A SURVEY OF TECHNIQUES FOR ANALYZING MEMORY INTERFERENCE IN MULTIPROCESSOR COMPUTER SYSTEMS

Dileep P. Bhandarkar
Samuel H. Fuller

Carnegie-Mellon University

Pittsburgh, Pa.

April 1973

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C-0107) and is monitored by the Air Force Office of Scientific Research. This document has been approved for public release and sale; its distribution is unlimited.

ABSTRACT

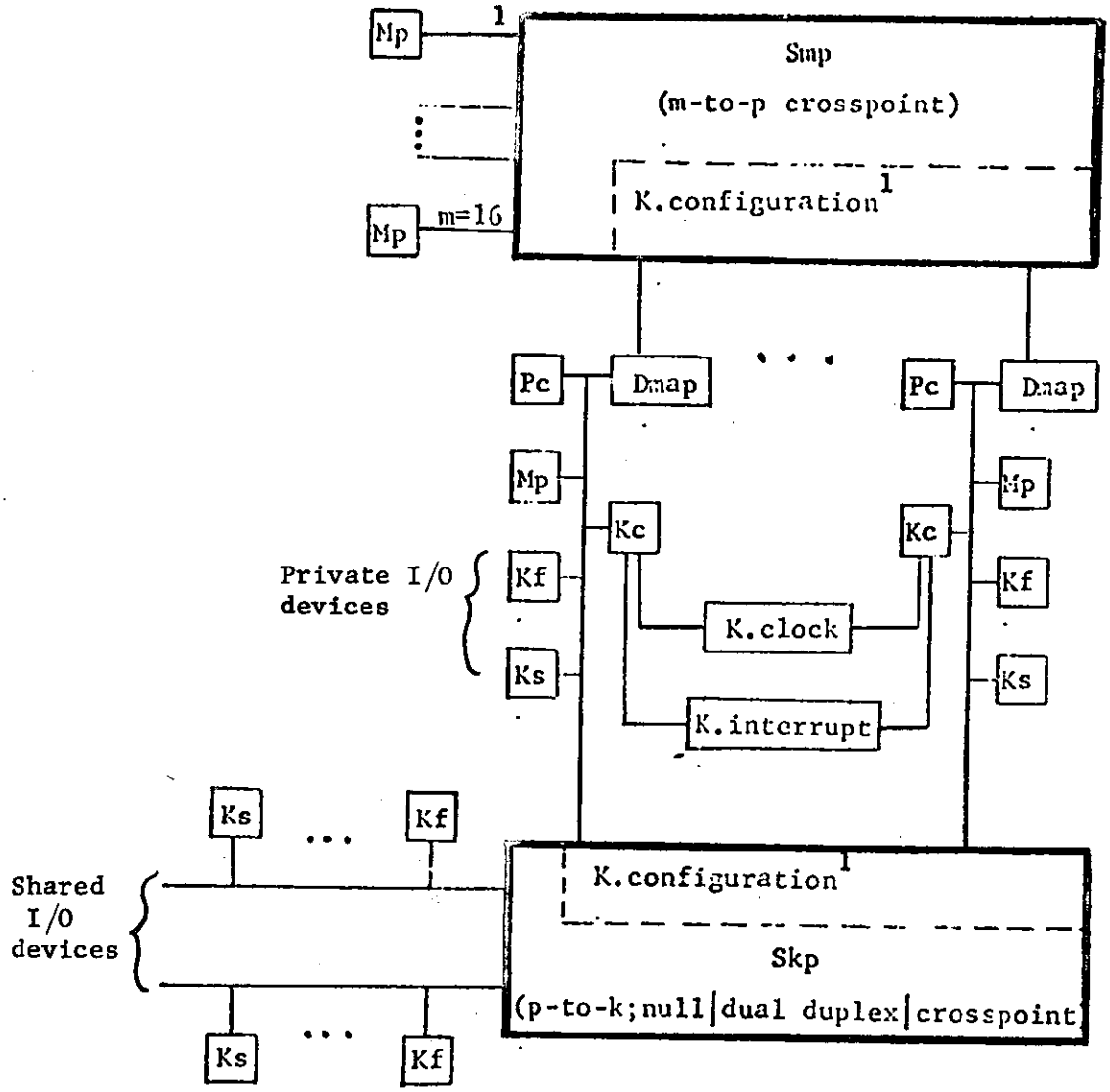
This paper surveys various analytic techniques for studying the extent of memory interference in a multiprocessor system with a crosspoint switch for processor-memory communication. Processor behavior is simplified to an ordered sequence of a memory request followed by a certain amount of processing time. The system is assumed to be bus bound; in other words, by the time the processor-memory bus completes servicing a processor's request the processor is ready to initiate another request and the memory module is ready to accept another request. The techniques discussed include discrete and continuous time Markov chain models, and some approximate analytic methods, viz. diffusion approximation and Strecker's approximation. The results are compared with a simulation model, in which the processing time has an exponential distribution and the memory cycle time is constant.

1.INTRODUCTION

Carnegie-Mellon University is currently in the process of constructing a multiprocessor computer system (C.mmp) that will have up to sixteen central processors (Pc's)[*] sharing the same physical address space [BellC71b; WulfW72] and concern has been expressed about the performance of such a system with this many active processors. Figure 1.1 illustrates the major components of a multiprocessor such as C.mmp. In addition to the processors, there is a set of memory modules that are able to operate independently; little would be gained if all the processors had to wait for service from a single memory module. Between the processors and the memory modules (Mp's) is a n by m switch. There are a number of ways of implementing the switch; Fig. 1.2(a) depicts a n by m crosspoint switch, and Fig. 1.2(b) illustrates the use of trunk lines; and combinations of these two basic schemes can yield many other other schemes. This report will examine the performance of a crosspoint switch since initial indications are that the crosspoint switch is the highest performance switching structure for a multiprocessor system and C.mmp is using such a switch. Other multiprocessors, although limited

---

[*]We use the PMS notation of Bell and Newell [1971] in this report to describe hardware organization.



where: Pc/central processor; Mp/primary memory; T/terminals;
Ks/slow device control (e.g., for Teletype);
Kf/fast device control (e.g., for disk);
Kc/control for clock, timer, interprocessor communication
Dmap/relocation registers for mapping Pc address into Mp address space.

[1] Both switches have static configuration control by manual and program control

Fig. 1.1 Proposed CMU multiminiprocessor computer/C.mmp.

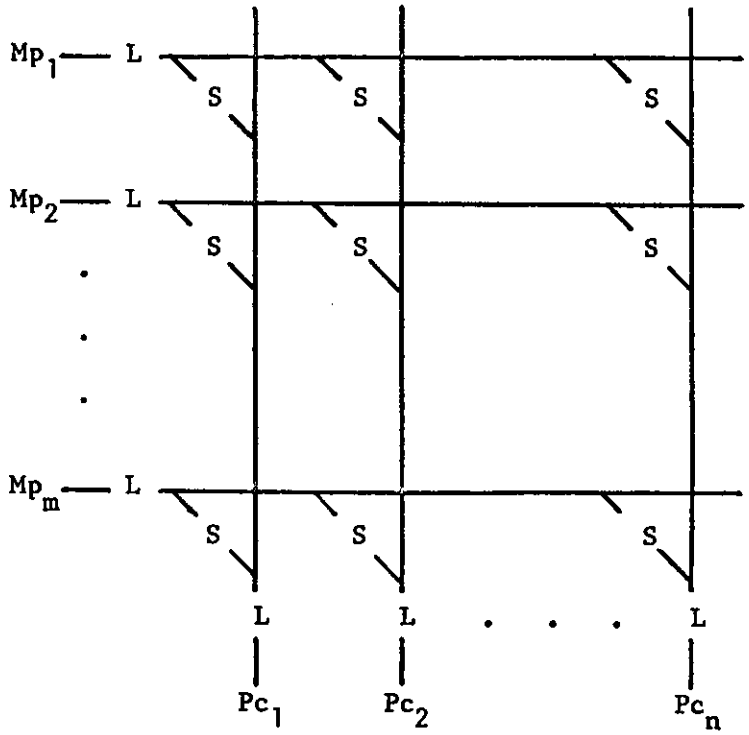


Figure 1.2a A n by m crosspoint switch

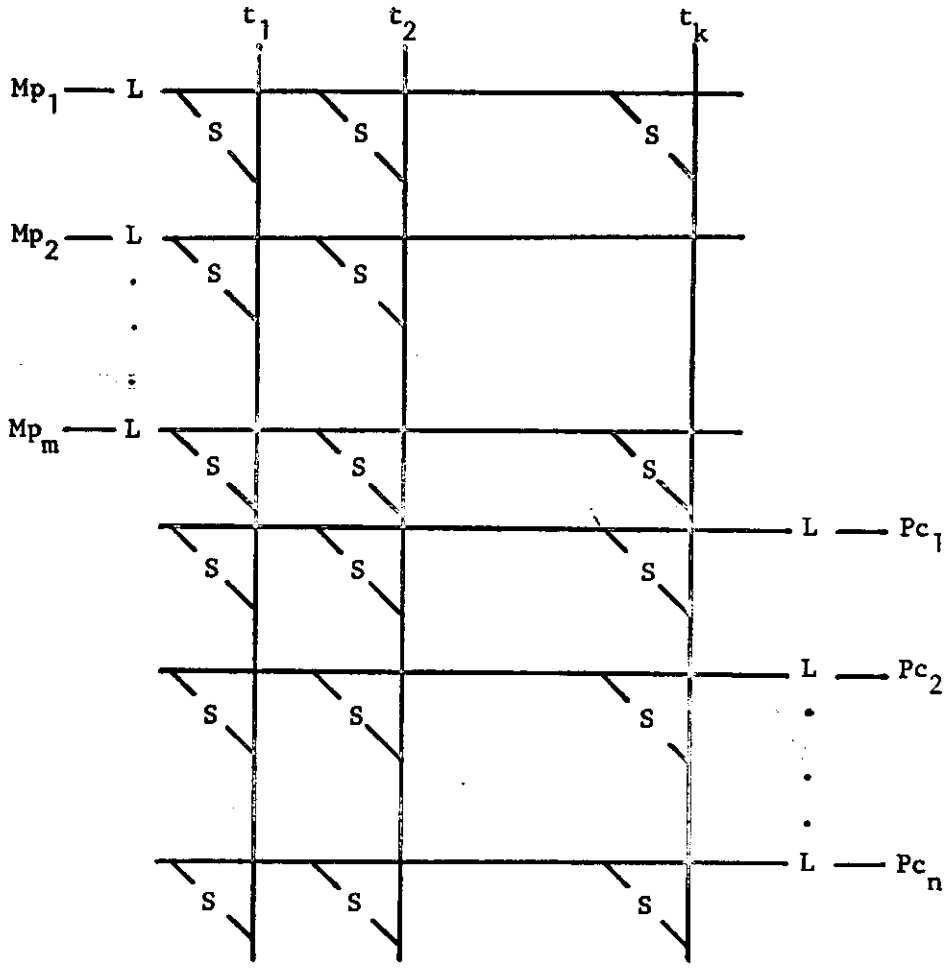


Figure 1.2b A k-trunk line switch

to a small number of Pc's, i.e. two to four also basically use a crosspoint switch, e.g. the Burroughs D825[AndeJ62] and Univac 1110. For further discussion of trunk lines, and a variety of other switching structures, the reader is referred to Bell and Newell [1971].

