on to the preceeding one. Since

we do not expect to carry out many iterations, we suggest Gaussian elimination as the solution method applied to the system

$$(\boldsymbol{I} - \boldsymbol{P}^{(k)})^T(\boldsymbol{\pi}^{(k)})^T = 0 \tag{2.7}$$

subject to the same constraint in step 4. Back substitution is the only solution phase required since the coefficient matrix is singular and the right-hand side is 0. Even forward elimination may resume from where it left off at the least iteration. The justification for this choice of the solution method is that, we plan to consider only sufficiently many starting states so that each matrix generated from a starting state is not larger than the size for which the computer produces solutions quickly. Additionally, a bound on the maximum number of operations to be performed may be estimated beforehand as opposed to the case with matrix iterative methods. One important observation regarding the solution procedure is that the last column in $\boldsymbol{P}^{(k)}$, which sums the rows to 1, has no significance. Once the transposed system is formed, this last column becomes the last row of the transposed system, and it will be all 0's after the system is reduced to upper-triangular form (since we have a singular system of linear equations).

Just as in [14], let us assume that on the average each state has r output transitions (excluding itself) and a fraction p of those end in unexplored states. With the given assumption, $\mathcal{U}^{(k-1)} = pr(k-1)$ and $\mathcal{U}^{(k)} = prk$. Therefore, the number of extra rows in $\boldsymbol{P}^{(k)}$ is $pr+1$. An upperbound on the number of multiplications required to carry out the reductions in between columns $((k-1)+pr(k-1)+1)$ and $(k+prk+1)$ for rows 1 through $((k-1)+pr(k-1))$ and to reduce the extra $pr+1$ rows to upper-triangular form is given by

$$\sum_{j=1}^{(k-1)+pr(k-1)} \sum_{i=1}^{j-1}(pr+1) + \sum_{j=(k-1)+pr(k-1)+1}^{k+prk+1} \sum_{i=1}^{j-1}(k+prk+1-i+1) = O(k^2(pr+1)^3).$$

For the single back substitution phase, we have $O((k+prk+1)^2) = O(k^2(pr+1)^2)$ multiplications. Hence, at iteration k the total number of multiplications performed to solve the system in step 4 is $O(k^2(pr+1)^3)$. The total number of operations over N iterations of the algorithm is $O(N^2(pr+1)^3)$. Since we carry out the solution procedure in temporary storage and choose not to overwrite $\boldsymbol{P}^{(k)}$, the storage requirements for the algorithm over N iterations is $(N+prN+1)^2 + (N+prN)r$. The total storage needed is then $O(N^2(pr+1)^2)$.

Let us return to the idea of starting the exploration process from multiple states. There are two alternatives that may be followed. The first approach is to start from a few states (we choose a single starting state) and to explore for a considerable amount of time (i.e., a threshold probability of 0.99999 in the set of explored states, for instance) from each one. The other approach that may be taken is to start from a larger number of states and to explore fewer transitions out of each starting state (i.e., a threshold probability of 0.99 in the set of explored states, for instance). Remember that, our ultimate goal is to explore sufficiently many states so that we can obtain good bounds on performance measures dependent upon these states. In the next section, we discuss how multiple sets of explored states may be combined to achieve bounds on performance measures.

3 Bounding Performance Measures

We are interested in computing bounds for certain performance measures in systems modeled as Markov chains. For queueing systems, these measures may be the average number of customers, the mean waiting time, or the blocking probability for a specific queue. In communication systems, they may be the total packet loss rate, the probability of an empty system, or any other relevant

measure. In all cases, the measures can be computed exactly if the stationary vector of the Markov chain is available. When the stationary vector is not computable due to the size of the system, or an approximation is sought in a short time, one may use the dynamic state exploration technique.

