Finding Steady State of Safety Systems Using the Monte Carlo Method

Ray Gallagher
rayg@csc.liv.ac.uk

Department of Computer Science, University of Liverpool
Chadwick Building, Peach Street,
Liverpool, UK

Abstract *In this paper we consider finding the steady state of a safety system. We present parallel Monte Carlo Algorithms for solving eigenvalue problems which we employ to find the steady state.*

The algorithms run on a cluster of workstations under MPICH. Examples would be drawn from safety systems such as controllers etc.

Keywords: Monte Carlo Algorithms, Markov Chain, Parallel Algorithms

1 Introduction

Markov modelling techniques have been used extensively in reliability engineering, particularly with regard to repairable systems that are typical in an industrial environment. These systems offer many advantages in terms of system availability and safety. We will show that the problem of reliability of the system considered can be reduced to finding the dominant eigenvalue(s). Thus, generally, the problem transposes to a linear algebra problem.

Let us now consider Monte Carlo methods for Linear Algebra. Monte Carlo methods give statistical estimates for the functional of the solution by performing random sampling of a certain random variable whose mathematical expectation is the desired functional. They can be implemented on parallel machines efficiently due to their inherent parallelism and loose data dependencies. Using powerful parallel computers it is possible to apply Monte Carlo methods for evaluating large-scale irregular problems, which are sometimes difficult to solve by well-known numerical methods.

Let J be any functional that we estimate by the Monte Carlo method; θ_N be the estimator, where N is the number of trials. The probable error for the usual Monte Carlo method [7] is defined as parameter r_N for which $Pr\{|J - \theta_N| \geq r_N\} = 1/2 = Pr\{|J - \theta_N| \leq r_N\}$. If the standard deviation is bounded, i.e. $D(\theta_N) < \infty$, the normal convergence in the central limit theorem holds, so we have

$$r_N \approx 0.6745 D(\theta_N) N^{-1/2}. \quad (1)$$

In this paper we present Monte Carlo algorithms for evaluating the dominant eigenvalue of large real sparse matrices and their parallel implementation on a cluster of workstations using MPICH and how they can be used to find the steady state of systems. These algorithms use the idea of the Power method combined with Monte Carlo iterations by the given matrix and the inverse matrix correspondingly. In [8], [7], [5], [6] one can find Monte Carlo methods for evaluation of the dominant (maximal by modulus) eigenvalue of an integral operator. In [9], [10] Monte Carlo algorithms for evaluating the smallest eigenvalue of real symmetric matrices are proposed. Here we generalize the problem. Finding dominant eigenvalues by parallel Monte Carlo methods can be found in [10],[11]

2 Background

2.1 Algebraic Approach to Calculating Steady State

Assume we are given a transition matrix P . (In most cases of safety systems such matrices are sparse.) Applying an algebraic approach to obtain the steady state transition matrix we have $P^{n+1} = PP^n = P^n P$. Let $K = \lim P^n$ as n tends to infinity, assuming the limit exists. Then K is the solution of $K = PK$ and $K = KP$.

Let us consider evaluating P^n . Consider a Markov Chain with a finite number of states k , where the transition matrix P is a $k \times k$ array (as above). We assume that the eigenvalues of P are λ_i , $i = 1, 2, \dots, n$ and $Px = \lambda x$ has non-zero solutions x if, and only if, λ is an eigenvalue of P . Hence, we can find the corresponding right-hand eigenvectors x_i of P . The matrix P can be decomposed into a similar system of matrices comprising its eigenvalues and eigenvectors $P = B\Lambda B^{-1}$. Where Λ is a diagonal matrix of its eigenvalues $\lambda_1, \lambda_2 \dots \lambda_n$, and where the matrix B is formed from the corresponding eigenvectors. It follows immediately that $P^n = (B\Lambda B^{-1})^n = B\Lambda^n B^{-1}$.

Therefore, we can calculate the steady state of the system. We will use the Monte Carlo method to calculate the eigenvalues and the method is described below.

