Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2014), *28*, 311–322.
© Cambridge University Press 2014 0890-0604/14 $25.00
doi:10.1017/S0890060414000493

A robust system reliability analysis using partitioning and parallel processing of Markov chain

PO TING LIN,[1] YU-CHENG CHOU,[2] YUNG TING,[3] SHIAN-SHING SHYU,[4] AND CHANG-KUO CHEN[4]

[1]Department of Mechanical Engineering, Research and Development Center for Microsystem Reliability, Center for Biomedical Technology, Chung Yuan Christian University, Chungli City, Taiwan
[2]Institute of Undersea Technology, National Sun Yat-sen University, Kaohsiung City, Taiwan
[3]Department of Mechanical Engineering, Chung Yuan Christian University, Chungli City, Taiwan
[4]Institute of Nuclear Energy Research, Atomic Energy Council, Chungli City, Taiwan

(Received February 18, 2014; Accepted May 5, 2014)

Abstract

This paper presents a robust reliability analysis method for systems of multimodular redundant (MMR) controllers using the method of partitioning and parallel processing of a Markov chain (PPMC). A Markov chain is formulated to represent the N distinct states of the MMR controllers. Such a Markov chain has N^2 directed edges, and each edge corresponds to a transition probability between a pair of start and end states. Because N can be easily increased substantially, the system reliability analysis may require large computational resources, such as the central processing unit usage and memory occupation. By the PPMC, a Markov chain's transition probability matrix can be partitioned and reordered, such that the system reliability can be evaluated through only the diagonal submatrices of the transition probability matrix. In addition, calculations regarding the submatrices are independent of each other and thus can be conducted in parallel to assure the efficiency. The simulation results show that, compared with the sequential method applied to an intact Markov chain, the proposed PPMC can improve the performance and produce allowable accuracy for the reliability analysis on large-scale systems of MMR controllers.

Keywords: Fault Tolerance; Markov Chain; M-Partitioning; Parallel Processing; Reliability Analysis

1. INTRODUCTION

For the purpose of reliability evaluation in system designs, different representative methods have been developed over the years, including the fault tree, reliability block diagram, reliability graph, event sequence diagram (Mutha et al., 2013), Bayesian network, and Markov chain. Some literature has indicated that, mainly due to its flexibility, the Markov chain is the most suitable technique to analyze the reliability of redundant systems (Liu & Rausand, 2011).

Two configurations of subsea blowout preventer control systems, including triple modular redundancy and double dual modular redundancy control systems, were evaluated in terms of the probability of failure on demand, availability, and reliability using the Markov method (Cai et al., 2012). A Markov chain based methodology was proposed to perform the reliability, availability, maintainability, and safety for a hybrid redundancy system modeled as a stochastic Petri net (Liu et al., 2013). A Markov modeling approach was applied to the reliability analysis for a patented modular converter system that has multiple identical and interchangeable power converter modules in a wind turbine (Zhang et al., 2013). The comparative reliability study concerning fault coverage on the all-voting triple modular redundancy system, dual-duplex system, and double 2-out-of-2 system was conducted using the discrete-time Markov chain (Kim et al., 2005; Wang et al., 2007). The availability, production rate, and reliability of multistate degraded systems subjected to minimal repairs and imperfect preventive maintenance were evaluated via the continuous-time Markov chain (Soro et al., 2010). The reliability and benefits for a programmable logic controller hot standby system, which has the manager–slave architecture and two types of repair mechanisms, were evaluated using a semi-Markov approach (Parashar & Taneja, 2007). A scalable technique based on the discrete-time Markov chain was developed to evaluate the reliability for both nonredundant and structural redundancy-based large circuit designs (Bhaduri et al., 2007). A redundant system, designed with a special failure dependency (i.e., the redundant dependency), was

Reprint requests to: Yu-Cheng Chou, Institute of Undersea Technology, National Sun Yat-sen University, 70 Lienhai Road, Kaohsiung City 80424, Taiwan. E-mail: ycchou@mail.nsysu.edu.tw

evaluated with respect to the system reliability through a continuous-time Markov model (Yu et al., 2007). Markov chain models and bifurcation analysis were used to compare the degree of redundancy and system reliability for several logic redundancy schemes, including von Neumann's multiplexing logic, N-tuple modular redundancy, and interwoven redundant logic (Han et al., 2005). In addition to redundant systems, different Markov models, including continuous-time/discrete-time and full/semi/quasi Markov chains, have also been applied to study issues on the reliability, safety, availability, and diagnosis of different other systems (Zhang et al., 2003; Dominguez-Garcia et al., 2006; Guo & Yang, 2008; Liu et al., 2011; Lisnianski et al., 2012; Matthews & Philip, 2012). However, no literature has been reported with respect to partitioning and reordering a Markov chain to enable seamless incorporation of parallel computing techniques.

This paper presents a robust reliability analysis method, called the partitioning and parallel processing of a Markov chain (PPMC), for systems of multimodular redundant (MMR) controllers. A Markov chain (MC) is formulated to represent the N distinct states of the MMR controllers. Such an MC graph has N^2 directed edges. Each edge weight represents the transition probability from the start state to the end state. A system of β modular controllers is shown in Figure 1. Each of the β controllers processes signals from α sensors. Each of the α sensors has γ states. Thus, the number of distinct states for this system of MMR controllers is calculated as $N = \alpha^{\beta\gamma}$. Because N can be drastically increased by any of the three factors, the system reliability evaluation, which is based on the N^2 edge weights, may require large computational resources, such as the central processing unit (CPU) usage and memory occupation.

To this end, techniques of partitioning and reordering an MC have been conducted to reduce the complexity of an MC graph. Lightly weighted edges are suspended based on the threshold parameter, and the MC graph is partitioned and reordered to produce multiple weakly connected subgraphs, which can be processed independently in parallel. Therefore, parallel computing techniques can be applied to improve or ensure the calculation efficiency. In this paper, the task-farming paradigm is adopted for parallel processing. The task-farming paradigm has a manager–workers pattern. The manager is responsible for decomposing a task, distributing subtasks among workers, and gathering partial results from workers to coordinate the final calculation for the reliability evaluation. The worker processes execute in a simple cycle: get a subtask, process the subtask, and send the result to the manager.

The efficiency and accuracy of reliability analysis depend on the suspension threshold, the partition level, and the availability of parallel processors. The worst-case reliability is evaluated to compensate for inaccuracy due to the suspended edge weights. Higher suspension thresholds produce simpler MC models and faster calculations but more conservative reliabilities. The performance of PPMC is maximized to find the optimal suspension and partition levels. The proposed methodology has been successfully integrated with the detection and diagnosis processes in the fault-tolerant system. The simulation results show that, compared to the sequential method applied to an intact MC, the proposed PPMC can improve the performance and produce allowable accuracy for the reliability analysis on large-scale systems of MMR controllers.