Let us briefly mention a result for NCD Markov chains. It is well-known that (see [2], [13]) the conditional stationary distribution of a NCD block may be tightly bounded. Furthermore, the marginal distribution, which removes this conditioning, may also be bounded by an aggregation step following the computation of the conditional stationary vector of each NCD block. Consequently, it is possible to compute the best known bounds for the stationary distribution of a NCD block, and if the states of interest are the states of a NCD block, this technique gives the best achievable bounds for the stationary probabilities of the corresponding block of states. Interestingly, the same technique gives good bounds for the conditional stationary of the most probable states, when the states explored as such through dynamic state exploration concentrate more than, say, 99% of the probability mass. As the percentage increases, the bounds get tighter.

Another technique is to replicate the set of explored states other than the starting state and to classify these replicated states according to the minimum number of steps taken from the starting state to reach each one. After the classification of replicated states into disjoint subsets is performed, each subset is aggregated (the way in which aggregation is done differs depending on whether a lower or an upper bound for the conditional stationary vector is sought) and an equivalent system is derived. By this technique, looser but yet computationally efficient lower and upper bounds on the stationary distribution of most probable states may be obtained. See [9], [14], and [15] for details.

We remark that either of the two techniques may be employed when the sought performance measure is dependent on the most probable states. So far, we have not come up with a scheme which would help bound measures depending on the least probable states (such as the blocking probability in a finite buffer) tightly. After careful consideration of the above two techniques, we think the first scheme may be implemented in such a way that, it has, if not less, then at least comparable computational efficiency to the second technique for a reasonable number of explored states (i.e., in the order of hundreds). The justification for this choice is the following. Since r, the average number of outgoing transitions from each state, is small in the systems considered, the second technique suggested in [9, 14, 15] requires the solution of a homogeneous linear system of order at least $\frac{1}{l+1}N$, where l satisfies $\sum_{i=0}^{l}[r(1-p)]^i = N$ — this is a best case analysis. Besides, there are extra operations involved in classifying the states into subsets and aggregating the subsets, which we have not accounted for. Moreover, the bounds computed by using the first technique will be definitely tighter than the bounds obtained through the approximation in the second technique since both schemes consider the same set of explored states.

Now we are in a position to show how bounds on the conditional stationary distribution of a set of states $\mathcal{S}' \subset \mathcal{S}$, where $\#(\mathcal{S}') = n' > 0$, may be computed (see [2], page 810).

1. As always, let $\boldsymbol{P}$ be the stochastic transition probability matrix of the Markov chain defined on the state space $\mathcal{S}$. From $\boldsymbol{P}$, extract the one-step transition probability matrix $\boldsymbol{P'}$ (which is not stochastic) confined to the states in $\mathcal{S}'$.

2. Augment $\boldsymbol{P'}$ with one column and one row to give the $(n'+1) \times (n'+1)$ stochastic matrix

$$\boldsymbol{W'} = \begin{pmatrix} \boldsymbol{P'} & \boldsymbol{e} - \boldsymbol{P'e} \\ \boldsymbol{x}^T & 0 \end{pmatrix}, \tag{3.1}$$

   where $\|\boldsymbol{x}\|_1 = 1$ and $\boldsymbol{e}$ is a column vector of 1's whose length is determined in the context in which it is used.

3. For $i = 1, 2, \ldots, n'$:

   (a) Substitute 1 to column position i in the last row of $\boldsymbol{W}'$; that is, $x_i = 1$.

   (b) Solve $\boldsymbol{z}_i \boldsymbol{W}' = \boldsymbol{z}_i$, $\|\boldsymbol{z}_i\|_1 = 1$.

   (c) Normalize the first n' components of $\boldsymbol{z}_i$ such that $\sum_{j=1}^{n'} z_{i,j} = 1$.

   (d) Now let $\boldsymbol{z}_i$ be the normalized row vector of length n'.

4. Form the stochastic matrix

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_{n'} \end{pmatrix}. \tag{3.2}$$

5. If $\boldsymbol{v}$ is the conditional stationary distribution vector of $\mathcal{S}'$, then

$$v_j^{inf} = \max\{\min_i(z_{i,j});\ 1 - \sum_{k \neq j} \max_i(z_{i,k})\}, \tag{3.3}$$

$$v_j^{sup} = \min\{\max_i(z_{i,j});\ 1 - \sum_{k \neq j} \min_i(z_{i,k})\}, \tag{3.4}$$

   where $j = 1, 2, \ldots, n'$.