Mathematical models of computer systems can be developed at various levels of abstraction. A large number of models for time-sharing systems consider a job as a basic unit[cf. MckiJ69], and in many models of multiprogrammed computer systems the block of instructions between I/O operations is taken as a basic unit[cf. BuzeJ71; GaveD67]. However, in this study a much more detailed model is used to analyze interference as processors access individual words from the memory modules. Each processor's performance is measured by the number of memory accesses per unit time. In a multiprocessor system the performance of each Pc is not independent of the behavior of the other Pc's.

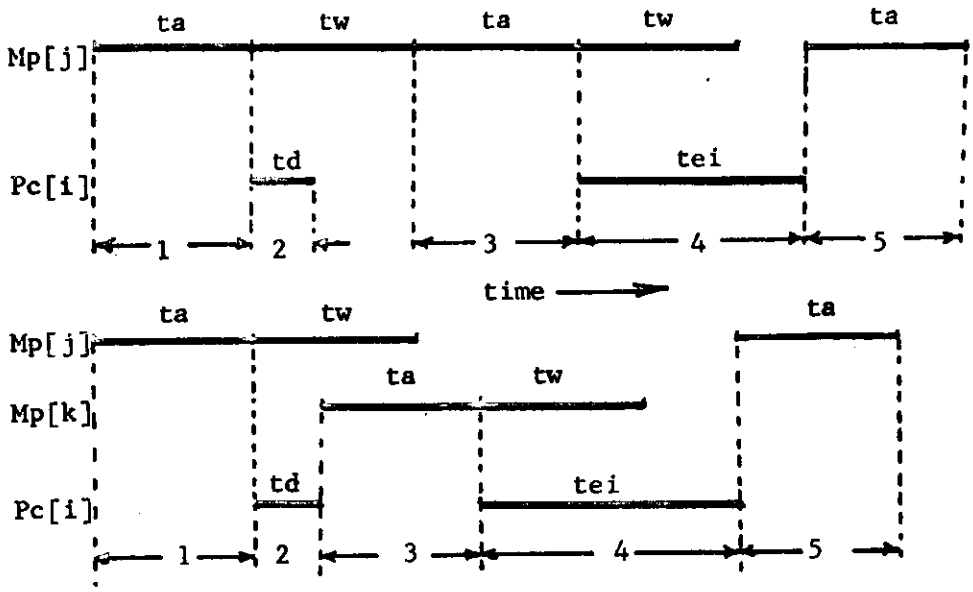
The following sections will discuss various techniques that can be used for analysing multiprocessor systems that are bus bound, i.e. systems in which the Pc is ready to initiate another request and the Mp module is ready to accept the next request exactly at the time the Pc-Mp bus recovers. The analysis is also valid for multiprocessor systems in which the effective processing time is equal to the memory rewrite time. The major contribution of this paper is a systematic

method for a discrete Markov chain model. Other techniques described include Strecker's approximation [StreW70], systems with exponentially distributed memory service time, and a diffusion approximation.

2. GENERAL MODELING ASSUMPTIONS

Due to the complexity of the problem, the exact detailed behavior of memory interference in a multiprocessor system is difficult to model. Some of the parameters that characterize the behavior of a Pc are:

(i) Instruction mix : Instructions can be characterized by their relative frequency. In general, processor behavior varies for different instructions. However, in this report differences in instructions are ignored. Processor behavior is modeled as a ordered sequence of a memory request followed by a certain amount of execution time. At this level of abstraction no distinction is made between the processing needed to decode an instruction and the processing corresponding to its execution. Thus, the processing time characterizing a Pc depicts only the aggregate behavior of the real Pc. Figure 2.1 depicts the actual and abstracted behaviors.



Legend:

1	instruction fetch	ta	memory access time
2	instruction decoding	tw	memory restore time
3	operand fetch	td	instruction decode time
4	instruction execution	tei	processor execution time
5	next instruction fetch		

Figure 2.1a: An example of the timing of a typical instruction.

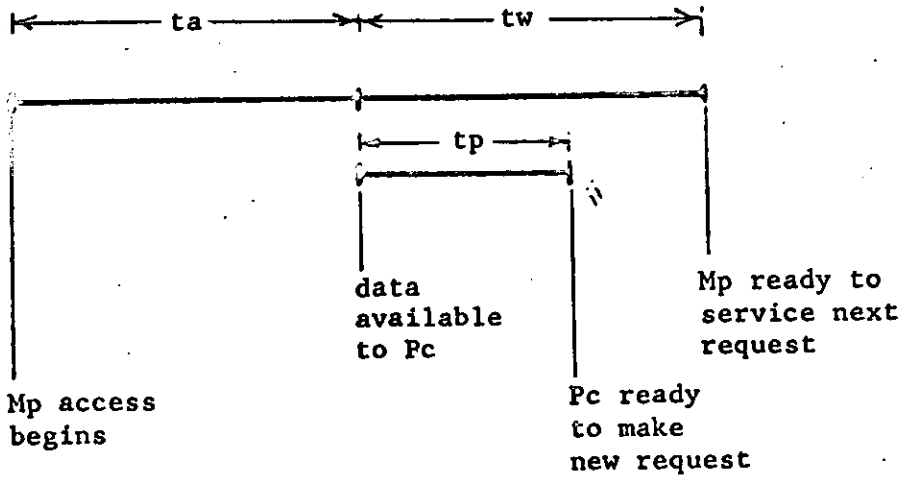


Figure 2.1b: Simplified processor behavior. Two such cycles model the instruction shown in Fig. 2.1a.

(ii) Average processing time: This is obtained from measurements similar to those used for determining the probability distribution.

(iii) Access pattern of a Pc: This is the trace of the pages or memory locations accessed by the Pc. In this study serial correlation between successive memory accesses will be ignored. Demand patterns will be modeled as sequences of Bernoulli trials. Memory accesses will be characterized by the memory unit to which it is addressed.

Primary memory behavior is a function of the fabrication technology i.e. core or semiconductor. Memory performance can be characterized by the access time (ta), rewrite time (tw), and cycle time (tc). Nominally, the cycle time is the sum of the other two. In this study, no distinction is made between read and write operations. The effect of interleaving within a Mp module is to make the access and cycle times variable.

3. CONTINUOUS TIME MARKOV CHAIN MODEL

Consider a multiprocessor system which consists of n Pc's and m Mp's connected by a single crosspoint switch. Let Pij denote the probability that the i-th processor requests service from the j-th memory unit. Thus, the demand pattern of each processor is equivalent to a sequence

of Bernoulli trials. A processor is said to be *queued* if it is waiting for or in the process of receiving memory service. A processor is said to be *active* if it is currently being serviced by a memory. Likewise, a memory is said to be *occupied or busy* if there is at least one processor queued for that memory unit.

If the system is bus-bound and the bus recovers at the same time that the memory is ready to service the next request, then the effective processing time (as seen by the memory) is equal to the memory rewrite time.

In our first model, we apply the classic simplifying assumption in queueing models: we model the service time, or cycle time, of the memory modules as exponentially distributed random variables [cf. WagnH69]. Clearly most memory systems do not have an exponentially distributed cycle time. However, techniques such as interleaving, cache memories, and the type of memory access(read, write, read-modify-write) suggest that this exponential assumption may be as good an approximation as the assumption that the memory cycle time is fixed, and not variable at all. Without further assumptions or approximations, we can use the results of Jackson [1963], and Gordon and Newell [1967] to find the performance of the multiprocessor system. This technique is also used by McCredie [1973] for multiprocessors with tp>tw.

Let the number of service centers be m. The states of the system are m-dimensional vectors with non-negative integer components, the j-th component representing the queue length at center j. If K=(k1,k2,....,km) is a state vector, then $S(K)=\sum_{i=1}^{m} ki$. Transition from one center to another is characterized by a routing probability Rij, i.e. the probability of going to center j on completion of service at center i. Jackson [1963] has obtained the equilibrium joint probability distribution of queue lengths for a broad class of queueing-theoretical models representing a network of service centers. Customer arrivals are modeled as a generalized Poisson process [cf. WagnH69], whose mean arrival rate varies almost arbitrarily with the total number of customers already in the system. Service completions at each center are also modeled as generalized Poisson processes, the mean service rate (u) at each center varying arbitrarily with the queue length there. Note that in Jackson's model all customers are identical. Muntz and Baskett [1972] have a more general queueing network model that allows different classes of customers to have different branching probabilities. Gordon and Newell [1967] have presented a solution technique for closed queueing systems, i.e. networks of queues in which the number of customers is constant.

