Stochastic Bounds for Microprocessor Systems Availability

Ihab Sbeity[1], Mohamed Dbouk[1], and Brigitte Plateau[2]
[1]Faculty of Sciences, Lebanese University, Lebanon
[2]Laboratoire Informatique de Grenoble, France

Abstract: *The computing of stochastic bounds has become an efficient technique to obtain performance predictions for computer systems by the mean of Markovian models. However, the quality of these bounds may be affected by several properties related not only to how to use the technique but also to the Markovian model itself. On the other hand, multiprocessor systems have become an efficient and widely used infrastructure to run several critical-life applications. In this paper, we describe how to calculate a stochastic bound for the multiprocessor system availability. We focus on the irreducibility of the model to show how it might influence the quality of bounds.*

Keywords: *Stochastic bounds, multiprocessor system, availability, irreducibility, and Markov chain.*

Received January 15, 2009; accepted August 20, 2009

1. Introduction

Clusters, grids and multiprocessor systems have become popular computing platforms for computationally intensive applications with diverse computing needs [1]. The evaluation of such systems in order to tune an incorporated Quality of Service (QoS) control mechanism raises several problems that must be solved. One of the issues is to build and analyze stochastic models for the availability of the system which incorporates task deadlines, processors or network failures, overloading, and so on.

In the context of Markov theory, exponential probability distribution functions are used. Such distribution functions facilitate the evaluation process with Markovian resolution techniques used to obtain exact predictions of the model. However, in most practical cases, a limitation due to the model state space size explosion occurs. To deal with this limitation, an issue consists in calculating a bound of the system availability. In fact, for such systems, we might be interested in knowing whether the system availability is larger than what is necessary to execute an application and not necessary its exact value.

The stochastic comparison of random variables has been introduced as an efficient technique in different applied probability domains [10]. In the case of Markovian analysis, an algorithm to compute stochastic bound has been developed by Fourneau *et al.* [5, 6]. Consider a Markov model with a transition matrix denoted P. When the exact solution of the model is too complex to be achieved, for reasons related principally to the size of P (resolution of $X_P P = X_P$, X_P is a probability vector), the algorithm proposed by Fourneau *et al.* allows to construct another Markov chain (with a transition matrix Q). The new matrix, i.e., Q (called bounding matrix), permits to obtain bounds for the stationary measures of P. Moreover, the new chain is monotone according to the strong stochastic order (st) of Stoyan [10] (st-monotone), and can be strongly aggregated (the matrix is lumpable [2]) according to a user-defined partition of the state space. The aggregated matrix (Qagg) permits to analyze the chain at a reduced cost. In fact, the state space is then smaller and equal to the partition size. This procedure has been implemented in an algorithm called LIMSUB [6]. A variety of LIMSUB has been also implemented for high-level Markovian formalism such as Stochastic Automata Network (SAN) [7] where the chain is represented as a formula with tensor products of rather small matrices.

However, a serious problem that is related to this procedure is to choose a partition that gives a good quality bound. The construction of the matrix Q is based on a transformation of the initial matrix P in order to ensure monotonicity and lumpability conditions. The bounds quality is then strongly related to the "quasi-monotonicity" and the "quasi-lumpability" of the initial matrix reordered according to the partition. In [4], authors have studied the impact of monotonicity on the bounds quality. The lumpability is an important property because it permits the analysis of large Markov chains with a reduced state space in a very small time. However, the bounds quality depends on the type of states we aggregate together (states of heavy or lightly probability). On the other hand, irreducibility of Markov chain is also an important property that facilitates its stationary analysis. In [5, 6], authors present the conditions for the bounding matrix Q to be irreducible. These

conditions are related to the choice of the partition of the state space.

In this paper, we show that irreducibility is an important factor for the choice of the partition and that a bad choice may lead to a poor quality of bounds. Our observation is illustrated by an availability model of a multiprocessors system, for which we calculate a bound for the probability of failure. This work permits to identify the cases where the computing of stochastic bound would be efficient in multiple domains of computer science.

