CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

CAPACITY ALLOCATIONS FOR DEMAND ASSIGNMENT MULTIPLE ACCESS SYSTEMS

by

KEVIN WONG

A thesis
presented to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
MASTER OF APPLIED SCIENCE
in
THE DEPARTMENT OF ELECTRICAL ENGINEERING

OTTAWA, Ontario, 1984

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

This thesis is concerned with the sharing of resources which are available in a demand assignment multiple access system. Such a system can be modelled probabilistically and is known to exhibit a closed product form solution for the state probabilities. However, a direct application of this solution to evaluate a system's performance requires a large amount of computation. By means of a recursive algorithm, these excessively long computations can be avoided. Using the recursive algorithm, in conjunction with a Pattern Search technique, an optimization package was developed for use in system design problems so that the system resources will be shared optimally.

ACKNOWLEDGEMENT

The author wishes to express his gratitude to his research group supervisor, Professor N.D. Georganas for his encouragement and guidance throughout this work.

Special thanks are also due to research group members Pavlos Kariotellis, Philip Mui and Miguel Rios for numerous stimulating discussions, both in matters of technical nature and otherwise.

The financial assistance of the Natural Sciences & Engineering Research Council is gratefully acknowledged.

CONTENTS

LIST OF FIGURES

Chapter I

INTRODUCTION

A mobile satellite system (MSAT) has been proposed by the Department of Communications of the Canadian Government to provide mobile radio and mobile telephone services in the 806 - 890 MHZ band. The economic viability of such a complex communication system depends upon many factors; one of the most critical is system utilization. In order to maximize utilization, the system should be sized to closely match the projected traffic demand. The research results presented in this thesis were motivated by the MSAT proposal.

A demand assignment multiple access (DAMA) system is considered here. In this system a pool of capacity (e.g. buffer memory, satellite frequency bands, satellite time slots, etc.) is to be shared, according to a given sharing policy, by a given number of user groups, each with a unique set of traffic characteristics. The mobile radio portion of MSAT can be viewed as such a system.

By properly choosing the sharing policy, a system's performance can be optimized. Sharing policies of practical interest are those that are simple to implement and require

no detailed measurement of arriving traffic. Six of these policies, namely, Complete Sharing, Complete Partitioning, Sharing with Maximum Allowed, Sharing with Minimum Allocated, Intermediate Sharing and Pre-emptive Priority Sharing are discussed here.

Analytical results for evaluating the performance of a system using any one of the aforementioned policies are available in the literature [1],[2],[3],[9]. However, straightforward application of these analytical results involves solving a set of equations simultaneously, requiring an intolerable amount of computation even for a small system. Clearly, an efficient algorithm to solve those equations is needed. Research results have appeared in recent publications which address this problem: notably, Kaufman's one-dimensional algorithm for the Complete Sharing policy [6] and Barberis et al's two-dimensional algorithm [1] applicable in all but the Pre-emptive Priority Sharing (An efficient algorithm for this policy is not available in the literature at this point in time). These algorithms permit the analytical results to be applied to systems of realistic sizes (i.e. large amount of resources shared by large numbers of user groups).

For a system with a given sharing policy, the aforementioned algorithms allow the performance of this system to be evaluated. However, in a design problem, the

questions confronted by the designer are: Which sharing policy should one choose? What are the optimal parameters associated with that chosen policy? The first question was answered by Foschini et al [7]; they proved that the policy of Sharing with Maximum Allowed is the optimal policy for systems with up to three user groups. To answer the second question, this thesis author has adopted the well known optimization technique- Pattern Search, along with Barberis' two-dimensional algorithm, and developed an optimization package for DAMA systems that use the policy of Sharing with Maximum Allowed. With slight modifications, this package can be used for systems with any sharing policy.

The balance of this thesis is organized as follows:- Chapter 2 discusses the six sharing policies that can be used in DAMA systems. Recursive algorithms for determining system performance are presented in Chapter 3. Chapter 4 addresses the system optimization problem using two numerical examples. Finally, the conclusions are drawn in Chapter 5.

Chapter II

DAMA CIRCUIT-SWITCHED SERVICE

2.1 MODEL

The system is modelled as a pool of N units of capacity shared by M groups of users on a demand assignment multiple access (DAMA) basis. In a frequency division multiple access (FDMA) system there are N frequency channels; in the case of a time division multiple access (TDMA) system, there are N time slots per cycle. Fig. 2.1 shows such a model.