We would like to emphasize that the lower and upper bounds obtained by the above technique are the tightest bounds computable when only partial information (i.e., $\boldsymbol{P}'$) is available and nothing is known about transitions from states in $\mathcal{S} \backslash \mathcal{S}'$ to states in $\mathcal{S}'$ (i.e., $\boldsymbol{x}$).

For our purposes, bounds on the conditional stationary vector of the set of explored states may be obtained by extracting the one-step transition probability matrix of the set of explored states from $\boldsymbol{P}^{(k)}$ (step 1). If multiple starting states are considered, then the final set of explored states is the union of the sets of explored states that come out of different starting states. In this case, the one-step transition probability matrix which needs to be formed from scratch has as many states as there are in the union. Since the one-step transition probability matrix is substochastic in both cases (given that all states of the Markov chain have not been explored), we augment it with one more column and one more row (step 2). The extra column sums the rows to 1. In order to compute the desired lower and upper bounds, we have to solve as many systems of linear equations as the number of states in the final set of explored states (step 3–4). Each system is formed by placing a 1 to a different column position in the extra row except for the extra state introduced. The equalities (3.3) and (3.4) are direct consequences of the restricted convex combination and of the normalization of the solution vector (step 5). Note that, a single $\boldsymbol{LU}$ decomposition, and as many backsubstitutions as there are states in the final set of explored states, needs to be performed. In the next section, we provide results of some experiments conducted on a large and sparse Markov chain.

4 Numerical Experiments

Before we specify the system parameters chosen, we will pause and discuss the applicability of dynamic state exploration in real-life problems. We especially would like to stress that the following is *not* an exercise in computing performance measures for a given problem by the best means available; but, sometimes due to the size of the problem under consideration, dynamic state exploration

may be the only viable technique for obtaining a solution. For the ATM traffic queue, dynamic state exploration gives satisfying results only for low to very low loads, because it is those cases in which the probability mass is accumulated in a reasonably small number of states. Therefore, the technique is *not* recommended for solving such a problem in the most general case. Our aim is to direct the attention to the feasibility of dynamic state exploration as a tool that may be useful when a specific criteria is met.

The application selected is from the communications field. A multi-media traffic queue on an ATM link is modelled in the form of a multi-dimensional Markov queueing process, where each individual arrival stream is treated as an independent Markov-modulated process. For details, see [20]. In the same reference it is indicated that for such a system the overall arrival process becomes a Markov-modulated Poisson process and the queueing model may be described in the form of a continuous Markov chain with states

$$\{(q(t), x(t)) \mid 0 \leq q(t) \leq K-1;\ x(t) \in X(t)\}. \tag{4.1}$$

Here $q(t)$ represents the queue length at time t ($K-1$ is the maximum queue size including the one in the server) and $x(t)$ is the arrival phase under the given buffer occupancy (it is in fact the state of the superimposed Markov-modulated traffic stream at time t). Hence, we have a Quasi-Birth-Death (QBD) process where each level corresponds to an arrival phase. Without overload control, $q(t)$ and $x(t)$ are independent. Therefore, the phase transitions at each level are identical except for the boundary levels, 0 and $K-1$. The transition rate matrix is then given by

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{U} & & & \\ \boldsymbol{D} & \boldsymbol{A} & \boldsymbol{U} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{D} & \boldsymbol{A} & \boldsymbol{U} \\ & & & \boldsymbol{D} & \boldsymbol{A}_{K-1} \end{pmatrix} \tag{4.2}$$

$\boldsymbol{Q}$ is block tridiagonal. The matrices $\boldsymbol{A}_0$, $\boldsymbol{A}$, and $\boldsymbol{A}_{K-1}$ specify changes in the arrival phase whereas $\boldsymbol{U}$ and $\boldsymbol{D}$, which are diagonal matrices, describe changes in buffer occupancy. The number of states at each level is given by $N = \prod_{i=1}^{M} N_i$, where M is the number of independent traffic streams and N_i is the number of states in the Markov chain that corresponds to the i^{th} traffic stream. Consequently, $\boldsymbol{Q}$ is of size $NK \times NK$.