2.2 Power Method.

Suppose $A \in R^{n \times n}$ is diagonalizable, $X^{-1}AX = \text{diag}(\lambda_1, \dots, \lambda_n)$, $X = (x_1, \dots, x_n)$, and $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$. Given $f^{(0)} \in C^n$, the *power method* ([2]) produces a sequence of vectors $f^{(k)}$ as follows:

$$\begin{aligned} z^{(k)} &= Af^{(k-1)}, \\ f^{(k)} &= z^{(k)} / \|z^{(k)}\|_2, \\ \lambda^{(k)} &= [f^{(k)}]^H Af^{(k)}, \quad k = 1, 2, \dots \end{aligned}$$

Except for special starting points, the iterations converge to an eigenvector corresponding to the eigenvalue of A with largest magnitude (*dominant eigenvalue*) with rate of convergence:

$$|\lambda_1 - \lambda^{(k)}| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right). \tag{2}$$

Consider the case when we want to compute the smallest eigenvalue. To handle this case and others, the power method is altered in the following way: The iteration matrix A is replaced by B , where A and B have the same eigenvectors, but different eigenvalues. Letting σ denote a scalar, then the three common choices for B are: $B = A - \sigma I$ which is called the *shifted power method*, $B = A^{-1}$ which is called the *inverse power method*, and $B = (A - \sigma I)^{-1}$ which is called the *inverse shifted power method*.

Table 1. Relationship between eigenvalues of A and B

B	Eigenvalue of B	Eigenvalue of A
A^{-1}	$\frac{1}{\lambda_A}$	$\frac{1}{\lambda_B}$
$A - \sigma I$	$\lambda_A - \sigma$	$\lambda_B + \sigma$
$(A - \sigma I)^{-1}$	$\frac{1}{\lambda_A - \sigma}$	$\sigma + \frac{1}{\lambda_B}$

Computational Complexity: Having k iterations, the number of arithmetic operations in the Power method is $O(4kn^2 + 3kn)$, so the Power method is not suitable for large sparse matrices. In order to reduce the computational complexity we propose a Power method with Monte Carlo iterations.

3 Monte Carlo Algorithms

3.1 Monte Carlo Iterations

Consider a matrix $A = \{a_{ij}\}_{i,j=1}^n, A \in R^{n \times n}$, and vectors $f = (f_1, \dots, f_n)^t \in R^{n \times 1}$ and $h = (h_1, \dots, h_n)^t \in R^{n \times 1}$. The algebraic transformation $Af \in$

$R^{n \times 1}$ is called *iteration* and plays a fundamental role in iterative Monte Carlo methods.

Consider the following Markov chain:

$$k_0 \rightarrow k_1 \rightarrow \dots \rightarrow k_i, \tag{3}$$

where $k_j = 1, 2, \dots, n$ for $j = 1, \dots, i$ are natural numbers. The rules for constructing the chain (3) are:

$$Pr(k_0 = \alpha) = \frac{|h_\alpha|}{\sum_{\alpha=1}^n |h_\alpha|},$$

$$Pr(k_j = \beta | k_{j-1} = \alpha) = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^n |a_{\alpha\beta}|}, \quad \alpha = 1, \dots, n.$$

Such a choice of the initial density vector and the transition density matrix leads to *almost optimal* Monte Carlo algorithms for matrix computations.

Define the random variables W_j using the following recursion formula:

$$W_0 = \frac{h_{k_0}}{p_{k_0}}, \quad W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \quad j = 1, \dots, i. \tag{4}$$

3.2 Direct Monte Carlo Algorithm

The dominant eigenvalue can be obtained using the iteration process mentioned in the introduction:

$$\lambda_{max} = \lim_{i \rightarrow \infty} \frac{(h, A^i f)}{(h, A^{i-1} f)},$$

where we calculate scalar products having in mind [7], [8], [4] that

$$(h, A^i f) = E\{W_i f_{k_i}\}, \quad i = 1, 2, \dots \tag{5}$$

Thus we have

$$\lambda_{max} \approx \frac{E\{W_i f_{k_i}\}}{E\{W_{i-1} f_{k_{i-1}}\}}$$

3.3 Balancing of Errors

There are two kind of errors in the Power method with Monte Carlo iterations:

- systematic error (from the Power method, (2)):

$$O\left(\left|\frac{\mu_2}{\mu_1}\right|^k\right),$$

where $\mu_i = \lambda_i$ if $B = A$, $\mu_i = \frac{1}{\lambda_i}$ if $B = A^{-1}$, $\mu_i = \lambda_i - \sigma$ if $B = A - \sigma I$, $\mu_i = \frac{1}{\lambda_i - \sigma}$ if $B = (A - \sigma I)^{-1}$ and λ_i and μ_i are the eigenvalues of A and B correspondingly;

- stochastic error (because we calculate mathematical expectations approximately, (1)):

$$O(D(\theta_N)N^{-1/2})$$

To obtain good results the stochastic error must be approximately equal to the systematic one. It is not necessary to use a *large* number of realizations N in order to have a *small* stochastic error if the systematic error is *large*.