In this system, each user group i, i=1,...,M is characterized by the number of traffic sources R_i (the result is an Engset model [1] if R_i is finite or an Erlang model if R_i is infinite), the offered load ρ_i and per call capacity requirement K_i. It is assumed that:

1. For each user group i, traffic is generated randomly (e.g. Poisson) with mean arrival rate λ_i and call duration exponentially distributed with average length $1/u_i$.
2. The system operates in a blocked-calls-cleared circuit-switched mode, i.e. an ith group user places a call on a statistical basis which is independent of all other users. When that user requests services,

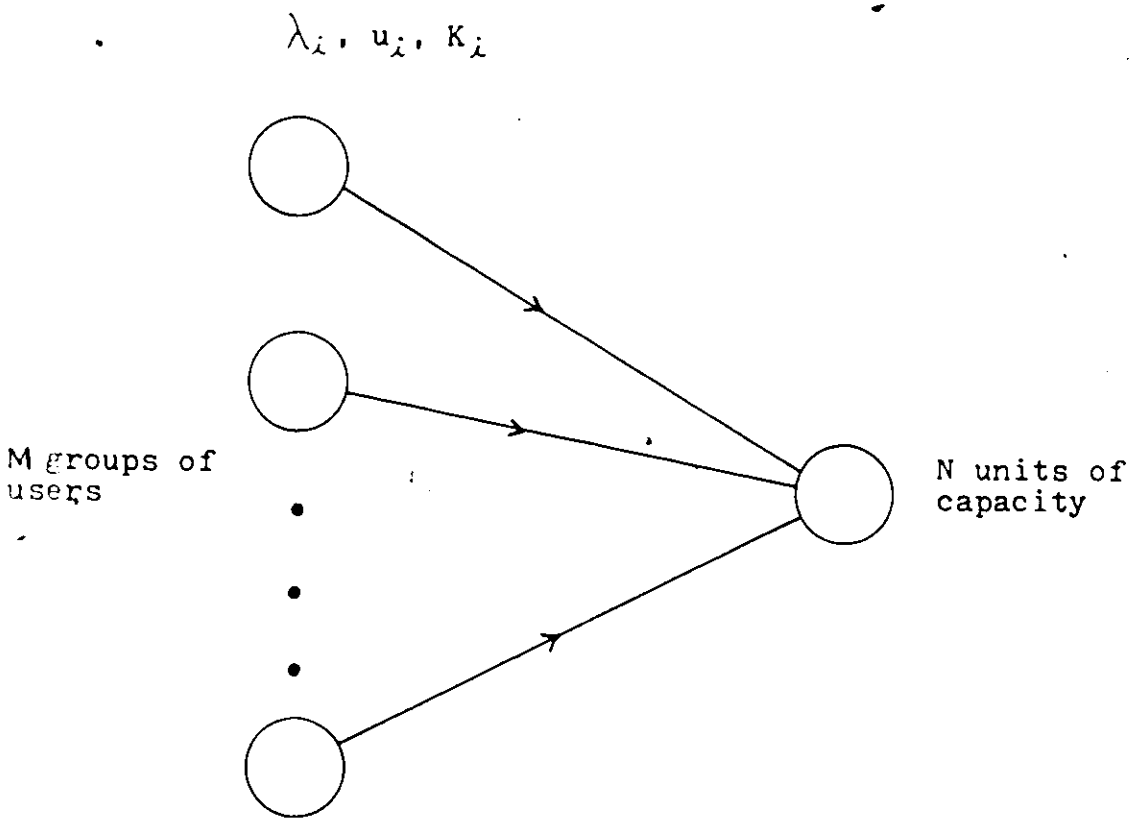


Fig 2.1 Statistical Model of the System

K_i units of capacity will be allotted if such a capacity is available. If not, the user is blocked and is modelled as returning to a status statistically identical to not having attempted the call.

3. The system capacity is shared linearly. This is the case when incoming users' resource requirements are independent of the users that are already in the system.

The system parameters are summarized as follows:-

N - Total number of capacity units available

M - Number of user groups

The ith group, i=1,...,M is characterized by:

R_i - Traffic source population

λ_i - Arrival rate

$1/u_i$ - Average call duration

ρ_i - offered load

K_i - Number of units of capacity required per call

N_i - Maximum number of calls allowed at any given instant

L_i - Number of calls with dedicated resources (logical reservation)

With the above assumptions, the statistical model leads to a closed product form solution [2],[3]. The state of the system can be defined by the vector $\vec{n}$, $\vec{n}=(n_1, \ldots, n_M)$ where

each n_i is the number of ith group calls in progress. The steady-state probability, $P(\vec{n})$ of the system being in state $\vec{n}$ is given in equation (2.1)

$$P(\vec{n}) = P(\vec{0}) \prod_{i=1}^{M} V_i(n_i), \quad \vec{n} \in A \tag{2.1}$$

where $V_i(n_i) = \frac{\rho_i^{n_i}}{n_i!}$ for an infinite population in group i

or $V_i(n_i) = \binom{R_i}{n_i} \rho_i^{n_i}$ for a finite population in group i

and $\rho_i = \frac{\lambda_i}{\mu_i}$

$A \triangleq$ set of feasible states which depends on the access discipline, system capacity N and number of user groups M.