For the statistical multiplexer that concentrates traffic from voice and video sources (i.e., $M = 2$), $K = 512$, $N_1 = 30$, and $N_2 = 20$ ([20]) giving rise to a $307,200 \times 307,200$. We consider a smaller, yet instructive example for which K, N_1, N_2 have the values 120, 30, 20 respectively. There are two reasons behind this choice. Firstly, the steady state distribution of both systems is the same for the loads considered. Secondly, in order not to overwrite $\boldsymbol{P}^{(k)}$ with the $\boldsymbol{LU}$ decomposition at the k^{th} iteration of the exploration algorithm, we need to use a second (temporary) matrix of the same size as $\boldsymbol{P}^{(k)}$. Since both matrices are two-dimensional and they are not stored in compact form, the system we chose is the largest system permissible on our SPARC 2 station under the described conditions. The linear system to be solved is $72,000 \times 72,000$ in this case, and the percentage of nonzero elements is $490,681/72,000^2 = 0.94652 \times 10^{-4}$. The number of outgoing transitions from a state (excluding itself) is approximately 6 (i.e., $r \approx 6$). For the system under consideration, the probability mass is distributed across many states when the load is larger than 10%, and a single search from a given starting state does not provide much information unless the search extends over a considerable number of states. However, when the load is less than 0.1, most of the probability mass is concentrated in a smaller number of states. We also noticed that together with the load on the system, the number of states that must be explored for a given

probability mass changes for different values of K, N_1, and N_2. We experimented with four different problems. These problems are derived from the ATM model using loads of 0.1, 0.01, 0.001, and 0.0001 respectively. It was quite interesting to find out that the stochastic transition probability matrix in each case was nearly completely decomposable. In Tables 1–4, the decomposability parameter, γ, is on the order of the degree of coupling that corresponds to the one-step transition probability matrix for the given size and number of aggregates, ngr.

Table 1: Sizes and number of aggregates for a load of 0.1:
$72,000 \times 72,000$ ATM model

γ	ngr	*size of aggregates*
10^{-5}	4	(120 – 66,120)
10^{-4}	539	(1 – 1,200)
10^{-3}	838	(1 – 120)
10^{-2}	2,623	(1 – 120)

Table 2: Sizes and number of aggregates for a load of 0.01:
$72,000 \times 72,000$ ATM model

γ	ngr	*size of aggregates*
10^{-6}	4	(120 – 66,120)
10^{-5}	442	(1 – 3,360)
10^{-4}	838	(1 – 120)
10^{-3}	2,266	(1 – 120)
10^{-2}	26,780	(1 – 120)

Table 3: Sizes and number of aggregates for a load of 0.001:
$72,000 \times 72,000$ ATM model

γ	ngr	*size of aggregates*
10^{-7}	2	(3,600 – 68,400)
10^{-6}	442	(1 – 3,600)
10^{-5}	838	(1 – 120)
10^{-4}	2,266	(1 – 120)
10^{-3}	25,947	(1 – 120)
10^{-2}	72,000	1

Table 4: Sizes and number of aggregates for a load of 0.0001:
$72,000 \times 72,000$ ATM model

γ	*ngr*	*size of aggregates*
10^{-8}	2	(3,600 – 68,400)
10^{-7}	423	(1 – 3,600)
10^{-6}	838	(1 – 120)
10^{-5}	2,266	(1 – 120)
10^{-4}	25,828	(1 – 120)
10^{-3}	72,000	1
10^{-2}	72,000	1

We filled out Table 5 by computing the stationary probability vector of each stochastic transition probability matrix using the iterative aggregation-disaggregation (IAD) algorithm with Gaussian elimination for the solution of the aggregates and the coupling matrix (see [3], [18]). The marginal probabilities are given by

$$\Pi(i) = \sum_{s_j \in \text{block } i} \pi_j \qquad \text{for } i = 0, 1, \ldots, K-1, \tag{4.3}$$

where $\boldsymbol{\pi}$ is the stationary vector of the corresponding matrix.