For closed queueing systems, Jackson's formulae for obtaining the equilibrium state probabilities are listed below.

$$P(K) = w'(K)/T'(S(K))$$

where,

$$w'(K) = \prod_{j=1}^{m} \prod_{i=1}^{km} [e(j)/u] \quad \text{for } j \in [1,m]$$

$$\text{where } e(j) = \sum_{i=1}^{m} e(i)R(i,j) \quad j \in [1,m]$$

$$T'(K) = \sum w'(K) \quad \text{summed over K with } S(K)=n$$

But, with Pc requests distributed uniformly and with the bus-bound situation or tp=tw, Jackson's model reduces to m servers with customers circulating with uniform routing probabilities i.e. Rij=Pij=1/m. Using the above formulae we get,

$$w(K) = (1/u)\uparrow n$$

$$T(K) = \binom{n+m-1}{m-1}(1/u)\uparrow n$$

$$P(K) = \left[\binom{n+m-1}{m-1}\right]^{-1} \quad \text{for all K such that } \sum_{i=1}^{m} ki=n$$

i.e. all the states of the system have equal probability. Physically, this indicates that states with greater congestion in the queues are as likely as evenly distributed queues. The probability that a particular Mp module is idle, Prob{Mp[i] is idle}, is the fraction of the total number of states that has ki=0.

In other words,

$$\text{Prob\{Mp[i] is idle\}} = \frac{\text{number of ways of assigning n Pc's to m-1 Mp's}}{\text{number of ways of assigning n Pc's to m Mp's}}$$

$$= \frac{\binom{n+m-2}{m-2}}{\binom{n+m-1}{m-1}}$$

$$\text{Prob\{Mp[i] is busy\}} = n/(n+m-1)$$

E[number of busy Mp's] = m*Prob{Mp[i] is busy}

= m*n/(m+n-1)

The above expression has a number of interesting properties: the expression is symmetric in m and n; it has a basic hyperbolic form, asymptotic to n as m gets large; and, if we let m=n the above expression becomes

n/(2-1/n)

and

E[number of busy Mp's] → n/2 for n>>1

The final observation has important implications. It states that as multiprocessor systems grow to include more and more Pc's, we are not faced with a law of diminishing returns: no matter how many Pc's are used, if we have the same number of memory modules, we can expect half the processors to be active.

4. A SIMPLE DISCRETE MARKOV CHAIN MODEL

For this analysis let us assume that all the Pc's are characterized by a single constant processing time tp. Also, all the

memory units are assumed to have the same cycle time tc and access time ta. Thus, the memory rewrite time is given by tw=tc-ta. If tp= tw then all memory units can be considered to be operating synchronously. Thus, during any memory cycle the number of active Pc's is equal to the number of busy Mp's. With tp=tw, the analysis is simpler than with tp<tw and tp>tw. Also, it is a boundary condition for the other two cases. Thus, tp=tw is an interesting case for a preliminary comparison of various modeling techniques, even when tp is not equal to tw in reality.

In this section, a simple Markov Chain Analysis is presented for the case in which the processors request every memory with equal likelihood. A multiprocessor system with n Pc's and m Mp's is likened to an occupancy problem with n balls and m urns. Balls are randomly assigned to the m urns at the beginning of a memory cycle. At the end of the cycle one ball is removed from each urn. Thus if there are k non-empty urns during cycle s then k balls are available for assignment during the (s+1)-th cycle.

The state of the above mentioned process is defined by a m-tuple (k1,k2,...,km), where $\sum_{i=1}^{m} ki=n$ and $0 \le ki \le n$ for all i. The number of distinct states of the system is given by the combination, $\binom{n+m-1}{m-1}$ i.e. the number of ways in which n balls can be assigned to m bins [FellW66]. However, since all the processors behave identically, a

number of the distinct states are equivalent i.e. they have the same occupancy and have the same components. e.g. states (2,1,1), (1,2,1), (1,1,2) are equally likely. Thus, the reduced states are given by the different ways in which the number n can be partitioned into m parts. i.e. the unordered solutions to the equation $\sum_{i=1}^{m} Xi=n$ for $0 \le Xi \le n$ represent equivalent classes of equally likely states. The number of such partitions (for $n \le m$) is asymptotic to

$$\frac{1}{4\sqrt{3}} \exp[\pi(2n/3)^{0.5}] \qquad \text{[cf. BeckE64]}$$

Also,

$$F(x) = \frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}$$

$$= 1+\sum_{i=1}^{\infty} p(i)x^i$$

is an ordinary generating function of the sequence (p(0), p(1), ..., p(m)), where p(i) denotes the number of partitions of the integer i that have no part exceeding m [LiuC68].

Let the representative state Si denote the set of compositions of the number n that yield the same partition e.g. the compositions (2,1,1), (1,2,1) and (1,1,2) correspond to the partition of the number 4 which has two 1's and one 2. Further, let Si.j be the individual compositions of the partition typified by representative state Si and Si.1 be that composition which has its components arranged in monotone

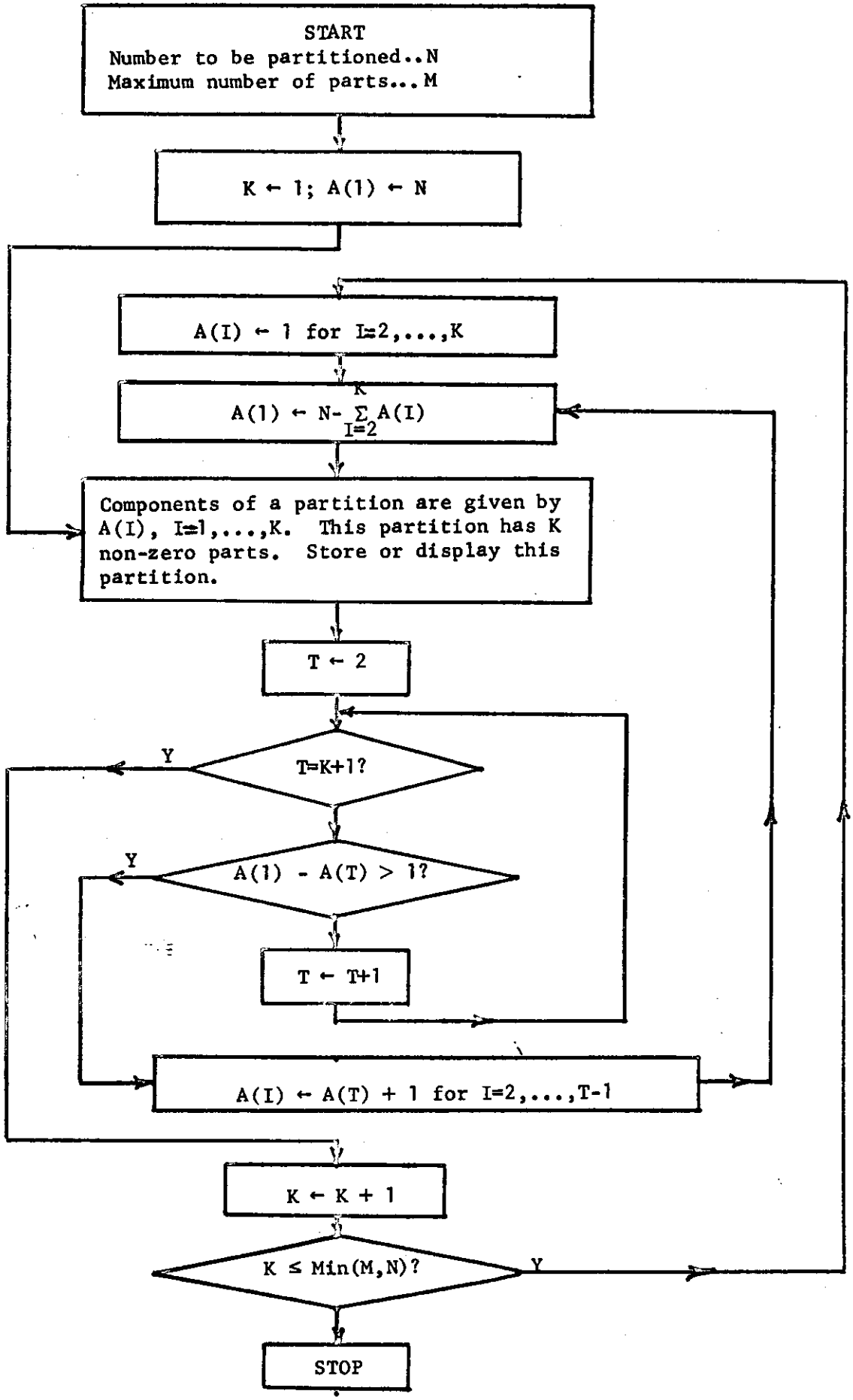


Figure 4.1 An algorithm for generating partitions

non-decreasing order, i.e. (2,1,1) for the above example. The algorithm shown in Fig. 4.1 generates all the partitions of n with the components in monotone non-increasing order.

Let Xij denote the probability of a transition from Sj to Si. Then, due to the symmetry of the problem,

$$Xij = \sum_{Si.k \in Si} \text{Prob}\{\text{Transition from Sj.1 to Si.k}\}$$

Let the m-tuple (k1,k2,....,km) denote the state of the Markov chain. If x is the number of non-zero elements in this vector then at the end of the memory cycle, x new processors have to be reassigned to memory modules. At the end of the current memory cycle the queue is characterized by the m-tuple (J1,J2,...,Jm), where

$$Ji = ki-1 \text{ if } ki>0$$

$$= 0 \text{ otherwise.}$$

A new state (L1,L2,...,Lm) is reachable from (k1,k2,...,km) if and only if $Li \geq Ji$ for $1 \leq i \leq m$. If the above condition is satisfied the probability of the state transition is given by

$$\binom{x}{d1} * \binom{x-d1}{d2} * \binom{x-d1-d2}{d3} * \cdots * \binom{x-\sum_{i=1}^{m-1} di}{dm}$$

where di= Li-Ji

i.e. $\dfrac{x!}{d1!\ d2!\ \ldots dm!} * (1/m)\uparrow x$

Note that since $\sum_{i=1}^{m} ki = \sum_{i=1}^{m} Li = n$, $\sum_{i=1}^{m} di = x$

Thus, we now have a formula for generating the transition probabilities. Due to the symmetry of the problem it suffices to generate only the transition probabilities for the representative class of states. All the different ways of obtaining the same partition are lumped together to form a reduced state.