$P(\vec{0}) \triangleq$ Probability of the state (0,...,0) occuring

$P(\vec{0})$ is also the normalization constant so that

$$\sum_{\vec{n} \in A} P(\vec{n}) = 1$$

$$P(\vec{0}) = \left\{ \sum_{\vec{n} \in A} \left[\prod_{i=1}^{M} V_i(n_i) \right] \right\}^{-1} \tag{2.2}$$

Hence $$P(\vec{n}) = \left\{ \sum_{\vec{n} \in A} \left[\prod_{i=1}^{M} V_i(n_i) \right] \right\}^{-1} \left\{ \prod_{j=1}^{M} V_j(n_j) \right\} \tag{2.3}$$

2.2 SHARING SCHEMES

As shown in the preceeding sub-section, the state of the system is described by $\vec{n} \triangleq (n_1,\ldots,n_M)$ whose coordinates n_i are non-negative integers representing the number of calls in progress from each of the M user groups. Constraints are placed upon the allowed states by the total capacity available and the manner in which it is to be shared amongst the user groups. For a system with a total of N units of capacity and each user from group i using K_i units of capacity, the total number of capacity units engaged at any instant is $\sum_{i=1}^{M} K_i n_i \leq N$. If the capacity needed to serve a new request were to make the summation to exceed N, then that request will be rejected. Additional constraints can be imposed depending on the access strategy, giving rise to a set of feasible states, A. Six sharing schemes are discussed in this chapter.

For each sharing strategy, its corresponding A-set can be represented by a M-dimensional geometrical object residing on the positive coordinate axes (fig 2.2). The outer surfaces of the object represent the states of the A-set at which new arrivals will be blocked. Thus the blocking probabilities of this system can be evaluated by summing up all the probabilities of being in the states represented by the outer surfaces.

The A-set shown in Fig 2.2 also possesses the important "coordinate-convexity" property [2] which provides a necessary condition for a product form solution (2.1), i.e. the orthogonal projection from any point in the A-set object to the plane formed by the positive coordinate axes below is contained in the A-set. This property can be defined formally as follows:-

Definition : An A-set is coordinate convex if

(1) $\vec{n} \in A \Rightarrow n_i \geq 0 \qquad i=1,\ldots,M$

(2) $\vec{n} \in A$ and $n_i > 0 \Rightarrow \vec{n}_i^- \in A$

where $\vec{n}_i^- \triangleq (n_1, \ldots n_{i-1}, n_i - 1, n_{i+1}, \ldots n_M)$

For an A-set that is not coordinate-convex, there exist some states in the system at which the termination of a call cannot take place until some other events occur (new arrival(s) or departure(s) by users from other groups). Fig 2.3 shows an example of an A-set which is non-coordinate-convex. Clearly, a sharing strategy that forms such an A-set is undesirable. A practical system should allow prompt departure upon call termination so that the resource can be returned immediately for reuse.

In the balance of this chapter, six sharing strategies that possess this property are described.

2.2.1 COMPLETE SHARING (CS)

If there are N units of capacity and the per call capacity requirement for the ith group users is K_i, then the instantaneous capacity in use, $\sum_{i=1}^{M} n_i k_i$ cannot exceed N. If the capacity needed to service a new call request makes the sum exceed N, this new call will be blocked. (This is the only condition on which a new arrival is blocked.) The allowed state-space, A is defined as follows:-

$$A=\{\vec{n}:\vec{n}=(n_1, n_2, \ldots n_M) \; ; \; \sum_{i=1}^{M} n_i K_i \leq N \} \; ,$$

Fig 2.2a shows the allowed states in the case of M=2. Here the dotted line is formed by the states at which all the capacity is occupied. By properly summing up the probabilities of the blocking states, the blocking probability of each group is found. Fig 2.2b depicts the M=3 case. It shows a tetrahedron bounded by the 3 positive axes and a tilted plane with a distance from the origin proportional to the total capacity of the system. Thus by representing a given A-set geometrically, the system's total capacity, sharing strategy used and the resultant blocking condition can be easily characterized.

2.2.2 COMPLETE PARTITIONING (CP)

Another strategy is to restrict user access to only that capacity pool allocated to its group. In this case, the problem reduces to one of M non-interacting systems, each of which is an one-dimensional Erlang B model [4] (assuming an infinite population size) with maximum allowed number of call N_i. The A-set is defined as follows:-

$$A=\{\vec{n}:\vec{n}=(n_1,n_2,\ldots,n_M);\ 0\leq n_i\leq N_i,\ \sum_{i=1}^{M} N_i K_i \leq N\}$$

Fig 2.2c and 2.2d show the resultant A-set for M=2 and M=3 respectively. Comparing fig 2.2d with fig 2.2b (CS), the rectangular box of 2.2d fits inside the A-set in 2.2b with its outermost corner touching the tilted plane.