2. RELIABILITY ANALYSIS OF MMR CONTROLLERS

Suppose the MMR controllers have N distinct states $S = \{1, 2, \ldots, N\}$ and the transitions between each state are represented in a MC. Figure 2 illustrates the graph $G(S, E)$ of the MC where the edge set E is composed of N^2 directed edges: $\{E_{1,1}, E_{1,2}, \ldots, E_{i,j}, \ldots, E_{N,N}\}$. Each directed edge has two edge weights $\mu_{i,j}$ and $Q_{i,j}$. The prior one is the transition probability that the state i moves to the state j while the later one represents the rate of change of the transition probability. The total transition probability that the state i moves to every state, including itself, equals 1, namely,

$$\sum_{j=1}^{N} \mu_{i,j} = 1 \quad \forall i. \tag{1}$$

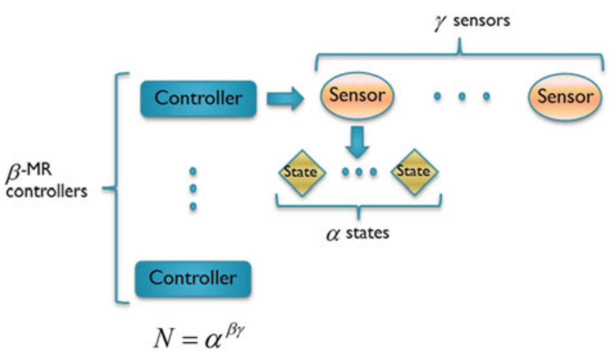


Fig. 1. N-state Markov chain for multimodular redundant controllers.

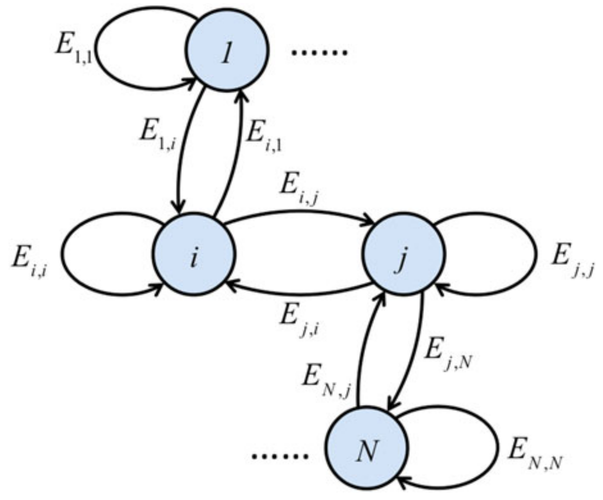


Fig. 2. N-state Markov chain for multimodular redundant controllers.

Therefore, the total rate of change of probability is zero, as in

$$\sum_{j=1}^{N} Q_{i,j} = 0 \quad \forall i. \tag{2}$$

Given a vector of the initial state probabilities,

$$\mathbf{p}(0) = 1\mathbf{e}_1 + \sum_{i=2}^{N} 0\mathbf{e}_i,$$

at the time $t = 0$, the vector of the state probabilities,

$$\mathbf{p}(t) = \sum_{i=1}^{N} p_i(t)\mathbf{e}_i,$$

is desirable for the evaluation of the system reliability.

For most MCs, the time-dependent state probabilities can be evaluated using the described method. However, the computational complexity increases dramatically when the number of states increases in the MMR controllers. It is very difficult to solve the large-scale eigenvalue problem and obtain the state probabilities in allowable working time. To this end, we will focus on the discrete-time approach, which requires less calculation.

2.1. Discrete-time MC

With the Markov property, the state variable $X(n+1)$ at the time $n+1$ depends only upon its state at the time n, $X(n)$ That is, given the present state of the system $X(n)$, the future state of the discrete-time MC (DTMC) $X(n+1)$ is independent of its previous discrete-time states $X(0)$, $X(1)$, . . . , $X(n-1)$. A DTMC with the finite state space S is time homogeneous if Equation (3) holds.

$$\mu_{i,j} \equiv P[X(n+1) = j|X(n) = i] = P[X(1) = j|X(0) = i] \tag{3}$$

for $n \geq 0$ and $i, j = 1 \ldots N$. For the DTMC, $\mu_{i,j}$ is called the one-step transition probability at the time n. In this paper, we assume the one-step transition probability remains constant for all times n. The state probabilities will be calculated using the given one-step transition probabilities.

We are interested in the probabilities

$$P[X(n) = j] \quad \forall j \in S \quad \text{and} \quad n \geq 0.$$

Statistically, the probability

$$P[X(n) = j]$$

is calculated by

$$P[X(n) = j] = \sum_{i=1}^{N} P[X(0) = i]P[X(n) = j|X(0) = i]. \tag{4}$$

Equation (4) is further rewritten in the matrix form as in

$$\mathbf{p}(n) = \mathbf{A}(n) \times \mathbf{p}(0), \tag{5}$$

where

$$\mathbf{p}(n) = \sum_{j=1}^{N} P[X(n) = j]\mathbf{e}_j$$

indicates the state probabilities at n;

$$\mathbf{p}(0) = \sum_{i=1}^{N} P[X(0) = i]\mathbf{e}_i$$

is for the initial probabilities; and

$$\mathbf{A}(n) = \sum_{i=1}^{N}\sum_{j=1}^{N} P[X(n) = j|X(0) = i]\mathbf{e}_j\mathbf{e}_i$$

is the matrix of the n-step transition probabilities. Because the DTMC is assumed to be time homogeneous, the matrix $\mathbf{A}(n)$ can be calculated by the nth power of $\mathbf{A}$ (Stewart, 2009), as in

$$\mathbf{p}(n) = \mathbf{A}^n \times \mathbf{p}(0). \tag{6}$$

To sum up, the transition probabilities of the DTMC are calculated by the matrix operations in Equation (6). The matrix multiplication algorithm has message-passing characteristics and is more applicable in message-passing systems, such as distributed memory based cluster of computers. In Section 3, the MC is partitioned into smaller systems in order to distribute the loads to several computers. In Section 4, the parallel processing of the MC is introduced.

3. PARTITIONING OF A MC

This section presents the partitioning of a large-scale MC. The lightly weighted edges are suspended based on the threshold parameters, and the MC is partitioned into multiple weakly connected subgraphs using the sparse matrix partitioning. The created subgraphs can be processed individually to produce partial results, which are combined to form the complete state probability distribution at a desired time instant.

3.1. Reduction of a MC

In this paper, we want to first reduce the complexity of the MC in terms of eliminating the connectivity of the weakly linked edges. Chou and Lin (2014) have eliminated the edges of minimal probabilities in the MC and greatly reduced the required calculations. The evaluated network probability of the reduced MC (RMC) was slightly underestimated and can be utilized as a worst-case probability, that is, the true probability of a MC is at least higher than or equal to the worst-case probability in the RMC.

A binary dependency matrix $\mathbf{D}$ represents the connectivity of the MC, as in

$$\mathbf{D} = \sum_{i=1}^{N}\sum_{j=1}^{N} D_{i,j}\mathbf{e}_j\mathbf{e}_i, \tag{7}$$

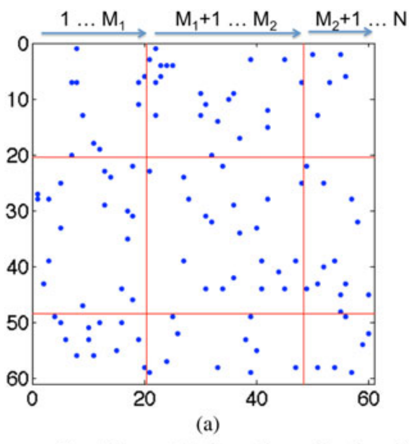
where $D_{i,j} = 1$ when the ith state is directed to the jth one; otherwise, $D_{i,j} = 0$. It is noted that $D_{i,j} = D_{j,i}$ only when there exist a reversible transition between the nodes i and j; however, $\mu_{i,j}$ and $\mu_{j,i}$ do not need to be the same. Given a threshold parameter τ, the connectivity $D_{i,j}$ of the RMC is reduced to zero if $\mu_{i,j} < \tau$. As a result, the dependency matrix becomes sparse. We want to reorder the sparse dependency matrix and partition them into smaller matrices in order to distribute the computational workloads to multiple computers.