The paper organization is as follows. In section 2, we recall the initial concepts of the algorithm LIMSUB that calculates stochastic bounds. In particular, we present the impact of monotonicity and lumpability on the bounds quality. We also show the conditions necessary to ensure the irreducibility of the bounding matrix. These conditions form the basic point of our observation in order to show the impact of irreducibility on the bounds quality. In section 3, this impact is illustrated based on the multiprocessors model that is described. In section 4, we calculate bounds for the model focusing on the influence of irreducibility requested to analyze the bounding matrix. Section 5 concludes the paper.

2. Stochastic Bounds

2.1. Algorithm LIMSUB

In this section we consider only Discrete Time Markov Chain (DTMC). Continuous Time Markov Chain (CTMC) can be introduced similarly after an *uniformization* of the transition matrix. Also, we are interested only by the stationary analysis of DTMC. Computing stochastic bound for the stationary distribution of a DTMC has been developed and implemented by Fourneau *et al.* in an algorithm called LIMSUB [6]. Consider an irreducible DTMC with a transition matrix P of size K, and denote the stationary distribution of the chain by X_P ($X_P P = X_P$). Given a partition of the state space *Par* of size M ($M \leq K$), LIMSUB computes a bounding matrix Q of P of size K, that is monotone and lumpable according to the partition *Par*. We denote the element of line "i" and the column "j" of a matrix P by $P_{[i,j]}$.

The algorithm LIMSUB is composed of two principal phases: the first phase consists in ensuring the monotonicity of the bounding matrix, and the second phase deals with lumpability. Thus, without lack of generality, we present in a first step, the construction of the bounding matrix Q that is only monotone. In a second step, we describe the construction of the bounding monotone matrix that is lumpable. We denote the bounding matrix which is not lumpable by R and the lumpable matrix by Q (R is a special case of Q).

In [5, 6], based on the strong stochastic order of Stoyan [10], the authors define the relation between R, respectively Q, and P by equation 1.

$$\sum_{k=j}^{K} P_{[i,k]} \leq \sum_{k=j}^{K} R_{[i,k]} \forall\, i,\ j = 1,...,\ K$$
$$\sum_{k=j}^{K} R_{[i,k]} \leq \sum_{k=j}^{K} R_{[i+1,k]} \forall\, i,\ j = 1,...,\ K$$

The first inequality implies that the matrix R is a super bound of P according to the strong stochastic order ($P <_{st} R$). The second inequality means that the matrix R is monotonic. Denoting X_R the stationary distribution vector of R, and if X_P and X_R exist, based on equation 1 and according to the fundamental theorem of Stoyan then $X_P <_{st} X_R$, meaning that:

$$\sum_{j=k}^{K} X_P(j) \leq \sum_{j=k}^{K} X_R(j) \qquad \forall\ k = 1,2,...,\ K$$

where $X_P(j)$, respectively $X_R(j)$, represents the probability of state "j" of the Markov chain represented by P, respectively R, (as shown in [6] for further details).

Inequalities of equation 1 are the basic starting point of the algorithm LIMSUB to construct the matrix R. The idea consists in replacing these inequalities by a set of "limiting" equalities, leading to the algorithm. The equalities are shown in equation 3:

- $\sum_{k=j}^{K} R_{[1,k]} = \sum_{k=j}^{K} P_{[1,k]} \qquad \forall\ j = 1,2,...,\ K$
- $\sum_{k=j}^{K} R_{[i+1,k]} = \max\left(\sum_{k=j}^{K} R_{[i,k]}, \sum_{k=j}^{K} P_{[i+1,k]}\right)$ $\forall\ i, j = 1,2,...,\ K$

To illustrate these procedures, let us consider the matrices P and R of example 1.