In terms of unweighted utilization, (weighted utilization will be dealt with in Chapter 4) it is obvious from these figures that the same system adopting CP is inferior to CS since its feasible state space is a subset of that of CS.

2.2.3 SHARING WITH MAXIMUM ALLOWED (MA)

This strategy is similar to CS with an added constraint on the maximum number of users, N_i from each group in the system at any instant. In this scheme, all users are free to share the available capacity with other users until $n_i=N_i$ (for any i). After that, any new arrival from this group

will be blocked even if there is sufficient resource to serve this new call. This strategy restricts a heavy user's access to the system so that light users will enjoy better service.

Defining: $A=\{\vec{n}:\vec{n}=(n_1,n_2,\ldots,n_M);\ \sum_{i=1}^{M} n_i K_i \leq N;$

$0\leq n_i \leq N,\ i=1,2,\ldots M\}$

$N_i < \left[\frac{N}{K_i}\right]$ for at least one i

Note that if $N_i=[N/K_i]$ for all i, this strategy reduces to CS. The geometrical object representing the A-set is shown in fig 2.2e for M=2 and fig 2.2f for M=3.

2.2.4 SHARING WITH MINIMUM ALLOCATED (MN)

Here the difference from CS is that at least one group is guaranteed access to the resource for a certain number of call's (minimum logical reservation), L_i. The sum of these guaranteed allocations cannot exceed N. The remaining resource is then shared freely by all users. Hence,

$$A=\{\vec{n}:\vec{n}(n_1,n_2,\ldots,n_M)\ ;\ L_i\leq n_i\leq N_i,\ \sum_{i=1}^{M} n_i K_i < N\}$$

$$N_i = \left[\frac{N-\sum_{j=1}^{M} K_j L_j}{K_i}\right] + L_i \quad ;\quad 0<L_i<N_i \text{ for at least one i}$$

If $L_i=0$ for all i, the system is reduced to CS. Again, this strategy favours light users although the mechanism is different from MA. This can be concluded by observing the

similarity between fig 2.2e and fig 2.2g, and also between fig 2.2f and 2.2h.

2.2.5 INTERMEDIATE SHARING (IS)

Another strategy of interest is a combination of MA and MN in which less than N units are allotted as dedicated (logical, not physical) and the remainder is used as an overflow channel. Calls that are blocked on their dedicated channels share the remaining system capacity with other overflow calls. There is also a constraint that the total number of calls from each group is restricted to N_i.

$$A=\{\vec{n}:\vec{n}=(n_1,n_2,\ldots,n_M),\quad \sum_{i=1}^{M} n_i K_i \leq N,\ L_i \leq n_i \leq N_i\}$$

$$N_i < \left[\frac{N-\sum_{j=1}^{M} L_j K_j}{K_i}\right] + L_i \quad ; \qquad 0<L_i<N_i$$

Again the A-set has a geometrical shape similar to that of MA and MN (see fig 2.2i for M=2 and fig 2.2j for M=3).