To illustrate a computational method** for generating the transition probabilities consider an example of a 4 by 4 system. The number 4 can be partitioned in 5 different ways as listed below:

4 0 0 0

3 1 0 0

2 2 0 0

2 1 1 0

1 1 1 1

**The use of a tree to generate the transition probabilities was suggested by F. Baskett and D.Chewning of Stanford University.

These partitions represent 5 equivalent classes that characterize the state of the Markov Chain. Let us consider the state (2,2,0,0). At the end of a memory cycle, the resultant partial state is (1,1,0,0) with 2 free processors to be reassigned. Figure 4.2 shows the different ways in which these 2 Pc's can be assigned, one at a time, to reach a new partial representative state. After both Pc's are assigned a terminal state is reached. The number on the arrow indicates the number of ways of reaching the partial or terminal state that the arrow points to. Now the number of ways in which a final state can be reached from the initial state can be computed by traversing the tree, e.g. there are 2x1 ways of reaching (1,1,1,1) and (2x2 + 2x3) ways of reaching (2,1,1,0) from (2,2,0,0).

It is possible to construct a single tree with different pointers for different initial states. Figure 4.3 shows a complete tree for a 4x4 system. Initial states are circled. The entire transition matrix can be filled by traversing this tree. A convenient way† of traversing this tree is by using a stack which has depth equal to one more than the number of Pc's. At each level the stack contains a partial state and has a pointer to the initial representative state (if any) from which it is derived. The stack is initialized to contain the path that

---

†An alternative method for traversing the tree is described in Appendix I.

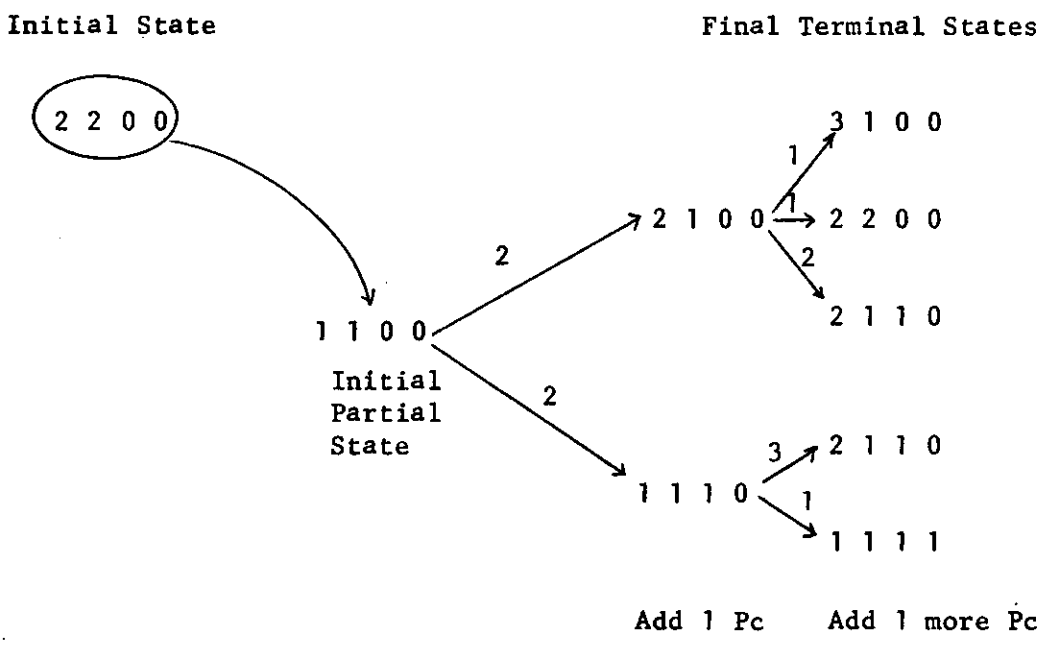


Figure 4.2 Next states accessible from initial state (2,2,0,0)

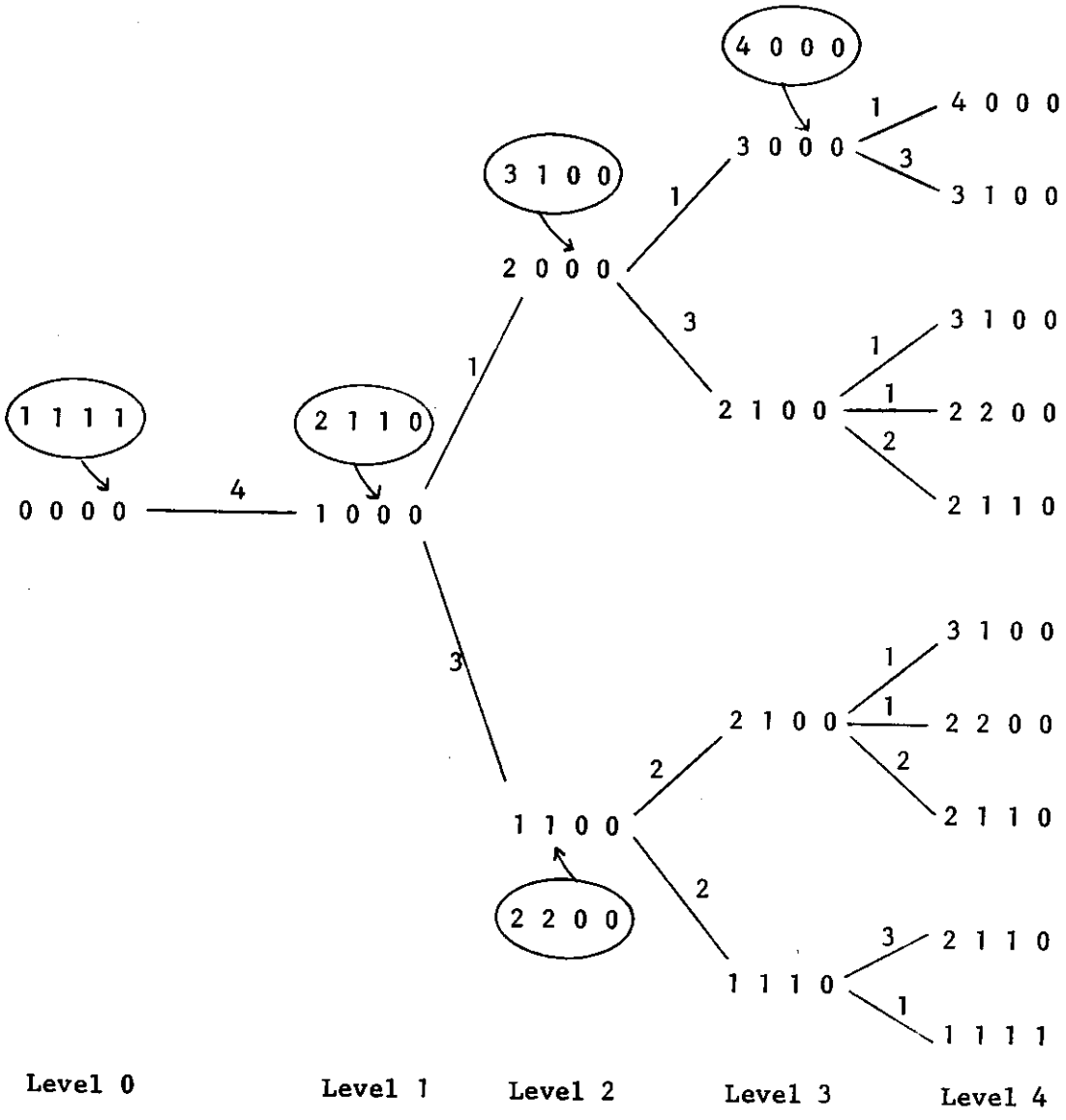


Figure 4.3 Enumeration tree for a 4 by 4 multiprocessor system

leads to the topmost final state. For this example the stack is initialized as shown in Fig. 4.4, and Fig. 4.5 shows an algorithm for using the tree to generate the transition matrix, shown in fig. 4.6.

The following theorem and lemma can be used to increase the efficiency of the program that generates the transition probabilities.

Theorem 1: There is a one-to-one correspondence between a representative state and a partial state that the representative state reduces to at the end of a cycle.

Proof: Let (k1,...,km) be a representative state. The partial state at the end of the cycle is given by

(J1,J2,...,Jm)

where Ji=ki-1 if ki>0

=0 otherwise

Since no two representative states are alike and $\sum_{i=1}^{m} ki=n$, it follows that the partial states are distinct.

Lemma A partial state at level L in the enumerative tree of Fig. 4.3 can correspond to a terminal state with exactly n-L occupied Mp's.