3.2. M-partitioning

Figure 3a shows an example of the partition of a 60×60 dependency matrix. Each blue dot represents a nonzero element in the matrix. The four red lines divide the matrix into 9 smaller matrices, including a $M_1 \times M_1$ submatrix, a $M_1 \times (M_2 - M_1)$ submatrix, . . . , and a $(N - M_2) \times (N - M_2)$ submatrix. Each submatrix is required for the calculations of the state probabilities in Equation (6). Given the initial permutation of the sparse matrices

$$\prod = \sum_{i=1}^{N} i\mathbf{e}_i,$$

we focus on finding the optimal permutation to minimize the connectivity between the partitioned systems. In other words,



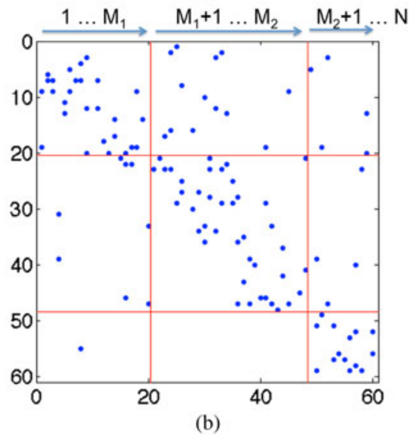


Fig. 3. Partitions of (a) an original and (b) a reordered dependency matrices.

we want to reorder the dependency matrix such that some submatrices are closer to zero matrices, as shown in Figure 3b. In this way, the required matrix multiplications are reduced.

Define a M-partition that partitions the dependency matrix of the RMC, $\mathbf{D}$, into two subsystems:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{\Phi}_1 \\ \mathbf{\Phi}_2 & \mathbf{D}_2 \end{bmatrix}, \tag{8}$$

where $\mathbf{D}_1$ and $\mathbf{D}_2$ are $M \times M$ and $(N - M) \times (N - M)$ submatrices, respectively. The sizes of the off-diagonal matrices $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are $M \times (N - M)$ and $(N - M) \times M$, respectively. Meanwhile, the permutation vector is partitioned into two vectors:

$$\prod\nolimits_1 = \sum_{i=1}^{M} i\mathbf{e}_i \quad \text{and} \quad \prod\nolimits_2 = \sum_{i=M+1}^{N} i\mathbf{e}_i.$$

The partition parameter M is controlled to balance the computational loads for each computer. When $M < N/2$, the computational complexity of the first subproblem is lower; when $M > N/2$, the second subproblem is more complex. Multiple matrix partitions can be accomplished using multiple M-partitions.

Based on the M-partition, the calculation in Equation (9) is partitioned as in

$$\begin{bmatrix} \mathbf{p}_1(n) \\ \mathbf{p}_2(n) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{\Psi}_1 \\ \mathbf{\Psi}2 & \mathbf{A}_2 \end{bmatrix}^n \begin{bmatrix} \mathbf{p}_1(0) \\ \mathbf{p}_2(0) \end{bmatrix}, \tag{9}$$

where

$$\mathbf{A}_k = \sum_{i \in \mathbf{P}_k} \sum_{j \in \mathbf{P}_k} D_{i,j}\mu_{i,j}\mathbf{e}_j\mathbf{e}_i$$

is calculated based on the permutation $\mathbf{\Pi}_k$. Furthermore, the off-diagonal matrices $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are calculated by

$$\sum_{i \in \mathbf{P}_1} \sum_{j \in \mathbf{P}_2} D_{i,j}\mu_{i,j}\mathbf{e}_j\mathbf{e}_i$$

and

$$\sum_{i \in \mathbf{P}_2} \sum_{j \in \mathbf{P}_1} D_{i,j}\mu_{i,j}\mathbf{e}_j\mathbf{e}_i,$$

respectively. The state probabilities in the first and second subproblems are then determined by

$$\mathbf{p}_1(n) = \mathbf{A}_1^n\mathbf{p}_1(0) + f_1(\mathbf{\Psi}_1, \mathbf{\Psi}_2), \tag{10}$$

$$\mathbf{p}_2(n) = \mathbf{A}_2^n\mathbf{p}_2(0) + f_2(\mathbf{\Psi}_1, \mathbf{\Psi}_2). \tag{11}$$

If the permutations of the matrices and vectors are reordered such that the total probability in the off-diagonal matrices $\mathbf{\Psi}_1$

and $\mathbf{\Psi}_2$ are close to zero, the functions f_1 and f_2 in Equations (10) and (11) become negligible. Thus, the state probabilities can be approximated by the following equation:

$$\mathbf{p}_i(n) \cong \mathbf{A}_i^n \mathbf{p}_i(0). \tag{12}$$

3.3. Iteration process of finding the minimal coupling between subproblems

The objective of our approach is to find the permutation order that minimizes the total transition probability between the two subsystems:

$$p_C = \sum_{i=1}^{M}\sum_{j=1}^{N-M}\Psi_{1,ij} + \sum_{i=1}^{N-M}\sum_{j=1}^{M}\Psi_{2,ij}. \tag{13}$$

One of the possible methods is to diagonalize the dependency matrix $\mathbf{D}$; however, it does not ensure the coupling probability p_C is minimized for the specific M-partition. In this paper, an effective iteration process is proposed as in the following steps to determine the optimal permutation order:

STEP 1. Given the initial permutation $\mathbf{\Pi}^{(0)}$ and dependency matrix $\mathbf{D}^{(0)}$, calculate the initial coupling probability $p_C^{(0)}$ using Equation (13).

STEP 2. Begin with $\Psi_{1,N-M}^{(0)}$ in the upper-right off-diagonal dependency matrix as the pivot element. The iteration number k equals one.

STEP 3. Suppose the location of the pivot element is (i, j). If the (i, j)th element in $\Psi_1^{(k-1)}$, or the (j,i)th element in $\Psi_2^{(k-1)}$, is not zero, the column number of the pivot element is denoted as $c_p^{(k)}$; go to step 5. Otherwise, go to step 4.

STEP 4. If the pivot element is the last element $\Psi_{M,1}^{(k-1)}$, the process cannot advance further; go to step 10. Otherwise, move to the adjacent element along the zig-zag trajectory, shown in Figure 4, and consider it as the new pivot element. Go to step 3.