Table 5: Marginal probability distribution:
$72,000 \times 72,000$ ATM model, loads 0.1, 0.01, 0.001, 0.0001

$\#ofcells$	0.1	0.01	0.001	0.0001
0	0.90485×10^{0}	0.99048×10^{0}	0.99905×10^{0}	0.99990×10^{0}
1	0.85317×10^{-1}	0.94169×10^{-2}	0.95054×10^{-3}	0.95142×10^{-4}
2	0.87473×10^{-2}	0.97265×10^{-4}	0.98245×10^{-6}	0.98342×10^{-8}
3	0.96049×10^{-3}	0.10753×10^{-5}	0.10868×10^{-8}	0.10879×10^{-11}
4	0.11180×10^{-3}	0.12595×10^{-7}	0.12737×10^{-11}	0.2×10^{-15}
5	0.13693×10^{-4}	0.15518×10^{-9}	0.16×10^{-14}	
6	0.17545×10^{-5}	0.19994×10^{-11}		
7	0.23410×10^{-6}	0.268×10^{-13}		
8	0.32405×10^{-7}	0.4×10^{-15}		
9	0.46387×10^{-8}			
10	0.68488×10^{-9}			
11	0.10405×10^{-9}			
12	0.16234×10^{-10}			
13	0.25965×10^{-11}			
14	0.4251×10^{-12}			
15	0.712×10^{-13}			
16	0.122×10^{-13}			
17	0.22×10^{-14}			
18	0.4×10^{-15}			
19	0.1×10^{-15}			

As it can be seen in Table 6, the minimum number of states required to accumulate a given probability mass decreases with decreasing load for the system.

Table 6: Minimum number of states required to accumulate a probability mass: $72,000 \times 72,000$ ATM model, loads 0.1, 0.01, 0.001, 0.0001

$Prob \backslash Load$	0.1	0.01	0.001	0.0001
0.9	122	82	79	79
0.95	176	107	102	101
0.99	331	195	155	152
0.999	615	342	262	224
0.9999	964	524	395	319
0.99999	1,380	720	521	445

The average number of cells in the buffer at steady state is given by

$$\bar{w} = \sum_{i=0}^{K-1} i\Pi(i). \tag{4.4}$$

Table 7: Average number of cells in the buffer at steady state: $72,000 \times 72,000$ ATM model, loads 0.1, 0.01, 0.001, 0.0001

$Load$	$avg.\#ofcells$
0.1	0.10622×10^{0}
0.01	0.96147×10^{-2}
0.001	0.95251×10^{-3}
0.0001	0.95162×10^{-4}

Note that the marginal probability distribution of having 0 cells in the buffer, which is a large portion of the probability mass for the chosen loads as one would expect, does not contribute to the average number of cells in the buffer. Hence, the values in Table 7 are due to marginal probabilities of having 1 or more cells in the buffer. In all problems, the probability of having 20 or more cells in the buffer turned out be 0.

We carried out two types of experiments. In the first group of experiments, we considered a single starting state. Since it is not quite clear which state is the most probable one, we chose state 1 as the starting state. This state lies in the first block, where there are no cells in the buffer. In the second group of experiments we considered 10 starting states. We chose states 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400; they are uniformly distributed over the first four blocks in the matrix. We chose not to sort the solution vector obtained in step 4 of the exploration algorithm in §2. We present the results in five decimal digits precision. Let us represent by Π_e and $\bar{w}_e$, the marginal probability and the average number of cells in the buffer at steady state conditioned on the system being in the set of explored states.