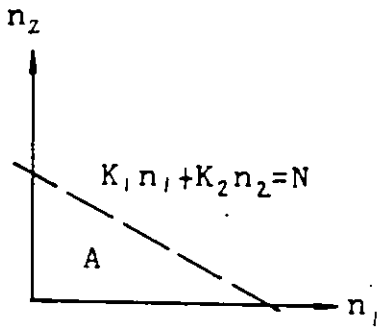


Fig 2.2a Complete Sharing
m=2

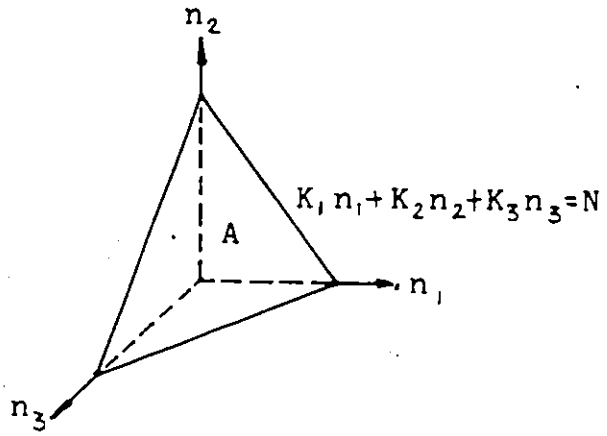


Fig 2.2b Complete Sharing
m=3

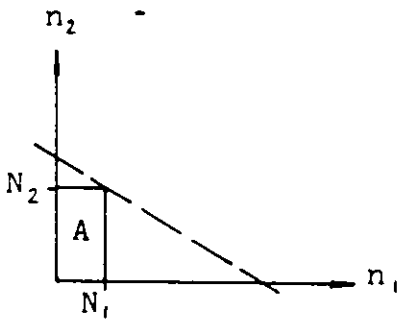


Fig 2.2c Complete Partitioning
m=2

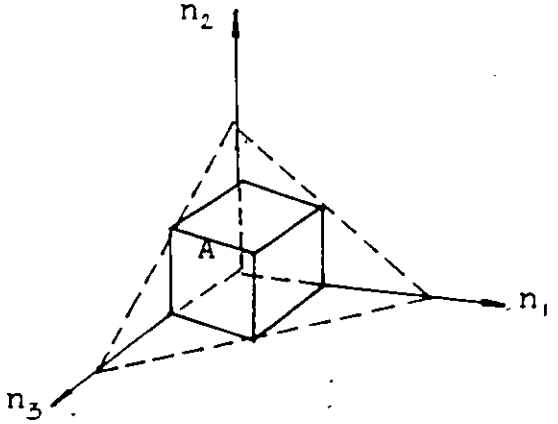


Fig 2.2d Complete Partitioning
m=3

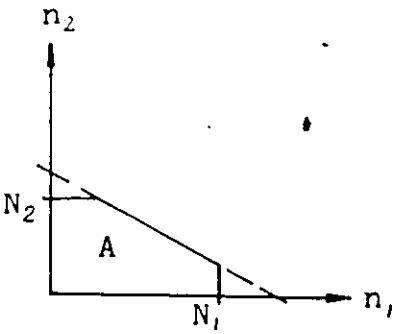


Fig 2.2e Sharing with Maximum
Allowed, m=2

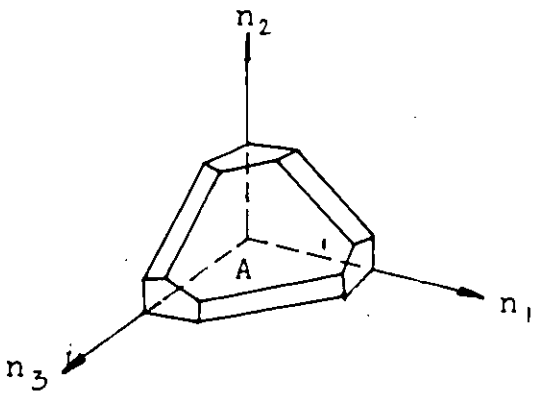


Fig 2.2f Sharing with Maximum
Allowed, m=3

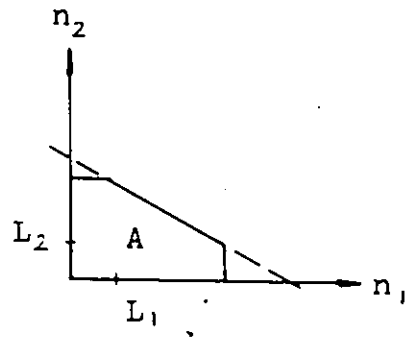


Fig 2.2g Sharing with Minimum Allocated, m=2

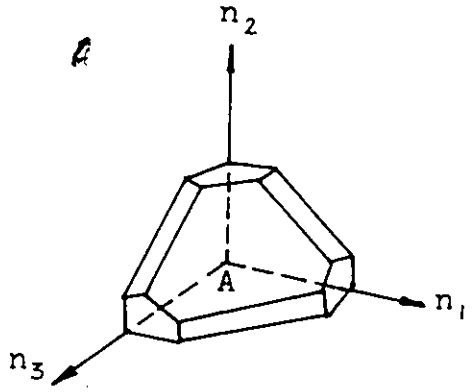


Fig 2.2h Sharing with Minimum Allocated, m=3

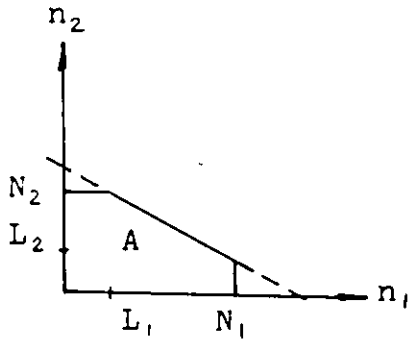


Fig 2.2i Intermediate Sharing m=2

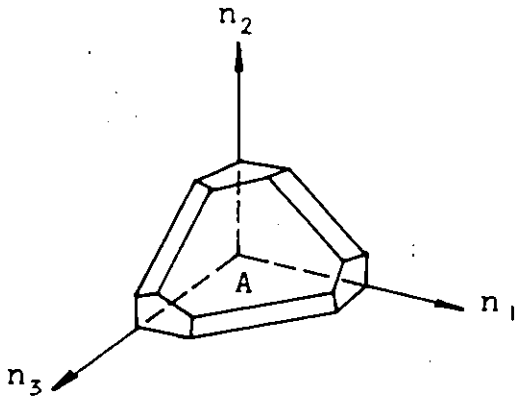


Fig 2.2j Intermediate Sharing m=3

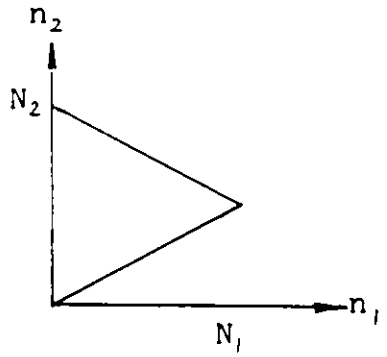


Fig 2.3 Non-Coordinate Convex Policy

2.2.6 PRE-EMPTIVE PRIORITY SHARING [5]

2.2.6.1 MODEL

The pre-emptive nature of the priority assignments implies that the presence of the lower priority users is transparent to higher priority ones. Let n_i be the number of ith group users in the system. Priority is assigned such that group i has higher priority than group i+1. Define $S^{(i)}_{n_i}$ as the event of allocating $n_i K_i$ units of capacity to the ith group users. Since N is finite, the maximum number of ith group users cannot exceed some finite number N_i. Then each group can be represented by the state set $Q^{(i)}$ which is defined as:

$$Q^{(i)} = \{S^{(i)}_{n_i} : n_i = 0, 1, \ldots, N_i\} \qquad (2.4)$$

$$N_i = \left[\frac{N}{K_i}\right]$$

for i=1 (i.e. highest priority) the group is modelled by