$$P = \begin{bmatrix} 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.5 & 0.1 \\ 0.0 & 0.3 & 0.5 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.4 & 0.3 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.9 & 0.0 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.4 & 0.2 \\ 0.0 & 0.3 & 0.1 & 0.4 & 0.2 \\ 0.0 & 0.3 & 0.1 & 0.4 & 0.2 \\ 0.0 & 0.1 & 0.0 & 0.7 & 0.2 \end{bmatrix}$$

$$O = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.1 \\ 0.0 & 0.1 & 0.0 & 0.3 & 0.2 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \end{bmatrix}$$

In this example, the matrix R is a bound of the matrix P. Remark that elements of P and R respect the equalities given above. The matrix O represents errors or perturbations introduced in R relatively to P in order to ensure the monotonicity of R. In fact for each i and j, the value of $O_{[i,j]}$ is given by:

$$O_{[i,j]} = \sum_{k=j}^{K} R_{[i,k]} - P_{[i,k]}$$

For example, $O[2,4]$= ($R[2,5]$ + $R[2,4]$) - ($P[2,5]$ + $P[2,4]$) = (0.2 + 0.4) - (0.1 + 0.1) = 0.4, and $O[3,2]$ = (0.3 + 0.1 + 0.4 + 0.2) - (0.3 + 0.5 + 0.1 + 0.1) = 0.

We call this type of perturbations the monotonicity perturbations. In [4], authors observed that the bounds quality is strongly related to these perturbations. The authors focused on the calculation of a bound for the last state of the chain. They have remarked that, in particular, if the perturbations on the last column of the matrix are important, then the bounds quality would be poor. In our example, the probability vectors of the stationary distributions of R and P are given respectively by: X_R = (0.0293, 0.2337, 0.0800, 0.4570, 0.2000), and X_P = (0.0684, 0.2248, 0.3165, 0.3223, 0.0672). Remark that $X_P <_{st} X_R$, meaning that X_R is an upper bound of X_P according to Stoyan stochastic order.

- $0.0672 \leq 0.2000$.
- $0.0672 + 0.3223 \leq 0.2000 + 0.4570$.
- $0.0672 + 0.3223 + 0.3165 \leq 0.2000 + 0.4570 + 0.0800$.
- $0.0672 + 0.3223 + 0.3165 + 0.2248 \leq 0.2000 + 0.4570 + 0.0800 + 0.2337$.
- $0.0672 + 0.3223 + 0.3165 + 0.2248 + 0.0684 \leq 0.2000 + 0.4570 + 0.0800 + 0.2337 + 0.0293$.

Observe that the bounds quality of the last state, i.e., state 5, is poor relatively to its exact value ($0.0672 \leq 0.2000$). This is related to the large monotonicity perturbations on the last column of R.

On the other hand, the fundamental principle of LIMSUB is to compute a bounding matrix which may be aggregated according to a partition *Par* of the state space. The resolution of the aggregated matrix is performed on a smaller space whose size is equal to the partition size. We recall that Q will denote the bounding matrix that is monotonic and also lumpable. The notion of lumpability is recalled in definition 1.

Definition 1: let Q be an irreducible stochastic matrix of a finite Markov chain, and let Par = $\{p_1, p_2, \ldots p_M\}$ be a partition of the state space of the chain (M is the size of the partition). We say that the chain is strongly aggregable (ordinary lumpable) according to Par if and only if for each states i_1 and i_2 of the same set p_i of Par, we have that

$$\sum_{j \in p_k} Q_{[i_1,j]} = \sum_{j \in p_k} Q_{[i_1,j]}$$

for each set p_k of Par. Consider a Markov chain, having a vector of stationary probabilities, denoted X_Q, strongly aggregable according to a partition $Par = \{p_1, p_2, \ldots, p_M\}$. The transition matrix Q of the chain is then aggregable according to *Par*. Let Q_{agg} the aggregated matrix and X_{Qagg} the associated vector of stationary probabilities. A fundamental property of this aggregation is [2]:

$$\pi_{Q_{agg}}(p_i) = \sum_{j \in p_i} \pi_Q(j) \qquad \forall p_i \in Par$$

Given an initial matrix of P of size K, and a partition of the state space $Par = \{p_1, p_2, \ldots, p_M\}$, the algorithm LIMSUB calculates implicitly a matrix Q of size K that is lumpable according to *Par*, *st*-monotone and which bounds P. However, it is the matrix Q_{agg} of size M (the aggregation of Q) which is the output of the algorithm. Each element of Q_{agg} is a macro-state that corresponds to states belonging to a set p_i of *Par*.

Finally, suppose that matrix P is composed of M^2 blocks, each block *(i,j)* represents transitions from the set p_i to the set p_j. The algorithm LIMSUB operates with decomposition per blocks of the initial matrix and requires two steps: the first one ensures the monotonicity inside a block, and the second one modifies the first column of the block in order to satisfy the constraints of lumpability.

Recall the matrix P of example 1, and let us consider a partition $Par = \{p_1, p_2, p_3\}$ of the state space such that $p_1 = \{1,2\}$, $p_2 = \{3,4\}$ and $p_5 = \{5\}$. The matrix P and its bounding matrix Q which is lumpable according to *Par* are given in Example 2.

$$P = \left[\begin{array}{cc|ccc} 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.5 & 0.1 \\ \hline 0.0 & 0.3 & 0.5 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.4 & 0.3 & 0.0 \\ \hline 0.0 & 0.1 & 0.0 & 0.9 & 0.0 \end{array}\right]$$

$$Q = \left[\begin{array}{cc|ccc} 0.1 & 0.2 & 0.2 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.4 & 0.2 \\ \hline 0.0 & 0.3 & 0.1 & 0.4 & 0.2 \\ 0.0 & 0.3 & 0.1 & 0.4 & 0.2 \\ \hline 0.0 & 0.1 & 0.0 & 0.7 & 0.2 \end{array}\right]$$

$$Q_{agg} = \left[\begin{array}{c|c|c} 0.3 & 0.5 & 0.2 \\ \hline 0.3 & 0.5 & 0.2 \\ \hline 0.1 & 0.7 & 0.2 \end{array}\right] \qquad A = \left[\begin{array}{ccccc} 0.0 & 0.0 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}\right]$$

In this example Q_{agg} represents the aggregated matrix of Q, The matrix A represents the perturbations introduced in order to ensure the aggregability of matrix Q. These perturbations are given, for each i and j, by:

$$A_{[i,j]} = \sum_{k=j}^{K} Q_{[i,k]} - R_{[i,k]}$$

Thus, the value of $A_{[1,3]}$ = (0.2+ 0.3+ 0.2) - (0.1 + 0.3 + 0.2) = 0.1. Probability vectors of Q and Q_{agg} are respectively given by X_Q = (0.0260, 0.2340, 0.0826, 0.4574, 0.2000), and X_{Qagg} = (0.2600, 0.5400, 0.2000). It is obvious to remark that $X_P <_{st} X_Q$. On the other hand, remark that $X_R <_{st} X_Q$, meaning that the quality of the bounds calculated with R is better than those calculated using Q. This is effectively the consequence of lumpability perturbations seen above. However, recall that the interest of using Q is that calculation is done with a smaller matrix (Q_{agg}) in a reduced time.

0.41 in

2.2. Irreducibility of Bounding Matrix

In the previous subsection, we have recalled the algorithm that computes stochastic bounds, and we have shown the impact of monotonicity and lumpability on the bounds quality. In this subsection, we present the impact of another important property on the bound quality. In fact, the irreducibility of the bounding matrix is necessary in order to resolve the Markov chain presented by this matrix. Nevertheless, in some cases, we show how insuring this property can have a negative influence on the quality of bounds computed by LIMSUB. This irreducibility property requires a certain structure of the partition that may lead to non-significant bounds. We will illustrate this in our multiprocessor model in section 3.