1. Load = 0.1
   Single starting state : 1
   Threshold probability mass in explored states : 0.999
   $k = 126, U^{(k)} = 55$

$$
\begin{array}{lcccl}
0.44356 \times 10^{-6} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.10739 \times 10^{-6} & \leq & \Pi_e(1) & \leq & 0.10000 \times 10^{1} \\
0.24862 \times 10^{-7} & \leq & \Pi_e(2) & \leq & 0.87076 \times 10^{0} \\
0.47159 \times 10^{-8} & \leq & \Pi_e(3) & \leq & 0.16769 \times 10^{0} \\
0.71047 \times 10^{-9} & \leq & \Pi_e(4) & \leq & 0.29289 \times 10^{-1} \\
0.10185 \times 10^{-11} & \leq & \Pi_e(5) & \leq & 0.53274 \times 10^{-2} \\
0.00000 \times 10^{0} & \leq & \Pi_e(6) & \leq & 0.45214 \times 10^{-3} \\
0.17411 \times 10^{-6} & \leq & \bar{w}_e & \leq & 0.33911 \times 10^{1}
\end{array}
$$

   Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400
   Threshold probability mass imposed on each starting state : 0.999
   $k = 303$

$$
\begin{array}{lcccl}
0.19859 \times 10^{-6} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.79026 \times 10^{-8} & \leq & \Pi_e(1) & \leq & 0.10000 \times 10^{1} \\
0.21722 \times 10^{-9} & \leq & \Pi_e(2) & \leq & 0.29416 \times 10^{-1} \\
0.80788 \times 10^{-15} & \leq & \Pi_e(3) & \leq & 0.42917 \times 10^{-3} \\
 & & & & \\
0.83370 \times 10^{-8} & \leq & \bar{w}_e & \leq & 0.10601 \times 10^{1}
\end{array}
$$

2. Load = 0.01
   Single starting state : 1
   Threshold probability mass in explored states : 0.9999
   $k = 550, U^{(k)} = 299$

$$
\begin{array}{lcccl}
0.86507 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.11184 \times 10^{-6} & \leq & \Pi_e(1) & \leq & 0.13493 \times 10^{0} \\
0.11655 \times 10^{-8} & \leq & \Pi_e(2) & \leq & 0.85478 \times 10^{-2} \\
 & & & & \\
0.11417 \times 10^{-6} & \leq & \bar{w}_e & \leq & 0.15203 \times 10^{0}
\end{array}
$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400
Threshold probability mass imposed on each starting state : 0.9995
$k = 267$

$$
\begin{array}{lcccl}
0.80007 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.21875 \times 10^{-6} & \leq & \Pi_e(1) & \leq & 0.19983 \times 10^{0} \\
0.14153 \times 10^{-13} & \leq & \Pi_e(2) & \leq & 0.63946 \times 10^{-4} \\
0.12835 \times 10^{-15} & \leq & \Pi_e(3) & \leq & 0.34401 \times 10^{-4}
\end{array}
$$

$$
0.21875 \times 10^{-6} \leq \bar{w}_e \leq 0.20006 \times 10^{0}
$$

3. Load = 0.001
Single starting state : 1
Threshold probability mass in explored states : 0.9999
$k = 341, U^{(k)} = 244$

$$
\begin{array}{lcccl}
0.90345 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.22791 \times 10^{-9} & \leq & \Pi_e(1) & \leq & 0.96552 \times 10^{-1}
\end{array}
$$

$$
0.22791 \times 10^{-9} \leq \bar{w}_e \leq 0.96552 \times 10^{-1}
$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400
Threshold probability mass imposed on each starting state : 0.9995
$k = 170$

$$
\begin{array}{lcccl}
0.99567 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.00000 \times 10^{0} & \leq & \Pi_e(1) & \leq & 0.42981 \times 10^{-2} \\
0.00000 \times 10^{0} & \leq & \Pi_e(2) & \leq & 0.23211 \times 10^{-4} \\
0.00000 \times 10^{0} & \leq & \Pi_e(3) & \leq & 0.13660 \times 10^{-4}
\end{array}
$$

$$
0.00000 \times 10^{0} \leq \bar{w}_e \leq 0.43855 \times 10^{-2}
$$

4. Load = 0.0001
Single starting state : 1
Threshold probability mass in explored states : 0.9999
$k = 294, U^{(k)} = 283$