3.4 Computational Complexity

Monte Carlo algorithms can be used to reduce the number of required operations to find the dominant eigenvalue. Dimov and Karaivanova have shown the mathematical expectation of the total number of operations for the Resolvent MC Method ([9], [10]) is:

$$ET_1(RMC) \approx 2\tau \left[(k + \gamma_A)l_A + \frac{1}{2}dl_L \right] lN + 2\tau n(1 + d), \quad (6)$$

where l is the number of moves in every Markov chain, N is the number of Markov chains, d is the mean value of the number of non-zero elements per row, γ_A is the number of arithmetic operations for calculating the random variable (in our code-realization of the algorithm $\gamma_A = 6$), l_A and l_L are arithmetic and logical sub-operations in one move of the Markov chain, k is the number of arithmetic operations for generating the random number (k is equal to 2 or 3).

The main term of (6) does not depend on n , the size of the matrix. This means that the time required for calculating the eigenvalue by RMC is practically independent of n . The parameters l and N depend on the spectrum of the matrix, but do not depend on its size n . The above mentioned result was confirmed for a wide range of matrices during the realized numerical experiments [10],[11].

4 Numerical Tests

The numerical tests are made on a cluster of 48 Hewlett Packard 900 series 700 Unix workstations under MPICH (version 1.1). The workstations are networked via 10Mb switched ethernet segments and each workstation has at least 64Mb RAM and run at least 60 MIPS. Each processor executes the same program for N/p number of trajectories, i.e. it computes N/p independent realizations of the random variable (here p is the number of processors). At the end the host processor collects the results of all realizations and computes the desired value. The computational time does not include the time for initial loading of the matrix because we consider our problem as part of a bigger problem (for example, *spectral portraits of matrices*) and suppose that every processor constructs it.

The test matrices are sparse and stored in *packed row format* (i.e. only nonzero elements). The results for average time and efficiency are given in table 2 and look promising. The relative accuracy is 10^{-3} [11].

We consider the *parallel efficiency* E as a measure that characterizes the quality of the proposed algorithms. We use the following definition:

$$E(X) = \frac{ET_1(X)}{pET_p(X)},$$

where X is a Monte Carlo algorithm, $ET_i(X)$ is the expected value of the computational time for implementation of the algorithm X on a system of i processors.

5 Discussion

5.1 A Numerical Method

An alternative numerical method to calculate the steady states is proposed by Goble [3].

Let us consider as an example a 1002 Duplex Control System which has 5 states. We apply a Markov model solution with a numerical approach (Steady State 1002 System) using failure rate values given in the transition matrix [1].

The limiting state probabilities for the system of eventual failure into state 3 and state 4 may be calculated as follows: The matrix P can be decomposed to form four sub-matrices of the form shown below:

$$P = \begin{bmatrix} Q & R \\ null & I \end{bmatrix}$$

Table 2. Direct Monte Carlo Algorithm using MPI (number of trajectories - 100000).

	matrix n=128	matrix n=1024	matrix n =2000
1 pr T (ms)	34	111	167
2 pr T (ms) E	17 1	56 0.99	83 1
3 pr T (ms) E	11 1.03	37 1	56 1
4 pr T (ms) E	8 1.06	27 1.003	42 1
5 pr T(ms) E	7 0.97	21 1.06	35 0.96

Where the upper-left sub-matrix Q is used to calculate the "Mean Time to Failure" or MTTF. The lower left matrix is a null matrix that contains only zeros and the lower right matrix is an "Identity" matrix. The upper-right matrix is called the R matrix and relates transition from the transient states to the absorbing states.

$$Q = \begin{bmatrix} 0.99999005 & 0.0000039537 & 0.0000006363 \\ 0.125 & 0.87499450 & 0.0 \\ 0.0 & 0.0 & 0.9999945 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.000005105 & 0.000000255 \\ 0.00000295 & 0.00000255 \\ 0.00000295 & 0.00000255 \end{bmatrix}$$

The matrix P is truncated to obtain a matrix Q formed by crossing-out the rows and columns of the absorbing states of transition matrix P . The Q matrix is then subtracted from the identity matrix I to form matrix $I - Q$. The above $I - Q$ matrix is then inverted to obtain the matrix N .

$$N = \begin{bmatrix} 166764.67 & 5.27 & 19293.16 \\ 166757.33 & 13.27 & 19292.31 \\ 0.0 & 0.0 & 181818.18 \end{bmatrix}$$

Multiplying the N matrix by the R matrix leads to the limiting state probabilities for the system. (a point where $P^{n+1} = P^n$ and the transition probabilities will not change for higher powers of matrix P).

$$NR = \begin{bmatrix} 0.908264011 & 0.091735989 \\ 0.908247648 & 0.091752352 \\ 0.536363636 & 0.463636364 \end{bmatrix}$$

The values in the elements of the matrix NR relate to the R sub-matrix in P as shown below.