In [6], necessary and sufficient conditions have been defined in order to the bounding matrix to be irreducible. These conditions are recalled in theorem 1.

Theorem 1: let P an irreducible stochastic matrix, Par = {p1, p2,..., p_M} a partition of the state space of P, and Q the lumpable bounding matrix of P. Then, the matrix Q is irreducible if and only if $P_{[1,1]} \neq 0$ and for any i, there exists in P a transition to a state j such that $i \in p_x$ and $j \in p_y$ and $y < x$.

The proof of this theorem can be found in [6]. Based on this theorem, we will give a corollary that will be the center point to show the impact of irreducibility on the bounds quality.

In some models, we may be interested in the calculation of a direct bound for the probability of some set of states (i.e., states of a global system failure in an availability model). We should then place these states in the last set according to the partition order, i.e., set p_M. Remark that in example 2, a direct bound can be found for the probability of state 5 of *P* using the aggregated matrix Q_{agg}. In fact $X_P(5) < X_{Qagg}(p_3) = 0.2000$.

Denote the set of states mentioned above by *J*. The set *J* may be, by choice, decomposed into multiple subsets $J_1, .., J_t$ $(t > 0)$. In order to calculate a direct bound for the probability of *J*, we define the sets of the partition *Par* such that: $p_M = J_t, \ldots, p_{M-t+1} = J_1$, by placing the subsets of *J* at the end according to the partition order. Nevertheless, the structure of the partition should respect some conditions in order to ensure the irreducibility of the bounding matrix.

Corollary 1: let P be an irreducible stochastic matrix, Par = {$p_1, p_2, \ldots, p_{M-1}, J_1, \ldots, J_t$} a partition of the state space of P, and R the bounding matrix of P calculated by LIMSUB that is aggregable according to Par. Let i a state of the Markov chain represented by P and succ(i) the set of successor states of i in the chain (for all j, $P[i,j] \neq 0$). If succ(i) is a subset of J, then a necessary condition in order to the matrix R be irreducible is that $i \in p_1$.

Based on theorem 1, the proof of the corollary is simple. Consider a state i of the Markov chain of P such that succ(i) is a subset of J and $i \in p_x$ $(x > 1)$. Then the set p_x contains a state i from which we cannot transit to a state j belonging to a set p_y such that $y < x$. According to theorem 1, the bounding matrix is not irreducible.

In the following, we present how the conditions described by the corollary may imply a bad bounding quality for some computer models.

3. Multiprocessor System Model

Multiprocessors systems have become an alternative frequently used to execute high performance applications. Very high availability rate is required in order to correctly execute life-critical applications. Some techniques, such that combinatorial analysis, Markov chains [8, 9, 11] and Monte-Carlo simulation [3] have been largely used in order to predict availability of multiprocessors system. However, the complexity of the model yields to difficulties for its exact resolution. An interesting alternative is to calculate a bound for the probability of system failure. In fact, for such systems, we might be interested in knowing whether the system availability is larger than the one necessary to execute applications and not necessary its exact value.

Consider a multiprocessors system with N cooperating processors. A processor alternates between two states: active where it functions correctly and failed when it is in a failed or a repair state after having been affected by a failure. We suppose that an active processor i may fail with an exponential rate α_i (then it passes to state failed). The exponential reparation rate of processor i is denoted β_i. On the other hand, each processor has a set of neighbor processors (all other processors of the system). When the processor i fails, its neighbors should enter in a state of reconfiguration of the system, in order to take in charge the job of the failed processor. When the processor is repaired, its neighbors should again enter the reconfiguration state in order to allow the processor to join back the system. We suppose that after a reconfiguration, each processor goes back to its active state. The time spent in the reconfiguration state is also exponential and it is equal to $1/\mu_i$ for each process i.