$$
\begin{array}{lcccl}
0.99665 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.00000 \times 10^{0} & \leq & \Pi_e(1) & \leq & 0.33497 \times 10^{-2}
\end{array}
$$

$$
0.00000 \times 10^{0} \leq \bar{w}_e \leq 0.33497 \times 10^{-2}
$$

Multiple starting states : 240, 480, 720, 960, 1200, 1440, 1680, 1920, 2160, 2400
Threshold probability mass imposed on each starting state : 0.9997
$k = 155$

$$
\begin{array}{lcll}
0.99995 \times 10^{0} & \leq & \Pi_e(0) & \leq & 0.10000 \times 10^{1} \\
0.00000 \times 10^{0} & \leq & \Pi_e(1) & \leq & 0.46305 \times 10^{-4} \\
0.00000 \times 10^{0} & \leq & \Pi_e(2) & \leq & 0.23167 \times 10^{-5} \\
0.00000 \times 10^{0} & \leq & \Pi_e(3) & \leq & 0.13672 \times 10^{-5}
\end{array}
$$

$$
0.00000 \times 10^{0} \leq \bar{w}_e \leq 0.55040 \times 10^{-4}
$$

The probability mass accumulated in the set of explored states (conditioned on the event that the process restarts from the starting state whenever it reaches the unknown states) increases asymptotically at each iteration. That is, the value of the probability mass in the explored states for the current iteration may be less than the mass in an earlier iteration; however, if sufficiently many iterations are performed, the probability mass in the set of explored states will exceed the prespecified threshold. Although the probability mass is conditional, the probability of being in the set of explored states increases with the number of states in the set. If all states in the Markov chain were explored, then the probability mass would be unconditional. In order to find out a lower bound on the exact percentage of the probability mass accumulated in the set of explored states when the search terminates, one should consider each state in the set of explored states as a starting state and take the minimum among the probabilities computed. Hence, there are two reasons behind the termination of the search process with a smaller number of explored states than the minimum number required to accumulate the unconditional probability mass. The first one is the fact that the probability mass accumulated in the set of explored states is conditional, and the second one is the fact that even this conditional probability mass may not be optimal due to the starting state chosen. For instance, even though a minimum of 615 states are required to accumulate a probability mass of 0.999 for the ATM traffic queue with load 0.1 (see Table 6), only 126 states were explored when the search started from state 1. In other words, the unconditional probability mass accumulated in these 126 explored states is much less than 0.999. So the key point in the algorithm is to accumulate as much probability mass as possible in the states to be explored.

When multiple starting states are considered, it is possible to enforce the exploration of states in other parts of the Markov chain by choosing at least some of the starting states in different parts of the chain. In this way, the likelihood of skipping a state that should be included in the set of explored states is reduced. Our experience with the ATM model suggests that the number of multiple starting states should be chosen on the order of 10's. For the given model parameters, this corresponds to less than 1% of the total number of states. Since parallel implementation of the algorithm is the ultimate goal, the number of processors on the target computer is certainly a determining factor. Nevertheless, this is an issue that needs to be investigated further. Additionally, the number of states in the union of the sets of explored states is at least as many as the number of states in the smallest of these sets. Hence, multiple starting states most likely give better bounds than a single starting state for the same threshold probability mass (see the first test case).

For smaller load values, we employed a larger threshold probability mass to be accumulated in the set of explored states. In each problem, a smaller threshold probability mass for multiple starting states gave almost as good bounds as a larger threshold did for a single starting state. We observed that the dynamic state exploration technique is most effective when a large portion of the

probability mass is concentrated in a small number of states as in the ATM model with the lightest load.

5 Conclusion

In this paper, we have shown how dynamic state exploration may be used to bound certain performance measures in systems with highly unbalanced stationary probabilities. The performance measures that may be bounded in this way are those that depend on the most probable states. However, the idea may also be used to locate states with smaller probabilities, and therefore help one acquire an understanding of the nature of the states in the chain.