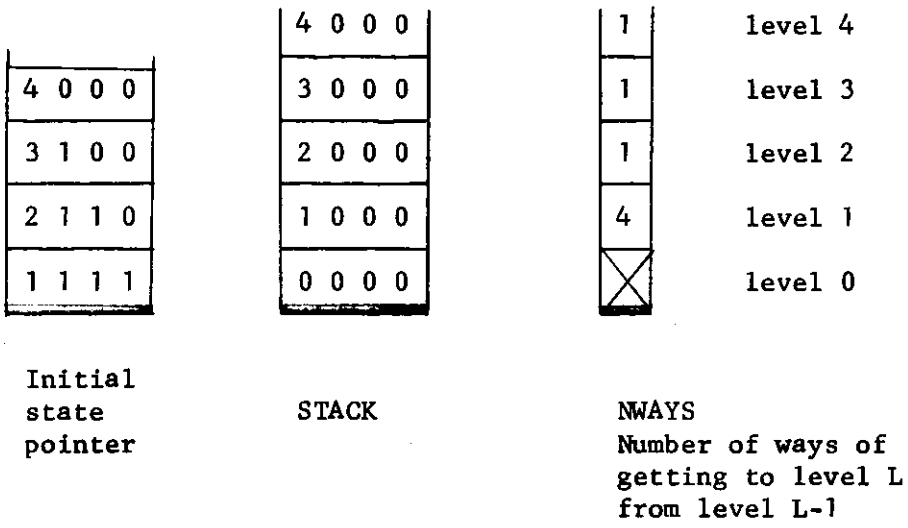


Figure 4.4 Initial contents of the stack for traversing the tree shown in Figure 2.3

Proof: Let J=(J1,J2,....,Jm) be a partial state in the tree depicted in Fig. 4.3. Furthermore, let the number of non-zero elements elements in the partial state be y and let $\sum_{i=1}^{m} Ji=n-x$. Since one Pc is always removed from a non-empty queue at the end of a cycle, J is a partial state that can be reduced from a valid representative state K=(k1,k2,...,km), if and only if

(i) The number of non-zero elements in K is x,and

(ii) x>y

Note that x and y are both less than or equal to min(m,n) and $\sum_{i=1}^{m} ki=n$. Then, if x>y, J has atleast x-y zeros. If x<y then there is no representative state K that corresponds to the partial state J. If x≥y, then the representative state is obtained by adding y 1's to the non-zero elements of J and replacing x-y zeros of J by 1. At level L, $\sum_{i=1}^{m} Ji=$ L. Therefore, x, the number of occupied Mp's in K, is equal to n-L.

Figure 4.7 shows the average number of busy Mp's when n=m. The curve has an almost constant slope of .586 for n>4. Figures 4.8 and 4.9 show the effect of adding a Pc and an Mp respectively on the average number of busy Mp's.

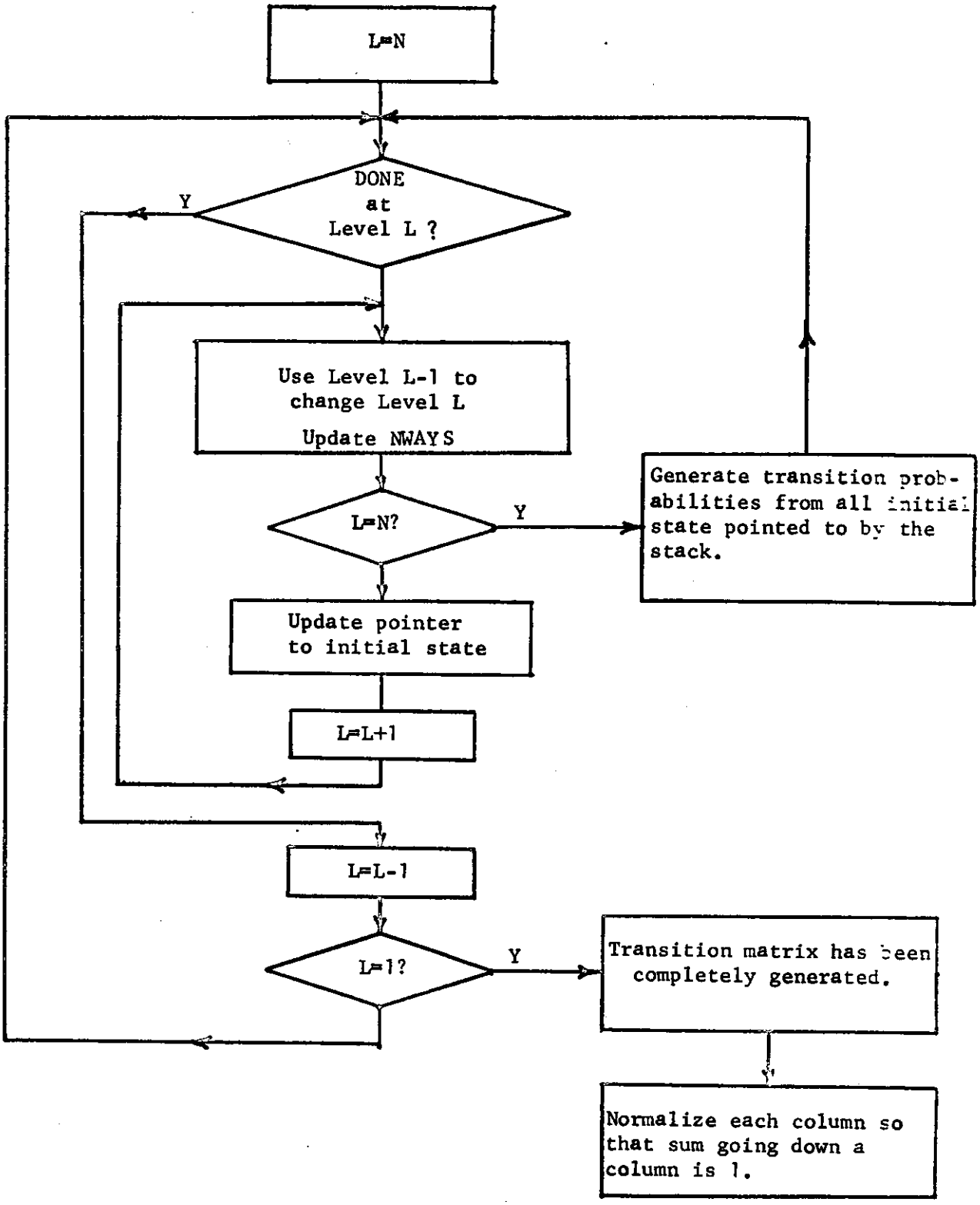


Figure 4.5 Algorithm for traversing the tree shown in Figure 4.3

	4 0 0 0	3 1 0 0	2 2 0 0	2 1 1 0	1 1 1 1
4 0 0 0	1	1	0	1	4
3 1 0 0	3	3+3	2	3+3+6	12+12+24
2 2 0 0	0	3	2	3+6	12+24
2 1 1 0	0	6	4+6	6+12+18	24+48+72
1 1 1 1	0	0	2	6	24

STEP 1: Xij is the number of ways of reaching i from j. (obtained from the tree of fig. 2.3)

STEP 2: Xij $\frac{Xij}{\sum_i Xij}$ (Note that $\sum_i Xij = m^x$, where x of the m components of j are non-zero)

Final equations to be solved simultaneously:

$$\begin{bmatrix} P_{4000} \\ P_{3100} \\ P_{2200} \\ P_{2100} \\ P_{1111} \end{bmatrix} \begin{bmatrix} 0.25 & 0.0625 & 0.000 & 0.015625 & 0.015265 \\ 0.75 & 0.3750 & 0.125 & 0.187500 & 0.187500 \\ 0.00 & 0.1875 & 0.125 & 0.140625 & 0.140625 \\ 0.00 & 0.3750 & 0.625 & 0.562500 & 0.562500 \\ 0.00 & 0.0000 & 0.125 & 0.093750 & 0.93750 \end{bmatrix} \begin{bmatrix} P_{4000} \\ P_{3100} \\ P_{2200} \\ P_{2100} \\ P_{1111} \end{bmatrix}$$

SUBJECT TO $P_{4000} + P_{3100} + P_{2200} + P_{2100} + P_{1111} = 1$

Figure 4.6 Steps in the generation of the transition matrix

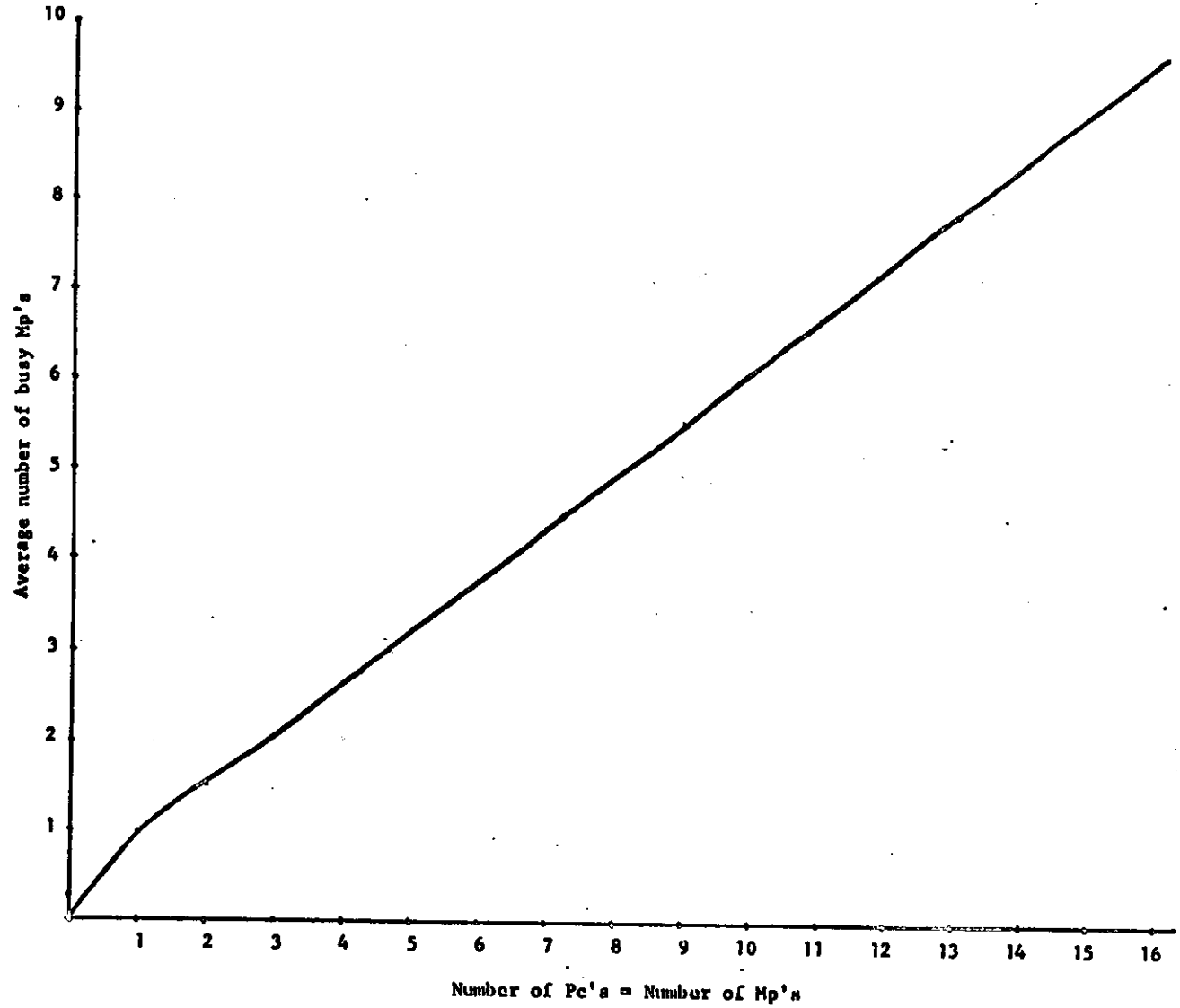


Figure 4.7: Multiprocessor Systems with n=m.