If we suppose that the processors rates are identical ($\alpha_i = \alpha$, $\beta_i = \beta$, $\mu i = \mu$, $\forall i$), then the model can be aggregated and its stationary solution can be easily calculated. However, we consider here different failure rates. In reality, the failure rate of a processor depends on its workload that is not the same for all processors of the system. So, we suppose that the failure rate of the processors are given such that $(\alpha_1 \neq \alpha_2 \neq \ldots \neq \alpha_i)$. Given this, the exact resolution of the model becomes difficult and depends on the size of the model that explodes with *N*.

On the other hand, the global failure of the system may be defined in two different forms. The first form (failure A) is the state where all processors have been failed. The second form (failure B) is defined by the fact that no processor is *active*. In this second form, we consider the state where some processors are failed and others are in reconfiguration as a global failure state. This assumption may appear pessimistic but it can be realistic [12].

In the remainder of this paper, we consider that a global state x is a triplet (n_a, n_r, n_f), where n_a, respectively n_r and n_f, represents the number of processors in *active*, respectively *reconfiguration* and *failure,* state ($n_a + n_r + n_f = N$). Failure A occurs when the system is in the global state $(0,0,N)$. Failure B occurs when the system is in one of the states of set $J = \{x / x = (0, r, f);\ r+f = N\}$. Based on this model we present in the next section the calculation of bounds for the probability of system failure. Recall that our objective is to show the impact of irreducibility on the stochastic bounds quality. By the way, we show also the interest of this calculation in the case it is possible.

4. Results Analysis

In this section, the availability analysis of the multiprocessors system is presented, by computing a bound for the probability of system failure. Several choices of the failure rate are possible. We suppose, for example, that α_i are given as $\alpha_i = \alpha + v.£$ where v is a random variable of the set $\{1,..,N\}$ and α and $£$ are two constants such that $\alpha > £$. Thus, we use a small deviation between the different failure rates.

Given the actual high reliability of actual processors and networks, a processor failure is often a scarce event. Following this, the parameters values are fixed to $\alpha = 0.001$, $£ = 0.0001$, $\beta = 1$ and $\mu = 10$. Thus, the mean failure average time is around $1/\alpha = 1000$ minutes, the mean reconfiguration time is 0.1 minute and the mean reparation time is 1 minute. In reality, the factor between the failure and reparation rates is of the order of 10^5 or 10^6. We use a factor of 10^3 for numerical precision requirements. Our objective being to present the quality of bounds calculated by LIMSUB, we have fixed the previous values of parameters. However, other values may be also used and produce similar results.

In section 3, we gave two different definition of the global failure (A and B). In section 4.1, we show bounds calculated for failure A. In section 4.2, based on failure B, we show how the computation of stochastic bounds might be inefficient for reasons related to irreducibility.

4.1. Failure A

The system global failure occurs when all processors are in their failure state. We use a partition of the state space given by $Par = \{p_0,...,p_{N-1}, J\}$ where the set p_i is composed of states where i processors have failed. The set J contains the only one global failure state. This partition permits to calculate a direct bound for the probability of system failure. The initial matrix ordered according to this partition is close to be monotone, and the most important perturbations are then those introduced to ensure lumpability.

For a model of N processors, the size of the initial matrix is of 3^N, the size of the bounding matrix is of $N+1$. The result of Table 1 shows that the obtained bounds are of very good quality examining the absolute deviation from the exact value.

Table 1. Probability of failure A.