$$Q^{(1)} = \{S^{(1)}_{n_1} : n_1 = 0, 1, \ldots, N_1\} \qquad (2.5)$$

i.e. the number of allowed group 1 users ranges from o to N_1. For each value that n_1 takes on above, it implies that the second group (next highest priority) is allowed $\left[\frac{N - n_1 K_1}{K_2}\right]$ number of users. Let this number be denoted by $N_2(n_1)$ - maximum n_2 allowed given the presence of n_1. Therefore due to possible pre-emption by the first group, the second group can be represented by a collection of

(N_1+1) subsystems, $q^{(2)}(n_1)$. Each subsystem consists of $N_2(n_1)+1$ states:-

$$Q^{(2)}=\{q^{(2)}(n_1):n_1=0,\ldots,N_1\} \qquad (2.6)$$

where $$q^{(2)}(n_1)=\{S^{(2)}_{n_2}(n_1):n_2=0,\ldots,N_2(n_1)\} \qquad (2.7)$$

The approach leads to the following representation for the ith group:

$$Q^{(i)}=\{q^{(i)}(n_1,n_2,\ldots,n_{i-1}):n_1=0,1,\ldots,N_1;n_2=0,1,\ldots,N_2(n_1);$$

$$n_{i-1}=0,1,\ldots,N_{i-1}(n_1,n_2,\ldots,n_{i-2})\} \qquad (2.8)$$

$$q^{(i)}(n_1,n_2,\ldots,n_{i-1})=\{S^{(i)}_{n_i}(n_1,n_2,\ldots,n_{i-1}):n_i=0,1\ldots$$

$$N_i(n_1,n_2,\ldots,n_{i-1})\} \qquad (2.9)$$

$$N_i(n_1,n_2,\ldots,n_{i-1})=[(N-\sum_{\substack{j=1\\ j\neq i}}^{M} n_j K_j)/K_i] \qquad (2.10)$$

Fig 2.4 shows the generation of subsystem capacities and model dimension.

The objective here is to determine the blocking probability experienced by each group. Due to pre-emption, the probability distribution in a system of higher priority biases the distribution in all other groups with lower priority. This difficulty can be circumvented by representing each lower priority system with a collection of subsystem rather than a single system. Thus each subsystem is imbedded in a state of a higher priority system and also