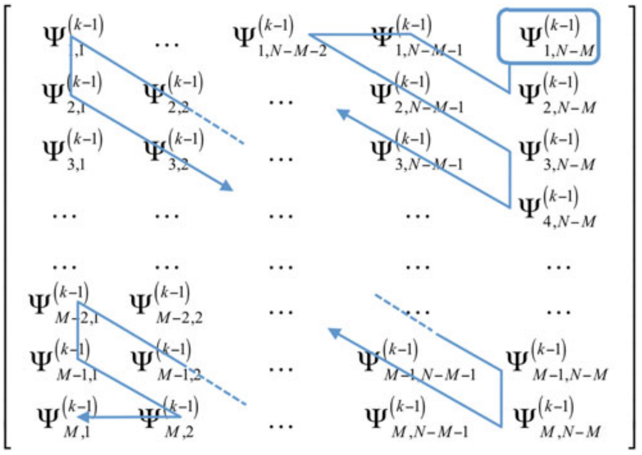


Fig. 4. Determination of pivot elements along the zig-zag trajectory in the off-diagonal dependency matrix.

STEP 5. Begin with the first column, that is, $c = 1$. Let $\mathbf{\Pi}^{(k)} = \mathbf{\Pi}^{(k-1)}$ and $n_C^{(k)} = n_C^{(k-1)}$.

STEP 6. If $c = c_p^{(k)}$, move to the adjacent column by $c = c + 1$. If $c \leq N$, go to step 7; otherwise, go to step 9.

STEP 7. Interchange c and $c_p^{(k)}$ in the permutation $\mathbf{\Pi}^{(k-1)}$ and calculate the coupling probability p_1 of the interchanged off-diagonal dependency matrix.

STEP 8. If $n_1 < n_C^{(k)}$, $n_C^{(k)} = n_1$ and assign the interchanged permutation as $\mathbf{\Pi}^{(k)}$. Advance to the adjacent column by $c = c + 1$ and go to step 6.

STEP 9. If $n_C^{(k)} = n_C^{(k-1)}$, the iteration process has converged. Go to step 10. Otherwise, update the interchanged matrix $\mathbf{D}^{(k)}$ using the determined permutation $\mathbf{\Pi}^{(k)}$ and advance to the next iteration by $k = k + 1$. Go to step 4.

STEP 10. The optimal permutation is $\mathbf{\Pi}^* = \mathbf{\Pi}^{(k)}$ and the optimal reordered dependency matrix is $\mathbf{D}^* = \mathbf{D}^{(k)}$.

4. PPMC

As illustrated in Section 3.2, the partitioned MC has multiple transition probability submatrices. As also mentioned in Section 3, these square matrices can be processed individually to produce partial results, which are combined to form the complete state probability distribution at a desired time instant. Thus, in this paper, the task-farming paradigm is adopted for parallel-processing purposes. The task-farming paradigm has a manager–workers pattern, as shown in Figure 5. The manager is responsible for decomposing a task, distributing subtasks among workers, and gathering partial results from workers to coordinate the final calculation for the reliability evaluation. The worker processes execute in a simple cycle: get a subtask, process the subtask, and send the result to the manager. In addition, due to the task-farming paradigm's dynamic load balancing characteristic, the proposed methodology can respond well to the failure of some processors. This feature facilitates the creation of robust applications that are able to survive from the loss of workers or even the manager. In this paper, for simplicity purposes, with a static load balancing feature, the task-farming paradigm is employed to obtain the state probability vector at any desired time instant.

For a large-scale MC that has been partitioned, each of its transition probability submatrices might still need a large memory space for processing on a single processor. Therefore, another level of parallelism might be required to release or alleviate this memory burden. In addition, the MC-based reliability evaluation is essentially a series of matrix multiplications. Moreover, the message-passing programming paradigm is one of the earliest and most widely used approaches for programming parallel computers, mainly because it imposes minimal requirements on the underlying hardware (Grama et al., 2004). Furthermore, the most natural message-passing architecture for matrix operations is a two-dimensional mesh, where each node in the mesh computes

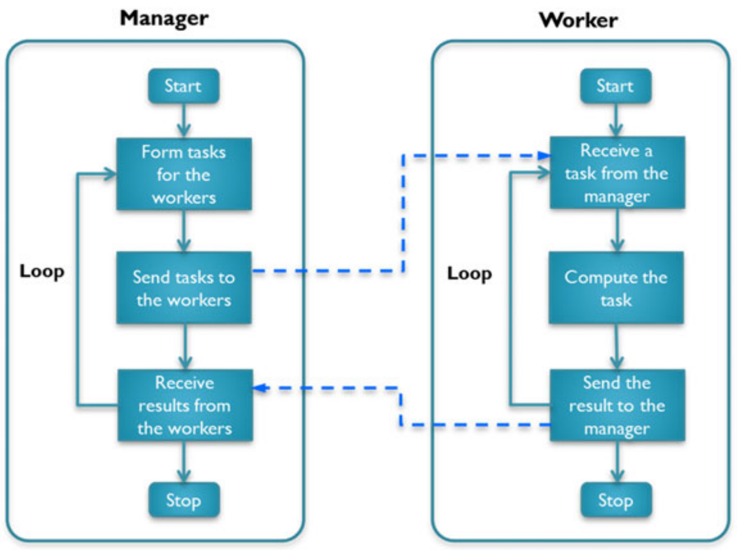

Manager
Worker
Start
Form tasks for the workers
Send tasks to the workers
Receive results from the workers
Stop
Loop
Start
Receive a task from the manager
Compute the task
Send the result to the manager
Stop
Loop

Fig. 5. Basic flowcharts for the manager and worker programs in the task-farming paradigm.

one element, or one submatrix, of the resultant array. The mesh connections allow messages to pass between adjacent nodes in the mesh simultaneously. Thus, in this paper, the Cannon algorithm (Cannon, 1969), a memory efficient algorithm based on the two-dimensional mesh framework, is adopted for the matrix multiplication. The Cannon algorithm uses a mesh of nodes with wraparound connections to shift submatrices. In this paper, a node in a mesh represents a processor in a cluster computer. An important point that has to be mentioned is that the Cannon algorithm can also be applied to a nonpartitioned MC for reliability evaluation purposes, which is how this algorithm is used in this paper.

For clarity, elements, instead of submatrices, of the arrays **a** and **b** are used to illustrate the Cannon algorithm as follows:

1. Initially, processor $\pi_{i,j}$ has elements $a_{i,j}$ and $b_{i,j}$ $(0 \leq i < n, 0 \leq j < n)$.
2. Elements are moved from their initial position to an aligned position. In other words, the complete ith row of **a** is left-shifted i positions, and then the complete jth column of **b** is upshifted j positions. This step places the element $a_{i,j+i}$ and the element $b_{i+j,i}$ in processor $\pi_{i,j}$. This pair of elements are required to calculate the element $c_{i,j}$.
3. Each processor $\pi_{i,j}$ multiplies its elements.
4. The ith row of **a** is shifted one position left, and the jth column of **b** is shifted one position upward. This step brings together the adjacent elements of **a** and **b**, which are also required in the computation of $c_{i,j}$.
5. Each processor $\pi_{i,j}$ multiplies the elements brought to it and adds the result to the accumulating sum.
6. Repeat Steps 4 and 5 until the final result of $c_{i,j}$ is obtained. In other words, given n rows and n columns of elements, a total of $n - 1$ shifts need to be conducted.