Thus, if the system starts in state "0" it will end in the state given by $[1 \ 0 \ 0 \ 0 \ 0] * P = [0 \ 0 \ 0 \ 0.908264011 \ 0.091735989]$

Similarly, starting from state 1, the multiplying row vector would be $[0 \ 1 \ 0 \ 0 \ 0]$ resulting in row 2 of matrix NR , $[0 \ 0 \ 0 \ 0.908247648 \ 0.091752352]$, and starting from state 3, the multiplying row vector would be $[0 \ 0 \ 1 \ 0 \ 0]$ resulting in row 3 of matrix NR , $[0 \ 0 \ 0 \ 0.536363636 \ 0.463636364]$.

Hence, the limiting state probability that the system fails with outputs de-energised (state 3) is 0.908264011. The limiting state probability the system fails with outputs energised "fail-to-danger" (state 4) is 0.091735989.

5.2 Comparison of Methods

Whereas, the Monte Carlo solution for the eigenvalues of the system is applicable to large scale problems, methods such as Goble's become impracticable. As we have seen the parallel Monte Carlo method can be used to find several of the maximal eigenvalues efficiently or, if necessary, all eigenvalues. In the case where we seek all the eigenvalues then we can use the spectral resolution representation, and we are able to calculate the relevant P^n and thus the steady state of the system. The method is advantageous in that it is the magnitude of the underlying eigenvalues (in particular the dominant eigenvalue(s)) that govern both the manner and the rate of degeneracy of these systems to their limiting state probabilities.

6 Conclusion

Parallel Monte Carlo algorithms for calculating eigenvalues are outlined. The application of these methods for the calculation of the steady states of safety systems is considered.

They can be applied for well balanced matrices (which have nearly equal sums of elements per row) in order to provide good accuracy. We propose to use them when it is required to calculate the dominant eigenvalue or several of the largest eigenvalues of very large sparse matrices. This is because the computational time is almost independent of the dimension of the matrix and also their parallel efficiency is superlinear.

7 Acknowledgements

This work was partially supported by the Health and Safety Executive, UK. Thanks to V. N. Alexandrov for his fruitful discussion and comments.

References

1. R. Gallagher, , MSc Dissertation, Liverpool, 1986.
2. G. H. Golub, Ch. F. Van Loon, *Matrix Computations*, **The Johns Hopkins Univ. Press**, Baltimore and London, 1996.
3. W. M. Goble, *Evaluating Control Systems Reliability*, **Instrument Society of America**, 1992 pp 235-270.
4. J.H. Halton, *Sequential Monte Carlo Techniques for the Solution of Linear Systems*, **TR 92-033**, University of North Carolina at Chapel Hill, Department of Computer Science, 46 pp., 1992.
5. G.A. Mikhailov, *A new Monte Carlo algorithm for estimating the maximum eigenvalue of an integral operator*, **Docl. Acad. Nauk SSSR**, **191**, No 5 (1970), pp. 993 – 996.
6. G.A. Mikhailov, *Optimization of the "weight" Monte Carlo methods* (Nauka, Moscow, 1987).
7. I.M. Sobol, *Monte Carlo numerical methods*, **Nauka**, Moscow, 1973.
8. V.S.Vladimirov, *On the application of the Monte Carlo method to the finding of the least eigenvalue, and the corresponding eigenfunction*, of a linear integral equation, in Russian: **Teoriya Veroyatnostej i Yeye Primenenie**, **1**, No 1 (1956),pp. 113 – 130.
9. I. Dimov, A. Karaivanova and P. Yordanova, *Monte Carlo Algorithms for calculating eigenvalues*, **Springer Lectur Notes in Statistics**, **v.127** (1998) (H. Niederreiter, P. Hellekalek, G. Larcher and P. Zinterhof, Eds)), pp.205-220.
10. I. Dimov, A. Karaivanova *Parallel computations of eigenvalues based on a Monte Carlo approach*, **Journal of MC Methods and Appl.**, 1998 (to appear).
11. I. Dimov, A. Karaivanova, V. N. Aleksandrov *Performance Analysis of Monte Carlo Algorithms for Eigenvalue Problem*, **ILAS Conference**, Barcelona, 1999.
12. Megson, G., V. Aleksandrov, I. Dimov, *Systolic Matrix Inversion Using a Monte Carlo Method*, **Journal of Parallel Algorithms and Applications** , **3**, No 1 (1994), pp. 311-330.