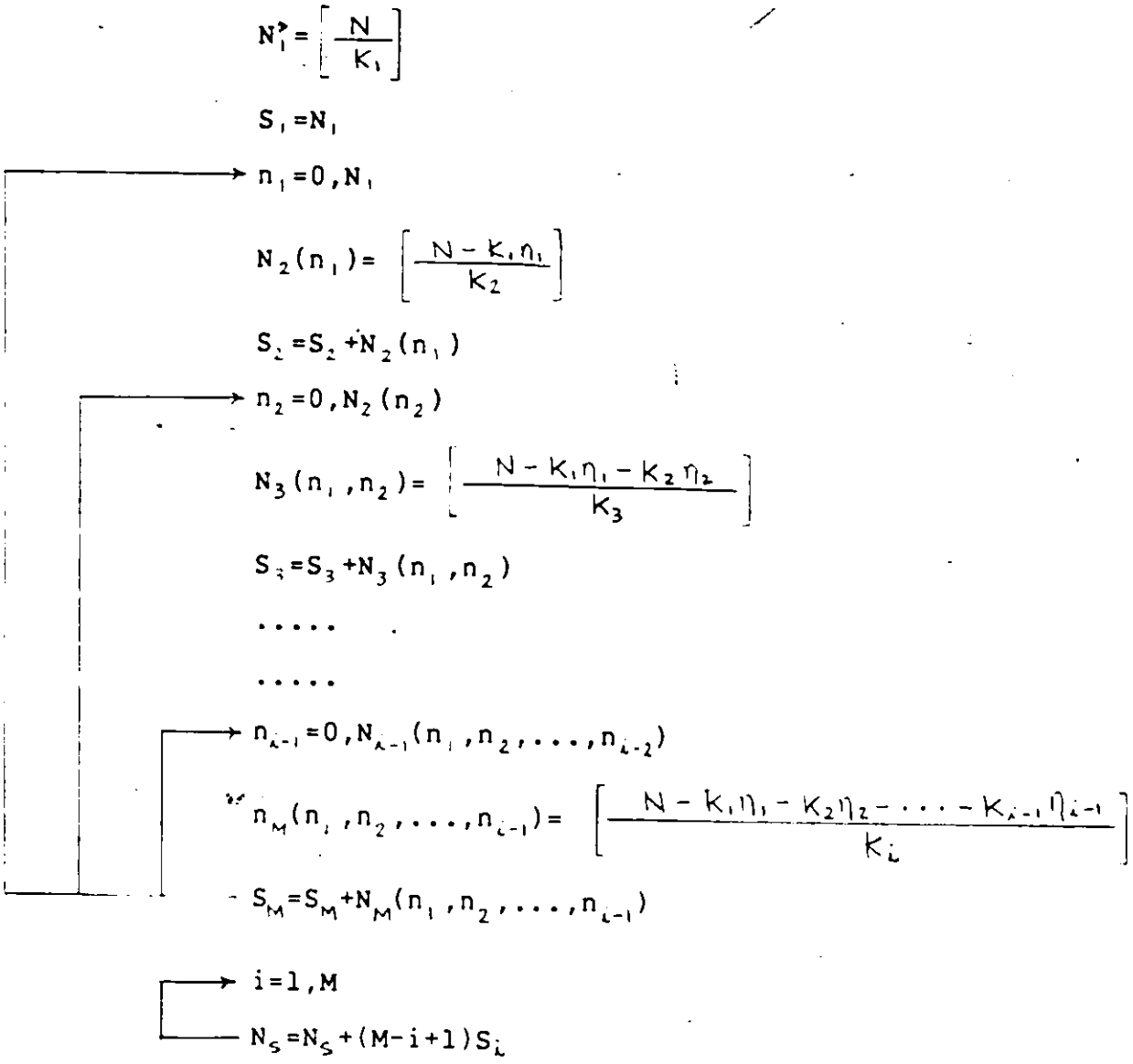


Fig 2.4 Generation of Subsystem Capacities and Model Dimension [5]

each state contains a lower priority subsystem. The following summarizes the approach:-

1. Choose a method to model the highest priority system.
2. Within each state of that system imbed a subsystem which models the next highest priority group conditionally.
3. Repeat (2) until all M groups are modelled.

The probability in each subsystem is determined for the groups' arrivals and departures. This probability is conditional and they must be combined to give the unconditional one.

2.2.6.2 CONDITIONAL PROBABILITY

Since all pre-emption effects have been imbedded in this model, the steady state probability can be found by treating each subsystem as an M/M/N/N queueing system where N is the system capacity. This model has the well known product form solution. Let $P^{(i)}_{n_i : n_{i-1}}$ be the steady state probability of having n_i users from the ith group in the system, given that there are n_{i-1} users from the (i-1)th group. For i=1 (highest priority, all other groups are transparent to it) the probability of state $S^{(1)}_{n_1}$ occuring in the $Q^{(1)}$ system is:

$$P^{(1)}_{n_1} = \left(\sum_{j=0}^{N_1} \frac{\rho_1^j}{j!} \right)^{-1} \frac{\rho_1^{n_1}}{n_1!} \qquad n_1 = 0, 1, \ldots, N_1 \qquad (2.11)$$

Each state $S^{(2)}_{n_2:(n_1)}$ in the $q^{(2)}_{(n_1)}$ subsystem of the $Q^{(2)}$ system has a conditional probability of occurence equal to:

$$P^{(2)}_{n_2:n_1} = \left(\sum_{j=0}^{N_2(n_1)} \frac{\rho_2^j}{j!} \right)^{-1} \frac{\rho_2^{n_2}}{n_2!} \qquad n_2 = 0,1,\ldots,N_2(n_1) \quad (2.12)$$