Next, given that both **a** and **b** have s^2 submatrices, each submatrix has $m \times m$ elements. Thus, based on the above Cannon algorithm, for both **a** and **b**, the initial alignment requires a maximum of $s - 1$ shift operations. Afterward, there are other $s - 1$ shift operations required for computation purposes. Each shift operation performs a communication involving $m \times m$ elements. Therefore, the communication time t_{comm} can be determined using Equation (14):

$$t_{\text{comm}} = 4(s-1)\left(t_{\text{startup}} + m^2 t_{\text{data}}\right). \tag{14}$$

Thus, the Cannon algorithm has a communication time complexity of $O(sm^2) = O(mn)$, where $s = n/m$ is assumed in this paper.

As for the computation aspect, each submatrix multiplication requires m^3 multiplications and m^3 additions. Therefore, with $s - 1$ shifts as mentioned above, the computation time t_{comp} can be determined using Equation (15):

$$t_{\text{comp}} = 2sm^3 = 2m^2 n. \tag{15}$$

Hence, the Cannon algorithm has a computation time complexity of $O(m^2 n)$.

In this paper, the Cannon algorithm is implemented through the message-passing interface (MPI; Gropp et al., 1998; Snir et al., 1998). MPI is a specification for building a library that provides standard functions for writing portable and efficient message-passing programs to be run on a variety of parallel computers. In this paper, the selected MPI library is a widely used open source library called MPICH2 (Gropp, 2002).

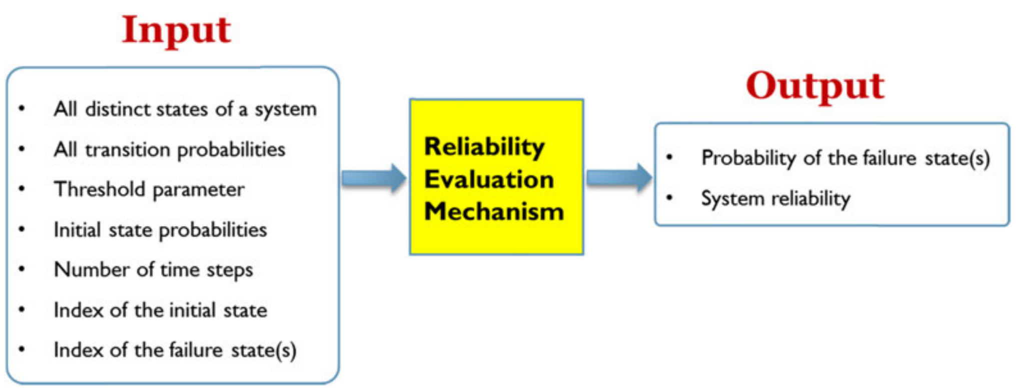

Input
All distinct states of a system
All transition probabilities
Threshold parameter
Initial state probabilities
Number of time steps
Index of the initial state
Index of the failure state(s)
Reliability Evaluation Mechanism
Output
Probability of the failure state(s)
System reliability

Fig. 6. Inputs and outputs of the parallel processing of a Markov chain reliability evaluation mechanism.

5. Numerical Examples

In this section, we study the MCs for three different examples to validate the proposed PPMC method. The inputs and outputs of the proposed PPMC reliability evaluation mechanism are shown in Figure 6. The inputs include all distinct states of a system, all transition probabilities, the threshold parameter, initial state probabilities, the number of time steps, the index of the initial state, and the index (or indices) of the failure state (or states). The outputs include the probability (or probabilities) of the failure state (or states), and the overall system reliability.

The PPMC reliability evaluation mechanism can be applied to all kinds of systems whose states can be modeled as a MC. Particularly, it is suitable for systems that potentially contain a vast number of distinct states, such as nuclear power systems (Aldemir et al., 2006). As shown in Figure 7, it is assumed that there are 14 sensors located at different components of a nuclear steam system. In addition, each sensor is assumed to have two states indicating that a component is either functioning well or not functioning at all. Thus, the number of distinct states of the system is $2^{14} = 16384$. Among these states, some of them can be defined as the failure states. The PPMC reliability evaluation mechanism can be used to efficiently generate, for a specified number of time steps, the failure states' probabilities and the overall system reliability. Therefore, if desired, engineers can perform necessary maintenance activities, in advance, on the components related to critical failure states. Afterward, the transition probability matrix can be changed by increasing or decreasing probability values on corresponding elements. Eventually, the PPMC reliability evaluation mechanism can be used to generate new results for necessary further improvements on the system reliability.

5.1. Example 1: 60-state MC for MMR controllers

The first example considers a randomly generated 60-state MC, which is composed of 277 distinct directed edges.

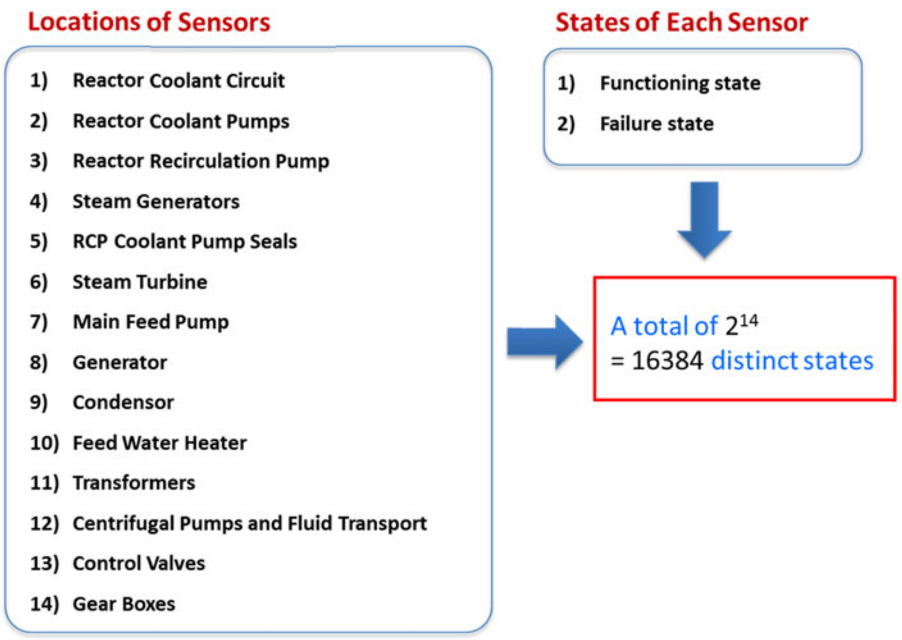


Fig. 7. A nuclear steam system containing a large number of different states.

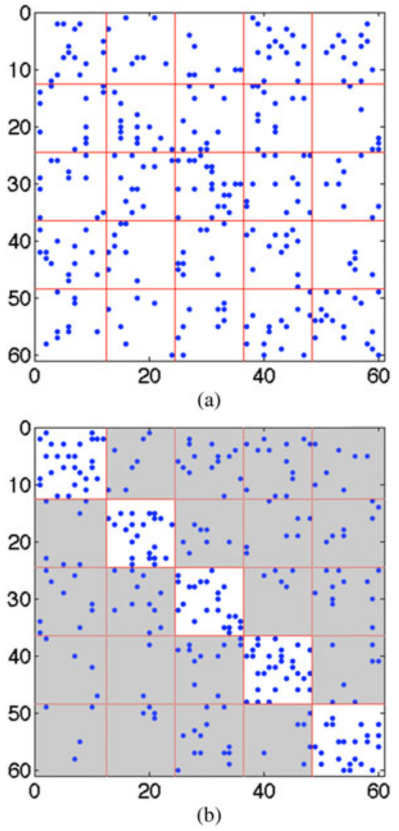


Fig. 8. The (a) original and (b) partitioned dependency matrices for the 60-state Markov chain.