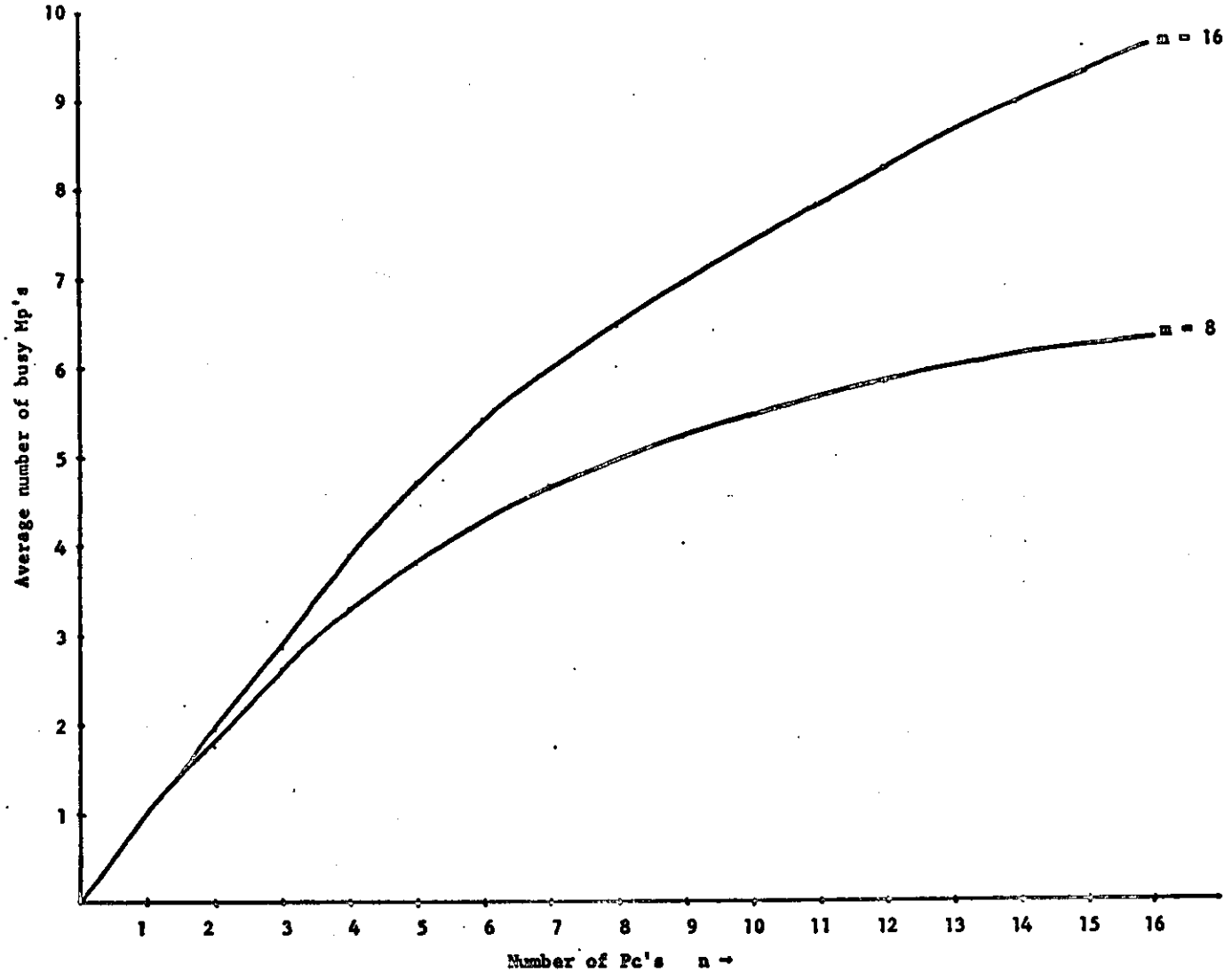


Figure 4.8: The effect of adding a Pc.

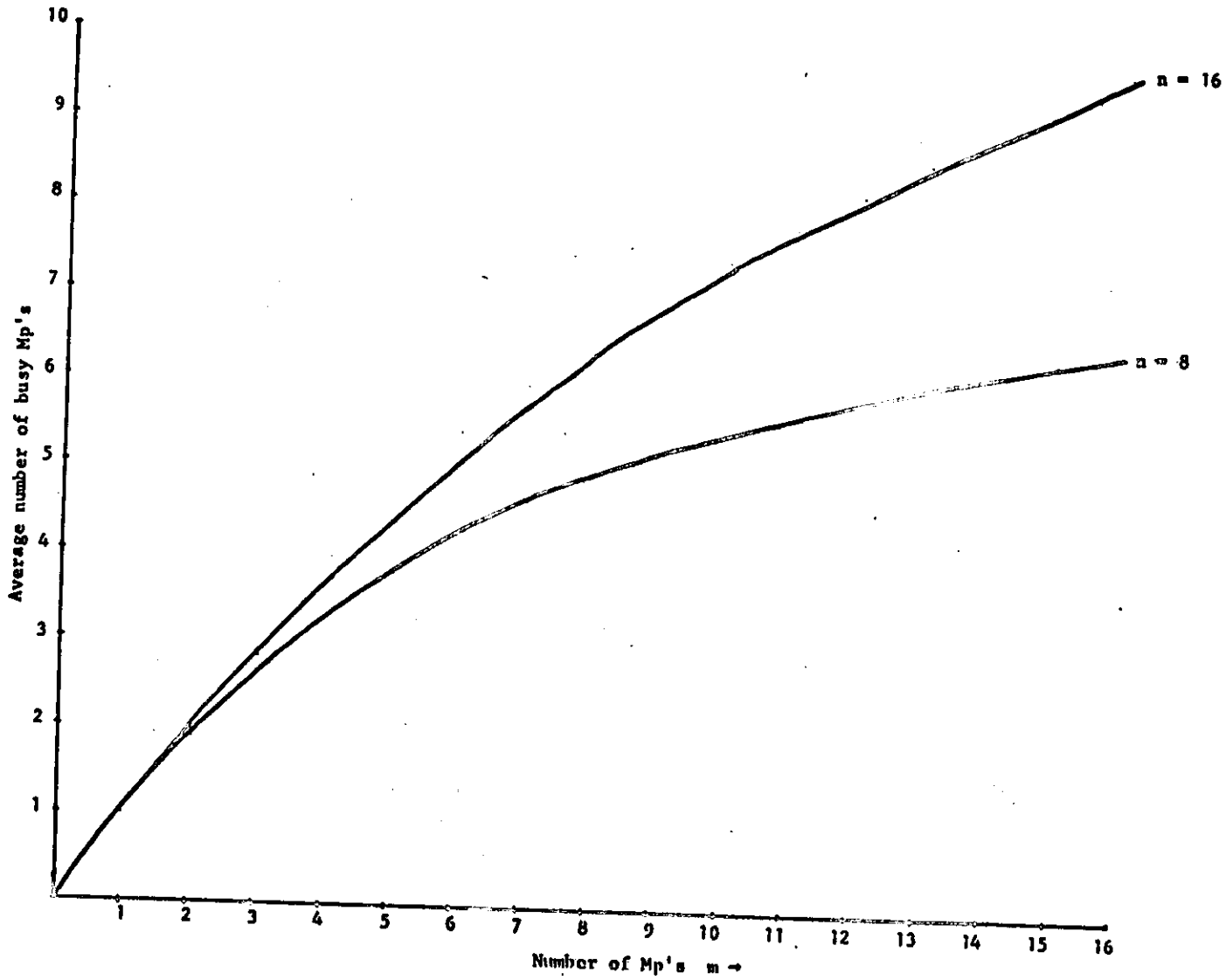


Figure 4.9: The effect of adding an Mp.

5. OTHER DISCRETE MARKOV CHAIN MODELS

5.1 Discrete Markov Model of Skinner and Asher

Skinner and Asher [1969] model the multiprocessor system as a discrete Markov chain. They assume a matrix of probabilities that express the likelihood that a given processor requests service from a given memory at the beginning of a memory cycle, provided the Pc is not queued. They also assume a matrix of probabilities that express the likelihood of the various outcomes that can arise when there are simultaneous requests to one memory by several processors. The state of the system is characterized by the processors queued for the different memory modules. A state transition matrix is formed from the access probabilities and the steady state probabilities of various states are determined by solving the state transition equations. The number of states of the system increases very steeply with an increase in the number of Pc's and Mp's. Closed form solutions are presented only for cases with up to 2 Pc's and n Mp's. The analysis in the previous section is similar to Skinner and Asher, but with uniformly random access patterns for all the Pc's, i.e. Pij=1/m for all i.