N	State Space Size	Value of Bound	Exact Value
3	27	1.94 E-09	1.29 E-09
4	81	3.05 E-12	1.39 E-12
5	243	5.30 E-15	1.48 E-15
6	729	1.01 E-17	1.58 E-18
7	2187	2.09 E-20	1.67 E-21
8	6561	4.70 E-23	1.77 E-24
10	59049	2.95 E-28	1.95 E-30

On the other hand, notice that when the value of N increases, the quality of the bound degrades relatively to the exact value. This is related to the perturbations introduced to ensure the lumpability of the bounding matrix. In fact, the partition *Par* consists of certain blocks of the matrix to aggregate values α_i together (while taking the maximum which is equal to $max = \alpha + N£$). This consists in replacing values α_i by *max* implying a perturbation that becomes important with a largest value of N. To justify our observation, for $N = 10$, the set p_9 (9 processors have failed) is split in order to eliminate lumpability perturbations in the block-columns belonging to p_9. Thus, we have defined sets p_9^j containing successively states where the processor j is *not* in its failed state. Then the new bound value is 3.85 E-29 (instead of 2.95 E-28 without splitting p_9). While splitting sets p_i successively the bound quality is also improved, but slowly.

4.1. Failure B

In this case, the system failure occurs when there is no active processor. We will show that such definition of global failure imposes a partition of the state space that affects negatively the calculation of direct bound for the probability of failure. More precisely, we will show that failure B requires some structure of the partition in order to the bounding matrix to be irreducible. This required structure leads to a poor bound quality.

Consider a partition of the state space $Par = \{p_1,...,p_{M-1},J\}$. As we have shown, the algorithm LIMSUB consists in positioning the set of failure states (J) at the end of the partition in order to get a direct bound.

On the other hand, from any state $y = (a, 0, f)$ ($\notin J$) ($a+f=N$), the possible transitions are illustrated by the Figure 1.

- One processor among the active processors fails, with a rate α_i. Other active processors should enter the reconfiguration state. The new state is then of the form $(0,a-1,f+1)$ ($\in J$).
- One processor among the failed processors is repaired with a rate β. Other active processors should enter the reconfiguration state. The new state is then of the form $(0,a+1,f-1)$ ($\in J$).

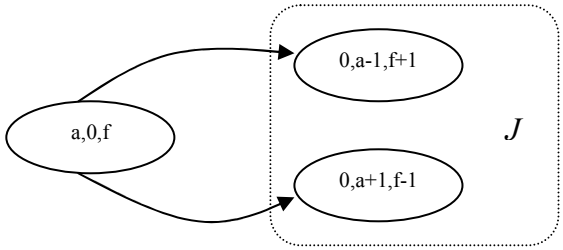

Figure 1. Pssible transitions to state *J*.

Given this, and according to the corollary 1, states $y = (a, 0, p)$ should be placed in the first set of the partition (p_1) in order to insure the irreducibility of the bounding matrix. Thus, respecting this condition, we can show that whatever the structure of the partition is, the bound quality is poor. To illustrate this, consider the initial matrix *P* of Figure 2, ordered according to the partition whose structure respects the previous condition. Looking at the columns of the last set of the partition, the greatest probabilities (the βs) belong to the first block (transitions from $(a, 0, p)$ to $(0,a+1,p-1)$). This will imply large perturbations introduced to the columns of the bounding matrix in order to be monotone. Values βs will propagate vertically in the last columns. Recall that in [4], authors have underlined that this perturbations imply a bad bound quality, which is also visible with our experimentations. Underline the fact that the origin of this poor quality is the constraint of irreducibility.

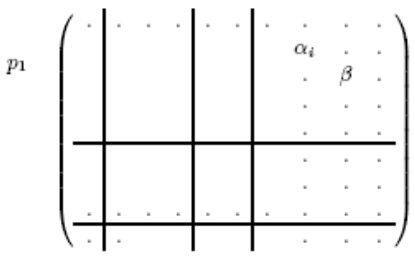

Figure 2. Initial matrix P.

5. Conclusion

The calculation of stochastic bounds is an efficient technique to cope with the difficulty of the exact resolution for very large Markov model. However, the bound quality depends on multiple factors. In this paper, the impact of irreducibility is shown, implying that the bound quality depends also on the system architecture and not only its behavior. Our perspectives are to propose a general heuristic giving the best partition allowing obtaining a good quality bound. We also plan to study other case study by taking, for example, other time distribution than exponential, i.e., phase type distributions.