Figure 8a shows the dependency matrix of the 60-state MC while each blue dot represents one of the directed connections. The blank areas stand for no connections between the states. Figure 8a can also be used to represent the transition matrix; in this case, each blue dot is a nonzero transition probability $\mu_{i,j}$.

When the size of MC is in the order of tens of states, it is not necessary to reduce the MC using the threshold parameter. In the latter case, when the MC size is larger, the reduction can help improve the numerical efficiency. To uniformly distribute the computational workloads to five computers, we partition the dependency matrix at the positions of $M = \{12, 24, 36, 48\}$. The red lines represent the desirable locations for matrix partitioning. According to the desirable partitioning locations, the dependency matrix is reordered such that the total probability in the off-diagonal submatrices, shown in the gray areas in Figure 8b, is minimized. The ratio of the total coupling probability and the total probability is $9.66/60 = 0.16$. In other words, around 84% of workloads are uniformly distributed in the five diagonal submatrices. The discrete-time state probabilities are calculated using Equation (7).

Suppose the initial condition is given as $p_{15}(0) = 1$ and $p_i(0) = 0$ for $i = 1 \ldots 14, 16, \ldots 60$, the state probabilities at time $n = 10$ are calculated using the proposed method. Furthermore, the numerical performances of the PPMC method

Table 1. *Results of the 60-state Markov chain*

Method	Index of Fail State	Probability of Fail State	System Reliability
Original	2	0.07%	99.93%
PPMC	2	12.5%	87.50%
Original	24	3.06%	96.94%
PPMC	24	12.5%	87.50%
Original	30	0.12%	99.88%
PPMC	30	0.13%	99.87%

Note: PPMC, Partitioning and parallel processing of Markov chain.

are compared with the calculations using the original MC in Equation (5). Three different simulation results are listed in Table 1. The first case considers the second state as the fail state. The original matrix multiplication shows the failure probability is 0.07%, while the proposed method overestimates the failure rate as 12.5%. A worst-case reliability, 87.50%, is then found, that is, the true probability, 99.93%, is at least larger than or equal to the underestimation, 87.50%.

The second case shows a higher failure probability of 3.06% when the 24th state is considered as the fail state. In this case, the proposed method is capable of finding the overestimated failure probability of 12.5% and the worst-case reliability of 87.5%. In the last case, the 30th state is considered as the fail state. In this case, the PPMC method obtains the worst-case reliability of 99.87%, which is just slightly lower than the true probability of 99.88%.

5.2. Example 2: 200-state MC for MMR controllers

The second case focuses on a 200-state MC, which contains 605 directed edges. Figure 9a shows the dependency matrix of the given MC. In this case, the MC reduction is also unnecessary. To distribute the workloads to five computers, the desirable partitioning locations are $M = \{40, 80, 120, 160\}$, indicated by the red lines. The off-diagonal probabilities are minimized to reorder the dependency matrix. As a result, the off-diagonal probability, the sum of probabilities in the gray areas, equals 34.33, which is around 17.16% over the total probability. The values in the off-diagonal submatrices will be neglected in Equation (7), yielding the underestimated measure of system reliability. Therefore, the worst-case reliability can be determined.

We now want to demonstrate different types of simulation results from the first example. Suppose the 5th state is considered as the fail state, then three different initial states are studied. The first case starts with $\mathbf{p}(0) = 1\mathbf{e}_{32}$. The original matrix multiplication shows the fail probability of 6.63% as the PPMC overestimates the fail probability of 7.05%; therefore, a worst-case reliability of 92.95% is determined. Another initial condition considers $\mathbf{p}(0) = 1\mathbf{e}_{18}$. The proposed method underestimates the reliability, that is, 82.83% < true probability = 83.25%, and utilizes the worst-case measure as a robust

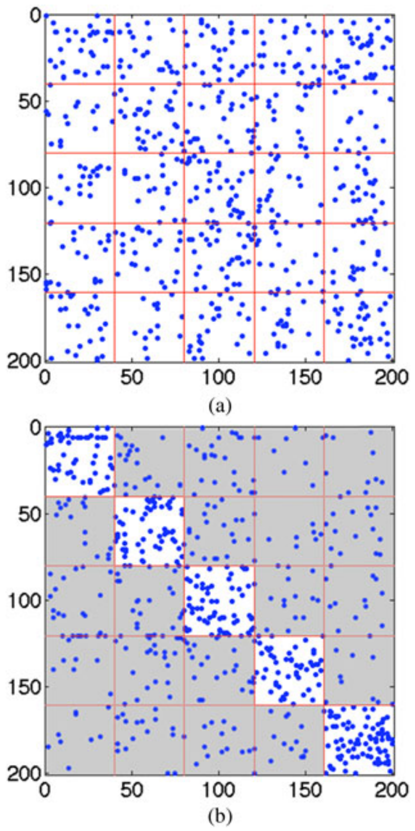


Fig. 9. The (a) original and (b) partitioned dependency matrices for the 200-state Markov chain.

estimator of the system probability. Finally, when the 99th state is considered as the initial state, a lower reliability of 77.01% is found using the original calculations. Even for the special situation of low reliability, the proposed method is capable of finding the worst-case reliability, that is, 75.32%. Table 2 lists the detailed information about the simulation results.

5.3. Example 3: 480-state MC for MMR controllers

The final example considers a large-scale MC that contains 480 states. Figure 10a shows the dependency matrix with 1899 blue dots, that is, there are 1899 distinct directed edges in the MC. In a large problem like this, the matrix reduction can effectively improve the numerical performances. We assume the threshold parameter is $\tau = 0.1$. Figure 10b shows the dependency matrix of the reduced MC, which now only contains 710 directed edges. The matrix permutation is then reordered and the resultant dependency matrix is shown in Figure 10c. The total off-diagonal probability equals 110.37, which is around 23% of the total probability. However, 77% of the workloads are uniformly distributed to eight different computers. The red lines represent the given partitioning locations $M = \{60, 120, 180, 240, 300, 360, 420\}$.

Table 3 lists two different cases of the simulations. The first case considers the 10th and 406th states as the initial and fail

Table 2. *Results of the 200-state Markov chain*

Method	Index of Initial State	Probability of Fail State	System Reliability
Original	32	6.63%	93.37%
PPMC	32	7.05%	92.95%
Original	18	16.75%	83.25%
PPMC	18	17.17%	82.83%
Original	99	22.99%	77.01%
PPMC	99	24.68%	75.32%

Note: PPMC, Partitioning and parallel processing of Markov chain.