5.2 Strecker's Approximation

Strecker [1970] has an approximate closed form solution to the discrete Markov Chain model presented here. His approach is equivalent to removing the queued processors from all the memory modules at the end of a memory cycle and reassigning them. Thus the state of the system is considered independent of the state during the last cycle. If we use this assumption the distribution of Pc's queued for an Mp follows the binomial distribution:

$$\text{Prob}\{Y=r\} = \binom{n}{r} * (1/m)^r * (1-1/m)^{n-r}$$

where Y is a random variable equal to the number of Pc's queued for Mp[j] and Pij=1/m for all i and j.

Thus,

Prob{Mp[j] is busy} = 1- Prob{nobody is queued for Mp[j]}

$$= 1 - (1-1/m)\uparrow n$$

In other words, the occupancy of Mp[j] is 1-(1-1/m)↑n, and

$$\text{E[no. of occupied Mp's]} = \sum_{j=1}^{m} \{\text{Occupancy of Mp[j]}\}$$

$$= m*[1-(1-1/m)\uparrow n]$$

Table 1 shows a comparison of Strecker's results and the exact

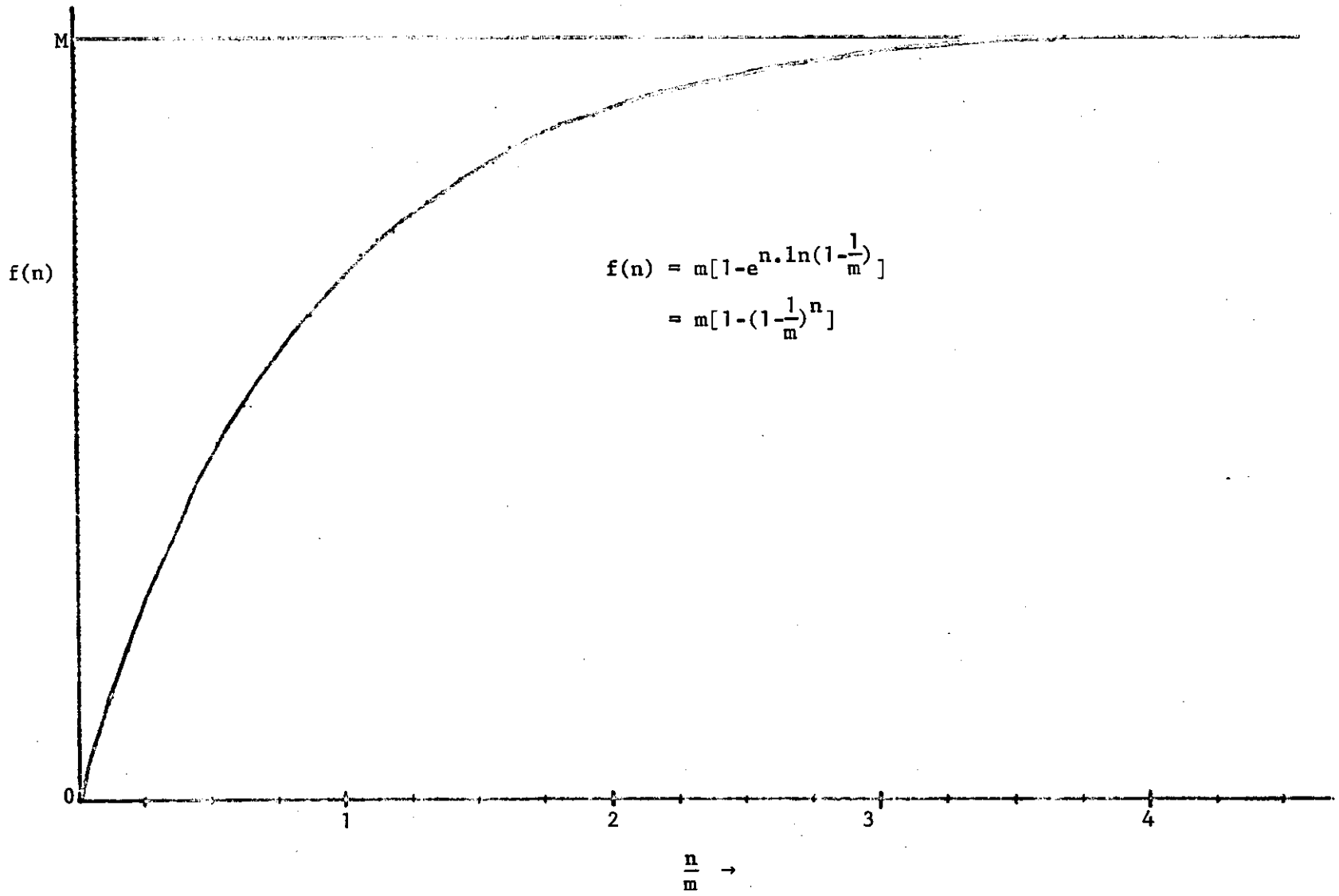


Figure 5.1 Strecker's formula for fixed m

Markov chain analysis. Note that Strecker's results are optimistic estimates of the unit execution rate. It is encouraging to note that such a simple expression is within 6 to 8% of the exact Markov Chain model for m/n>0.75. This is because his analysis assumes that all n Pc's always make a new request at the beginning of each memory cycle, whereas in the discrete Markov chain only those Pc's that receive service are allowed to make new requests. Moreover, note that the expression m*[1-(1-1/m)↑n] can be written in an exponential form as

$$m*\{1-\exp[n* ln\ (1-1/m)]\}$$

Figure 5.1 shows a plot of the above expression for fixed m; the relaxation time $[\ ln\ (1-1/m)]^{-1}$ approaches m as m gets large.

6. DIFFUSION APPROXIMATIONS

An approximation method that has been proposed for the solution of general queueing networks is the diffusion approximation [c.f. NeweG71; KobaH73]. A discrete-state process is approximated by a Wiener-Levy diffusion process with a continuous path. The key assumption in such an analysis is that incremental changes in the queue lengths are normally distributed. This leads to a characterization of the queueing network by a set of diffusion equations. The accuracy of the approximation

depends on three factors: (i) approximation of a discrete-state process by a time-continuous Markov process, (ii) choice of proper reflecting barriers, and (iii) discretization of the continuous density function for queue lengths. Surprisingly, for the simple discrete Markov Chain model of section 4, the diffusion approximation yields a result identical to that with exponential servers derived from Jackson's formulae. However, the main utility of the diffusion approximation in this context is that it can be used to analyze the effect of different coefficients of variation (ratio of standard deviation to the mean) for the service time distribution.

7. CONCLUDING REMARKS

Tables 1 and 2 compare the numerical results obtained from the different models described. Strecker's approximation gets better as m/n increases, whereas the continuous time and discrete Markov models get closer for larger n/m ratios. Table 3 shows some simulation results obtained with exponential distributions for the processing time, with mean equal to tw.

i.e. $\text{Prob}\{tp=x\} = \lambda \exp(-\lambda x)$ where $\lambda=1/tw=1/ta=1/E[tp]$

Note that the values in Table 3 lie between those predicted by Strecker and Jackson. Table 4 shows the characteristics of the parameters in the various models.

TABLE 1

Expected number of busy memories in one cycle
Number of Pc's = 1,2,...,8 (rows)
Number of Mp's = 1,2,...,8 (columns)

Discrete Markov Chain Model

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750
1.0000	1.6667	2.0476	2.2692	2.4095	2.5054	2.5748	2.6272
1.0000	1.7500	2.2701	2.6210	2.8630	3.0365	3.1657	3.2652
1.0000	1.8000	2.4102	2.8633	3.1996	3.4530	3.6482	3.8019
1.0000	1.8333	2.5059	3.0370	3.4533	3.7809	4.0415	4.2518
1.0000	1.8571	2.5751	3.1663	3.6486	4.0418	4.3636	4.6292
1.0000	1.8750	2.6274	3.2657	3.8024	4.2521	4.6294	4.9471

Strecker's Approximation

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750
1.0000	1.7500	2.1111	2.3125	2.4400	2.5278	2.5918	2.6406
1.0000	1.8750	2.4074	2.7344	2.9520	3.1065	3.2216	3.3105
1.0000	1.9375	2.6049	3.0508	3.3616	3.5887	3.7613	3.8967
1.0000	1.9687	2.7366	3.2881	3.6893	3.9906	4.2240	4.4096
1.0000	1.9844	2.8244	3.4661	3.9514	4.3255	4.6206	4.8584
1.0000	1.9922	2.8829	3.5995	4.1611	4.6046	4.9605	5.2511

Percentage Error

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	4.9979	3.1012	1.9082	1.2658	0.8941	0.6602	0.5100
0.0000	7.1429	6.0482	4.3266	3.1086	2.3053	1.7658	1.3874
0.0000	7.6389	8.0782	6.5484	5.0631	3.9299	3.1002	2.4935
0.0000	7.3856	9.2063	8.2680	6.8340	5.5463	4.5157	3.7114
0.0000	6.8548	9.6812	9.4685	8.2991	7.0191	5.8896	4.9512
0.0000	6.2507	9.7244	10.2214	9.4335	8.2900	7.1521	6.1450

TABLE 2

Expected number of busy memories in one cycle
Number of Pc's = 1,2,...,8 (rows)
Number of Mp's = 1,2,...,8 (columns)

Discrete Markov Chain Model

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.5000	1.6667	1.7500	1.8000	1.8333	1.8571	1.8750
1.0000	1.6667	2.0476	2.2692	2.4095	2.5054	2.5748	2.6272
1.0000	1.7500	2.2701	2.6210	2.8630	3.0365	3.1657	3.2652
1.0000	1.8000	2.4102	2.8633	3.1996	3.4530	3.6482	3.8019
1.0000	1.8333	2.5059	3.0370	3.4533	3.7809	4.0415	4.2518
1.0000	1.8571	2.5751	3.1663	3.6486	4.0418	4.3636	4.6292
1.0000	1.8750	2.6274	3.2657	3.8024	4.2521	4.6294	4.9471