References

[1] Avizienis A., Laprie C., and Randell B., "Dependability and its Threats: A Taxonomy," *Rapport LAAS No04386,* France, 2004.

[2] Buchholz P., "Exact and Ordinary Lumpability in Finite Markov Chains," *in Proceedings of Performance Evaluation*, pp. 211-226, 2002.

[3] Conway E. and Goyal A., "Monte Carlo Simulation of Computer System Availability and Reliability Models," *in Proceedings of FTCS-17*, pp. 33-38, 1987.

[4] Dayar T. and Pekergin N., "Stochastic Comparison, Re-Orderings, and Nearly Completely Decomposable Markov Chains," *in Proceedings of the International Conference on the Numerical Solution of Markov Chains*, pp. 228-246, 1999.

[5] Fourneau M. and Pekergin N., "An Algorithmic Approach to Stochastic Bounds," in Proceedings of *LNCS 2459, Performance Evaluation of Complex Systems: Techniques and Tools*, pp. 64-88, 2002.

[6] Fourneau M., Lecoz M., and Quessette F., "Algorithms for an Irreducible and Lumpable Strong Stochastic Bound," *in Proceedings of the International Conference on the Numerical Solution of Markov Chains*, USA, pp. 156-159, 2003.

[7] Fourneau M., Plateau B., Sbeity I., and Stewart J., *SANs and Lumpable Stochastic Bounds: Bounding Availability*, Imperial College Press, 2006.

[8] Geist R. and Trivedi S., "Ultra-Heigh Reliability Prediction for Fault-Tolerant Computer Systems," *Computer Journal of IEEE Transactions*, vol. 32, no. 12, pp. 84-89, 1986.

[9] Sbeity I. and Plateau B., "Structured Stochastic Modeling and Performance Analysis of a Multiprocessor System," *in Proceedings of Markov Anniversary Meeting*, USA, pp. 752-758, 2006.

[10] Stoyan D., *Comparison Methods for Queues and Other Stochastic Models*, Press Wiley, 1983.

[11] Trivedi S., "Probability and Statistics with Reliability, Queuing and Computer Science Applications," *Prentice Hall*, 1982.

[12] Trivedi S., Sathaye S., Ibe C., Howe C., and Aggarwal A., *Availability and Performance-Based Sizing of Multiprocessor Systems*, SAE International, 1996.

Ihab Sbeity received a Maîtrise in applied mathematics from the Lebanese University in 2002, a Master in computer science systems and communications from the Université Joseph Fourier, France, in 2003, and a PhD from Institut National Polytechnique de Grenoble, France in 2006. His PhD work is related to performance evaluation and system design. Currently, he occupies a full time position at the Lebanese University, Faculty of Sciences, Computer Sciences Department. His research interests include modeling and performance evaluation of parallel and distributed computer systems, numerical solution and simulation of large Markov models, UML modeling, and software performance engineering.

Mohamed Dbouk reievied Bachelor's Honor" in applied mathematics; computer science, Lebanese University, Faculty of Sciences (I)-Beirut, PhD from Paris Sud 11 University Orsay, France, 1997. His research interests include software engineering and information systems, performance modeling and optimization, geographic information systems and hypermedia, datawherehousing and data mining.

Brigitte Plateau received a Master in applied mathematics from the University of Paris 6 in 1976, a Thèse de Troisième Cycle in computer science from the University of Paris 11 in 1980, and a These d'Etat in computer science from the University of Paris 11 in 1984. She was Charg´e de Recherche at CNRS, France from 1981 to 1985, assistant professor at the Computer Science Department of the University of Maryland from 1985 to 1987. Currently, she holds a position of professor at the Engineering School ENSIMAG in Grenoble, France and is heading le LIG (Laboratoire d'informatique de Grenoble).