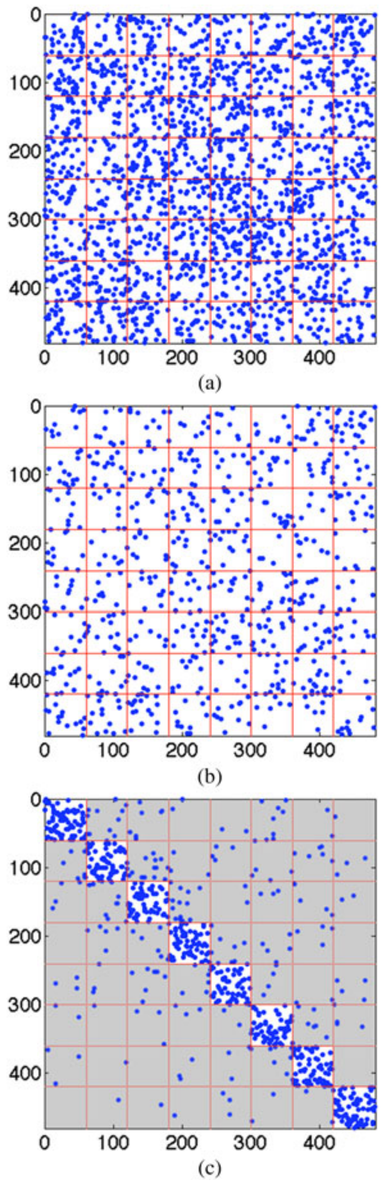


Fig. 10. The (a) original, (b) reduced, and (c) partitioned dependency matrices for the 480-state Markov chain.

Table 3. *Results of the 480-state Markov chain*

Method	Index of Initial State	Index of Fail State	Probability of Fail State	System Reliability
Original	10	406	0.25%	99.75%
PPMC	10	406	0.96%	99.04%
Original	432	31	0.10%	99.90%
PPMC	432	31	29.22%	70.79%

Note: PPMC, Partitioning and parallel processing of Markov chain.

Table 4. *Computation times for three scenarios*

	No. of CPUs	No. of Data Points/CPU	Computation Time (ms)	Speedup Ratio
Scenario 1	1	480×480	25501	1
Scenario 2	16	120×120	2397	10.6
Scenario 3	8	60×60	74	344.6

Note: PPMC, Partitioning and parallel processing of Markov chain.

states. The proposed method is able to find the worst-case reliability, 99.04%, which is slightly lower than the true measure, 99.75%. In the other case, in which the 432nd and 31st states are the initial and fail states, the PPMC method obtains an underestimation of the system reliability, that is, 70.79% << true probability = 99.90%, due to the errors from the matrix reduction and approximated calculation in Equation (7). However, the proposed method still guarantees that the true system reliability is at least larger than or equal to the underestimated measure.

5.4. Numerical performances in Example 3

A 25-node cluster computer is used to run example 3 in the following scenarios:

1. Evaluate the system reliability of the original MC at the 10th time step by serial matrix multiplication method using one CPU.
2. Evaluate the system reliability of the original MC at the 10th time step by the Cannon algorithm using 16 CPUs.
3. Evaluate the system reliability of the partitioned MC at the 10th time step by applying serial matrix multiplication method on eight diagonal submatrices using eight CPUs.

All of the nodes of the cluster computer have identical key specifications, including a dual core CPU, 1-GHz core CPU frequency, and 1.837 GB of memory. Due to homogeneous hardware features, the time spent by each CPU to finish its tasks is almost the same even though the computation time is still defined as the longest time spent by any CPU. Moreover, for each scenario, the computation time only changes very slightly in every execution. Thus, the average of 10 computation times is provided for each scenario. As shown in Table 4, in scenario 1, a single CPU needs to process 230,400 data points; in scenario 2, each of the 16 CPUs needs to process 14,400 data points; in scenario 3, each of the 8 CPUs needs to process 3600 data points. The average computation times for scenarios 1, 2, and 3 are 25501, 2397, and 74 ms, respectively. The speedup ratios for scenarios 1, 2, and 3 are 1, 10.6, and 344.6, respectively.

There are two factors that affect the accuracy of reliability evaluation: the reductions of directed edges with transition probabilities lower than the selected threshold parameter and the unconsidered edges in the off-diagonal matrix. Lin et al. (2014) have further investigated the numerical performance of PPMCs with various level of threshold parameter τ. In Example 3, 1899 distinct directed edges in Figure 10a deduced to 710 edges when $\tau = 0.1$ was considered. Fewer edges would be reduced as a lower threshold parameter is considered but the complexity of MC increases. More edges would be reduced as a higher threshold parameter is utilized but the accuracy of reliability evaluation may be jeopardized. A proper selection of threshold parameter has been found related to the distribution of transition probabilities in the MC. For more information, please refer to Lin et al. (2014).

6. Conclusions

In this paper, the method of PPMC has been presented to perform a robust reliability analysis for complex systems, such as MMR controllers. The number of different states N can be easily increased by three factors, including the number of modular redundant controllers, the number of sensors each controller manages, and the number of states each sensor indicates. The proposed method partitioned the complex MC and solved for a worst-case reliability using parallel computational processes. In PPMC, a threshold parameter has been chosen to reduce the dependency of MC in terms of eliminating the edges of negligible probabilities between different states. A M-partitioning procedure has been developed to gather together M states that are mutually related and decomposed the complex problem into multiple subproblems, which are lowly dependent from each other. In addition, calculations regarding the submatrices can be performed independently in parallel. This research is the first attempt to reduce, partition, and reorder a MC, in order to enable seamless incorporation of parallel computing techniques for resource-efficient calculations. Compared with the sequential method applied to a nonpartitioned MC, the proposed PPMC reduces the overall computation time, lowers the amount of memory required on a single processor, and produces allowable accuracy for the reliability analysis on large-scale systems of MMR controllers in the three demonstrated examples.

The main advantage of the proposed method is its capability of efficiently performing the reliability analysis of a MC in various engineering problems. It also assumes a worst-case scenario when compared to evaluation of the true system reliability, which is conservative in its nature. The proposed method is also anticipative in the advances in the field of computing sciences, which could further reduce the computational parameters. However, PPMC currently assumes the transition probabilities between different states are known and remain constant. Furthermore, the decision of threshold parameter for the reduction of MC is not automatically determined. It is expected that the numerical performances of PPMC depend on proper selection of the threshold parameter. The methodology developments include the adaptiveness to the dynamic transition probabilities between different states, the optimal selection of threshold parameter for MC reduction with the consideration of the distribution of the probability density, the enhancement of the numerical efficiency of the decomposition procedure, and the improvement of the numerical accuracy of the reliability analysis. These developments are expected to make the proposed method competitive in the implementation of evaluating and analyzing MCs in complex engineering systems.

ACKNOWLEDGMENTS

This research was supported by the Atomic Energy Council of Taiwan (Grant 1012001INER010), the Ministry of Science and Technology of Taiwan (Grants 101-2628-E-033-001-MY3, 102-2218-E-033-002-MY2, and 102-2221-E-033-020), the Research and Development Center for Microsystem Reliability at Chung Yuan Christian University, and the Center for Biomedical Technology at Chung Yuan Christian University.