Continuous Time Markov Chain Model

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.3333	1.5000	1.6000	1.6667	1.7143	1.7500	1.7778
1.0000	1.5000	1.8000	2.0000	2.1429	2.2500	2.3333	2.4000
1.0000	1.6000	2.0000	2.2857	2.5000	2.6667	2.8000	2.9091
1.0000	1.6667	2.1429	2.5000	2.7778	3.0000	3.1818	3.3333
1.0000	1.7143	2.2500	2.6667	3.0000	3.2727	3.5000	3.6923
1.0000	1.7500	2.3333	2.8000	3.1818	3.5000	3.7692	4.0000
1.0000	1.7778	2.4000	2.9091	3.3333	3.6923	4.0000	4.2667

Percentage Difference

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	11.1133	10.0018	8.5714	7.4056	6.4910	5.7671	5.1840
0.0000	10.0018	12.0922	11.8632	11.0645	10.1940	9.3794	8.6480
0.0000	8.5714	11.8982	12.7928	12.6790	12.1785	11.5519	10.9059
0.0000	7.4056	11.0904	12.6882	13.1829	13.1190	12.7844	12.3254
0.0000	6.4910	10.2119	12.1930	13.1266	13.4412	13.3985	13.1591
0.0000	5.7671	9.3899	11.5687	12.7939	13.4049	13.6218	13.5920
0.0000	5.1840	8.6549	10.9196	12.3369	13.1653	13.5957	13.7535

TABLE 3

Expected number of busy memories in one cycle :

Exponential distribution for tp

Constant tw=ta=E[tp]

Simulation results

m=	2	3	4	5	6	7	8
n=2	1.4088	1.5931					
n=3	1.6185	1.9878	2.2075				
n=4		2.2198	2.5643	2.8004			
n=5			2.7980	3.1472	3.4300		
n=6				3.4088	3.7122	4.0040	
n=7					3.9990	4.3196	4.5804
n=8						4.5666	4.9028

TABLE 4

	Processing Time	Memory Cycle Time	Analysis	Computational Ease
Discrete Markov Chain	Constant tp=tw	Constant	Exact	Solution is algorithmic. Unwieldy for large n.
Strecker's Approximation	Constant	Constant	Approximate	Closed form solution. Simple formula.
Continous Time Markov Chain	Exponential	Exponential	Exact	Closed form solution. Simple formula.
Diffusion Approximation	Constant	Constant	Approximate	Closed form solution. Simple formula.
Simulation Model	Exponential E[tp]=tw=ta	Constant	Approximate	Unwieldy due to slow stochastic convergence.

REFERENCES

AndeJ62 Anderson, J.P. et al.: D825 - A Multiple Computer System for Command and Control, AFIPS Proc. FJCC, Vol. 22, pp. 86-92, 1962.

BeckE64 Beckenbach, E. (editor): Applied Combinatorial Mathematics, Wiley, New York, 1964.

BellC71a Bell, C.G. and A. Newell: Computer Structures: Readings and Examples, McGraw-Hill, New York, 1971.

BellC71b Bell, C.G. et al.: C.mmp: The CMU Multiminiprocessor Computer, Department of Comp. Sci., Carnegie-Mellon Univ, August 1971

BuzeJ71 Buzen, J.B.: Queueing Network Models of Multiprogramming, Ph.D. Thesis, Harvard University, ESD-TR-71-345, August, 1971.

FellW66 Feller, W.: An Introduction to Probability Theory and its Applications, Vol. 2, Wiley, New York, 1966.

GaveD67 Gaver, D.P.: Probability Models for Multiprogramming Computer Systems, JACM, Vol. 14, No. 3, July 1967, pp. 623-638.

GordW67 Gordon, W.J. and G.F. Newell: Closed queueing systems with exponential servers, Oper. Res., 15 (1967), pp. 254-265.

JackJ63 Jackson, J.R.: Jobshop-like queueing systems, Management Sci., 10, 1 (Oct. 1963), pp. 131-142.

KobaH73 Kobayashi, H.: Application of the Diffusion Approximation to Queueing Networks: Part I - Equilibrium Queue Distributions, 1st Annual SIGME Conference on Measurement and Evaluation, March 1973, pp. 54-60.

LiuC68 Liu, C.L.: Introduction to Combinatorial Mathematics, New York, McGraw-Hill, 1968.

McCrJ73 McCredie, J.W.: Analytic Models as Aids for Multiprocessor Design, Proc. of the 7th Annual Princeton Conference on Information Science and Systems, March 1973.

MckiJ69 McKinney, J.M.: A Survey of Analytic Time Sharing Models, Computing Surveys, Vol. 1, No. 2, pp. 105-116, 1969.

MuntR72 Muntz, R.R. and F. Baskett: Queueing Network Models with Different Classes of Customers, Proc. of COMPCON 72, Sept. 1972, pp. 205-209.

NeweG71 Newell, G.F.: Applications of Queueing Theory, London, Chapman and Hall, 1971.

SkinC69 Skinner, C. and J. Asher: Effect of Storage Contention on System Performance, IBM Sys. J., Vol. 8, no. 4, 1969, pp. 319-333.

StreW70 Strecker, W.D.: Analysis of the Instruction Execution Rate in Certain Computer Structures, Ph.D. thesis, CMU. 1970.

WagnH69 Wagner, H.M.: Principles of Operations Research, Prentice-Hall, Englewood Cliffs, 1969.

WulfW72 Wulf, W.A. and C.G. Bell: C.mmp - A Multi-mini-processor, AFIPS FJCC Proc. 1972, Vol. 41, Part II, pp. 765-777.

APPENDIX I

The tree in Fig. 4.3 can be converted into a mesh by lumping together all occurrences of a partial state in the tree. e.g. state 2100 at level 3 appears twice. the resulting mesh for the 4 by 4 example is shown in Fig. 1. the algorithm for generating the transition matrix is shown in Fig. 4. though the implementation of this algorithm involves a matrix multiplication and requires more temporary storage it is faster than the algorithm in section 2 for larger n. Thus, a space-time trade-off affects the selection of the algorithm to be used.

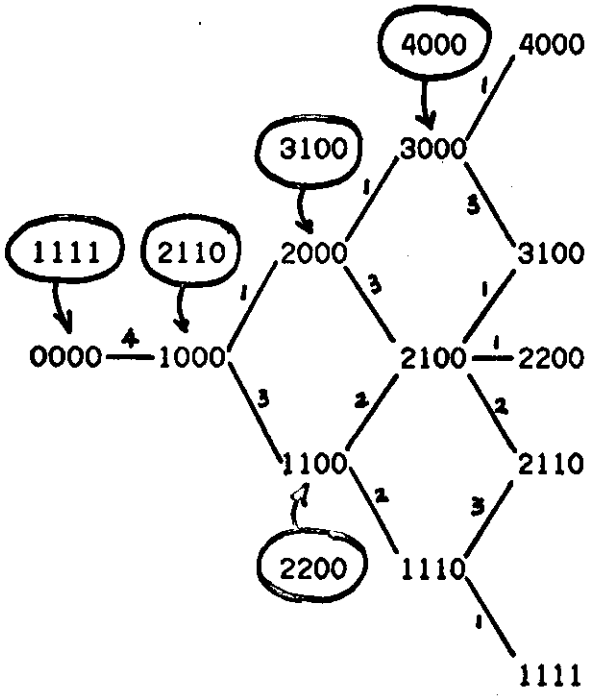


Fig. 1. Enumeration mesh for a 4 by 4 multiprocessor system.

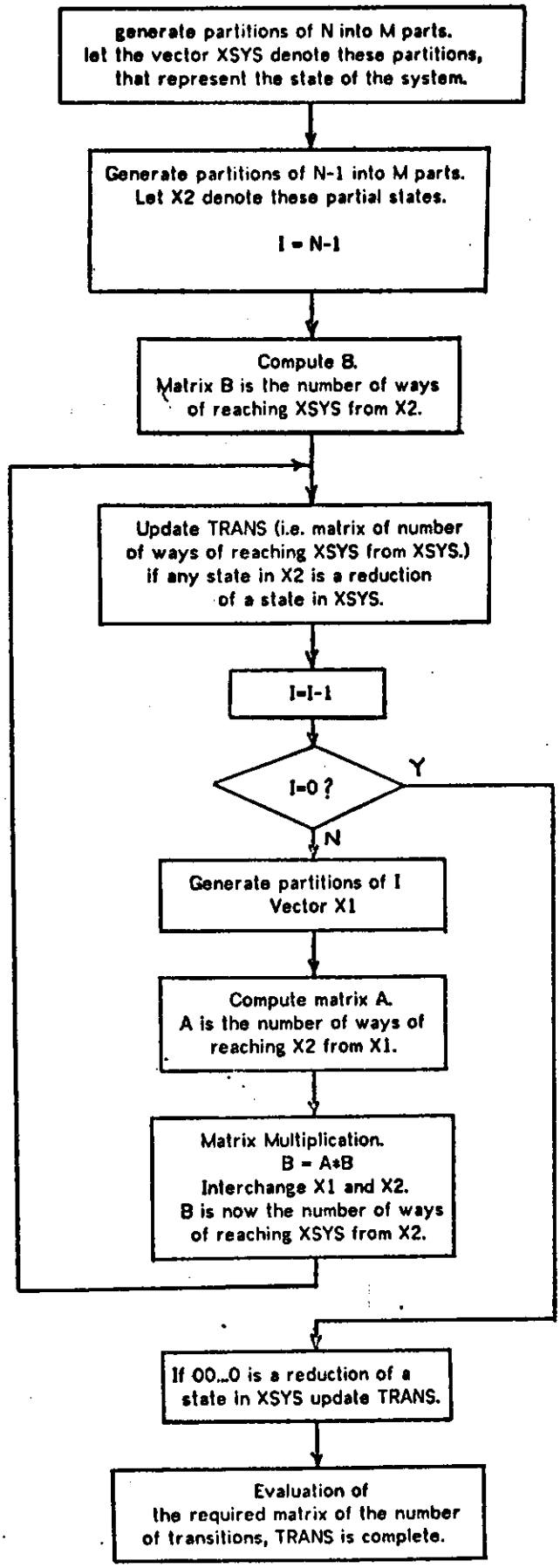


Figure 2 An algorithm for evaluating the transition matrix.