The technique amounts to exploring the most probable states from a starting state until a stopping criterion is met. Once the total probability of being in the explored states exceeds the preset threshold, the search ends. An improvement over the original algorithm is to sort the solution vector obtained at a given step of the exploration process, and reform the set of explored states before starting with the next step. However, this is a costly operation and should not be performed frequently. We also stress that the exploration process can be further speeded up by starting the search from multiple states and exploring for a smaller threshold probability mass from each one.

We have reported results on an ATM traffic queue having 72,000 states. The results indicate that a direct solution method such as Gaussian elimination may be employed effectively at each step of the exploration process for a small number of explored states. However, if the probability mass to be accumulated, extends over, say, more than 500 states, direct methods tend to be slow, and one should use matrix iterative methods in that case. We think dynamic state exploration is most effective when the minimum number of states accumulating a large portion of the probability mass, 0.99999 for instance, is in the order of 100–200's. Since the ATM model considered is NCD, future work may focus on quasi-lumpability as an alternate technique for obtaining bounds on performance measures of sparse, large NCD Markov chains.

References

[1] W. L. Cao and W. J. Stewart, Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains, *J. ACM* **32**, (1985), 702–719.

[2] P.-J. Courtois and P. Semal, Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition, *J. ACM* **31**, (1984), 804–825.

[3] T. Dayar and W. J. Stewart, On the effects of using the GTH method in iterative aggregation-disaggregation, to appear in *SIAM J. Sci. Comput.* **17**, (1996).

[4] D. D. Dimitrijevic and M.-S. Chen, Dynamic state exploration in quantitative protocal analysis, *Protocol Specification, Testing and Verification, IX '90*, (1990), North-Holland, 327–338.

[5] G. Franceschinis and R. R. Muntz, Bounds for quasi-lumpable Markov chains, *Performance 93*, 1–29.

[6] W. J. Harrod and R. J. Plemmons, Comparison of some direct methods for computing the stationary distributions of Markov chains, *SIAM J. Sci. Statist. Comput.* **5**, (1984), 453–469.

[7] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, D. Van Nostrand, New York (1960).

[8] J. R. Koury, D. F. McAllister and W. J. Stewart, Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains, *SIAM J. Alg. Disc. Meth.* **5**, (1984), 164–186.

[9] J. C. S. Lui and R. R. Muntz, Algorithmic approach to bounding the mean response time of a minimum expected delay routing system, *Performance Evaluation Review* **20**, (1992), 140–151.

[10] C. D. Meyer, Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems, *SIAM Rev.* **31**, (1989), 240–272.

[11] C. D. Meyer, personal communication.

[12] B. Philippe, Y. Saad and W. J. Stewart, Numerical methods in Markov chain modelling, *Oper. Res.* **40**, (1992), 1156–1179.

[13] P. Semal, Analysis of Large Markov Models, Bounding Techniques and Applications, Doctoral thesis, Université Catholique de Louvain, Louvain, Belgium (1992).

[14] E. de Souza e Silva and P. Mejía, State space exploration in Markov models, *Performance Evaluation Review* **20**, (1992), 152–166.

[15] E. de Souza e Silva and H. R. Gail, Performability analysis of computer systems: from model specification to solution, *Performance Evaluation* **14**, (1992), 157–196.

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

[17] G. W. Stewart, W. J. Stewart and D. F. McAllister, A two-stage iteration for solving nearly completely decomposable Markov chains, in *The IMA Volumes in Mathematics and its Applications* **60***: Recent Advances in Iterative Methods*, ed. by G. H. Golub, A. Greenbaum and M. Luskin, Springer-Verlag, New York (1994), 201–216.

[18] W. J. Stewart and W. Wu, Numerical experiments with iteration and aggregation for Markov chains, *ORSA J. Comput.* **4**, (1992), 336–350.

[19] Y. Takahashi, A lumping method for the numerical calculation of stationary distributions of Markov chains, Research Report B-18, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan (1975).

[20] J. Ye and S.-Q. Li, Analysis of multi-media traffic queues with finite buffer and overload control – Part I: Algorithm, *Proc. IEEE INFOCOM '91*, (1991), 1464–1474.