NOMENCLATURE

$\mathbf{a}$	An array in the Cannon algorithm
$\mathbf{A}$	Transition matrix
$\mathbf{A}_i$	i-th partitioned transition matrix
$\mathbf{b}$	Another array in the Cannon algorithm
c	Column number
c_p	Column number of the pivot element in dependency matrix
C_i	ith coefficient
$\mathbf{D}$	$N \times N$ dependency matrix
$\mathbf{D}_i$	ith partitioned dependency matrix
D_{ij}	Connectivity parameter, where $D_{ij} = 1$ when the ith and jth states are connected; otherwise, it is zero.
$\mathbf{e}_i$	ith normal basis
m	Dimensional parameter of submatrices for multiple processors
M	Dimensional parameter for the partitioned matrix
n	Time parameter in the computer processing
N	Number of states in Markov chain
$\mathbf{p}$	Vector of transition probabilities
p_C	Total probability of the partitioned Markov chains couple with each other
$\mathbf{p}_i$	ith partitioned vector of transition probabilities
p_I	Total coupling probability for the interchanged dependency matrix
P(A)	Probability of A
P(A | B)	Conditional probability of A given B
$\mathbf{Q}$	Infinitesimal generator of continuous-time Markov chain
s	A dimensional parameter
S	Finite state space
t	Time
$X(t)$	Variable with Markov property at time t
λ_i	ith eigenvalue
$\mu_{i,j}$	Transition probability (failure/recovery rate) of the state i moves to the state j
$\mathbf{v}_i$	ith eigenvector
δ	A small number
$\pi_{i,j}$	A processor with elements $a_{i,j}$ and $b_{i,j}$
Π	$N \times 1$ permutation vector
Π_i	ith partitioned permutation vector
τ	Threshold parameter
Φ_i	ith off-diagonal dependency matrix
Ψ_i	ith off-diagonal transition matrix

Superscripts

$'$	Derivative with respect to time
$*$	Optimal solution
(k)	kth iteration in finding the optimal partition
(n)	nth step in computer processing

REFERENCES

Aldemir, T., Miller, D., Stovsky, M., Kirschenbaum, J., & Buccim, P. (2006). *Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments*. Washington, DC: US Nuclear Regulatory Commission.

Bhaduri, D., Shukla, S.K., Graham, P.S., & Gokhale, M.B. (2007). Reliability analysis of large circuits using scalable techniques and tools. *IEEE Transactions on Circuits and Systems I 54(11)*, 2447–2460.

Cai, B., Liu, Y., Liu, Z., Tian, X., Li, H., & Ren, C. (2012). Reliability analysis of subsea blowout preventer control systems subjected to multiple error shocks. *Journal of Loss Prevention in the Process Industries 25*, 1044–1054.

Cannon, L.E. (1969). *A cellular computer to implement the Kalman filter algorithm*. PhD Thesis. Montana State University.

Chou, Y.-C., & Lin, P.T. (2014). An efficient and robust design optimization of multi-state flow network for multiple commodities using generalized reliability evaluation algorithm and edge reduction method. *International Journal of Systems Science*. Advance online publication. doi:10.1080/00207721.2013.879228

Dominguez-Garcia, A.D., Kassakian, J.G., & Schindall, J.E. (2006). Reliability evaluation of the power supply of an electrical power net for safety-relevant applications. *Reliability Engineering & System Safety 91(5)*, 505–514.

Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2004). *Introduction to Parallel Computing*. Essex: Pearson Education.

Gropp, W. (2002). MPICH2: A new start for MPI implementations. *Proc. 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*.

Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., & Snir, M. (1998). *MPI: The Complete Reference—The MPI-2 Extensions*. Cambridge, MA: MIT Press.

Guo, H., & Yang, X. (2008). Automatic creation of Markov models for reliability assessment of safety instrumented systems. *Reliability Engineering & System Safety 93(6)*, 829–837.

Han, J., Gao, J., Jonker, P., Qi, Y., & Fortes, J.A. (2005). Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design & Test of Computers 22(4)*, 328–339.

Kim, H., Lee, H., & Lee, K. (2005). The design and analysis of AVTMR (all voting triple modular redundancy) and dual-duplex system. *Reliability Engineering & System Safety 88(3)*, 291–300.

Lin, P.T., Chou, Y.-C., Manuel, M.C.E., & Hsu, K.S. (2014). Investigation of numerical performance of partitioning and parallel processing of Markov chain (PPMC) for complex design problems. *Proc. ASME 2014 Int. Design & Engineering Technical Confs. and Computers & Information in Engineering Conf., IDETC/CIE 2014*, Paper No. DETC2014-34652, Buffalo, NY.

Lisnianski, A., Elmakias, D., Laredo, D., & Ben Haim, H. (2012). A multi-state Markov model for a short-term reliability analysis of a power generating unit. *Reliability Engineering & System Safety 98(1)*, 1–6.

Liu, Y., & Rausand, M. (2011). Reliability assessment of safety instrumented systems subject to different demand modes. *Journal of Loss Prevention in the Process Industries 24(1)*, 49–56.

Liu, Z., Liu, Y., Cai, B., Liu, X., Li, J., Tian, X., & Ji, R. (2013). RAMS analysis of hybrid redundancy system of subsea blowout preventer based on stochastic Petri nets. *International Journal of Security and Its Applications 7(4)*, 159–166.

Liu, Z., Ni, X., Liu, Y., Song, Q., & Wang, Y. (2011). Gastric esophageal surgery risk analysis with a fault tree and Markov integrated model. *Reliability Engineering & System Safety 96(12)*, 1591–1600.

Matthews, P., & Philip, A. (2012). Bayesian project diagnosis for the construction design process. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 26(4)*, 375–391.

Mutha, C., Jensen, D., Tumer, I., & Smidts, C. (2013). An integrated multi-domain functional failure and propagation analysis approach for safe system design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 27(4)*, 317–347.

Parashar, B., & Taneja, G. (2007). Reliability and profit evaluation of a PLC hot standby system based on a master-slave concept and two types of repair facilities. *IEEE Transactions on Reliability 56(3)*, 534–539.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., & Dongarra, J. (1998). *MPI: The Complete Reference—The MPI Core*. Cambridge, MA: MIT Press.

Soro, I.W., Nourelfath, M., & Ait-Kadi, D. (2010). Performance evaluation of multi-state degraded systems with minimal repairs and imperfect preventive maintenance. *Reliability Engineering & System Safety 95(2)*, 65–69.

Stewart, W.J. (2009). *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton, NJ: Princeton University Press.

Wang, S., Ji, Y., Dong, W., & Yang, S. (2007). Design and RAMS analysis of a fault-tolerant computer control system. *Tsinghua Science & Technology 12*(Suppl. 1), 116–121.

Yu, H., Chu, C., Châtelet, Ė., & Yalaoui, F. (2007). Reliability optimization of a redundant system with failure dependencies. *Reliability Engineering & System Safety 92(12)*, 1627–1634.

Zhang, C.W., Zhang, T., Chen, N., & Jin, T. (2013). Reliability modeling and analysis for a novel design of modular converter system of wind turbines. *Reliability Engineering & System Safety 111*, 86–94.

Zhang, T., Long, W., & Sato, Y. (2003). Availability of systems with self-diagnostic components-applying Markov model to IEC 61508-6. *Reliability Engineering & System Safety 80(2)*, 133–141.

Po Ting Lin is a Professor in the Department of Mechanical Engineering, Research and Development Center for Microsystem Reliability and Center for Biomedical Technology at Chung Yuan Christian University.

Yu-Cheng Chou is a Professor in the Institute of Undersea Technology at National Sun Yat-sen University.

Yung Ting is a Professor in the Department of Mechanical Engineering at Chung Yuan Christian University.

Shian-Shing Shyu is a Scientist at the Institute of Nuclear Energy Research of the Atomic Energy Council in Taiwan.

Chang-Kuo Chen is a Scientist at the Institute of Nuclear Energy Research of the Atomic Energy Council in Taiwan.