Similarly, the $S^{(i)}_{n_i(n_1,n_2,\ldots,n_{i-1})}$ state of the $q^{(i)}_{(n_1,n_2,\ldots,n_{i-1})}$ subystem of $Q^{(i)}$ has a conditional probability of occurence equals to:

$$P^{(i)}_{n_i:n_1,n_2,\ldots,n_{i-1}} = \left(\sum_{j=0}^{N_i(n_1,n_2,\ldots,n_{i-1})} \frac{\rho_i^j}{j!} \right)^{-1} \frac{\rho_i^{n_i}}{n_i!} \qquad (2.13)$$

2.2.6.3 UNCONDITIONAL PROBABILITY

From the conditional probabilities in the preceeding subsection, unconditional probabilities can be determined:- for i=1, equation 2.11 gives the unconditional probability.

for i=2 $$P^{(2)}_{n_2} = \sum_{n_1=0}^{N_1} P^{(1)}_{n_1} P^{(2)}_{n_2:n_1} \qquad n_2 = 0,1,\ldots,N_2 \qquad (2.14)$$

in general $$P^{(i)}_{n_i} = \sum_{n_1=0}^{N_1} P^{(1)}_{n_1} \cdot \sum_{n_2=0}^{N_2(n_1)} P_{n_2:n_1} \cdots \sum_{n_{i-1}=0}^{N_{i-1}(n_1,\ldots,n_{i-2})} P_{n_{i-1}:n_{i-2},n_{i-3}\cdots n_1} \cdot P^{(i)}_{n_i:n_{i-1},\ldots,n_1} \quad ; \quad n_i = 0,1,\ldots,N_i \quad (2.15)$$

2.2.6.4 BLOCKING PROBABILITY

The blocking probability BP of each user group can now be found by summing all the states at which the maximum allowed N_i user from the ith group is met:-

$$BP_i = P^{(i)}_{N_i} \qquad (2.16)$$

$$BP_2 = \sum_{n_1=0}^{N_1} P_{n_1}^{(1)} P_{N_2(n_2):n_1}^{(2)} \qquad (2.17)$$

$$BP_3 = \sum_{n_1=0}^{N_1} P_{n_1}^{(1)} \cdots \sum_{n_{i-1}=0}^{N_{i-1}(n_1,\ldots,n_{i-2})} P_{n_{i-1}:n_{i-2},\ldots n_1}^{(i-1)} \cdot P_{N_i(n_1,n_2,\ldots n_{i-1}):n_{i-1}\ldots n_1}^{(i)} \qquad (2.18)$$

Chapter III

SOLUTION ALGORITHMS

In the previous chapter, it was observed that the statistical model of the DAMA system under consideration leads to a product form analytical solution (2.3) for the steady-state probabilities. From these quantities, the performance of the system in terms of throughput, utilization, delay etc. can be evaluated [3]. However, direct application of this equation even for a system with a small number of capacity units, N, and a small number of users groups, M, may require an intolerable amount of computation. Clearly an efficient algorithm is required for the design of realistic systems.

In the next section, an one-dimensional recursive method is presented. This method is applicable only for a system which adopts the complete sharing policy. A two-dimensional algorithm that is applicable for all but the pre-emptive priority policy is presented in the subsequent section. (One that is suitable for pre-emptive priority sharing is not available in the literature at this point in time.)

These algorithms provide efficient means of evaluating the performance of a given system analytically. But more

importantly, their efficient nature allows a practical design problem to be addressed when it is used in conjunction with a suitable optimization technique. This is the subject of discussion of the following chapter.

3.1 ONE-DIMENSIONAL RECURSIVE METHOD [6]

In section 2.1, it was observed that the steady-state probabilities, $P(\vec{n})$ are given by:

$$P(\vec{n}) = P(\vec{0}) \prod_{i=1}^{M} V_i(n_i) \qquad , \vec{n} \in A \qquad (3.1)$$

where $\vec{n} = (n_1, \ldots, n_M)$; $n_i \triangleq$ number of users from group i in the system

$V_i(n_i) = \frac{\rho_i^{n_i}}{n_i!}$ for an Erlang model

or $V_i(n_i) = \binom{R_i}{n_i} \rho_i^{n_i}$ for an Engset model

$\rho_i = \lambda_i / u_i$

$P(\vec{0}) \triangleq$ Normalization constant, which is also the probability that the system is empty.

and $A \triangleq$ Set of feasible states which depends on the system capacity N, number of group of users, M and resource sharing policy.

In this section, only the complete sharing policy is considered. Fig 3.1 shows the feasible state space of this policy for M=2. n_1 and n_2 are the numbers of users from group 1 and 2 respectively.

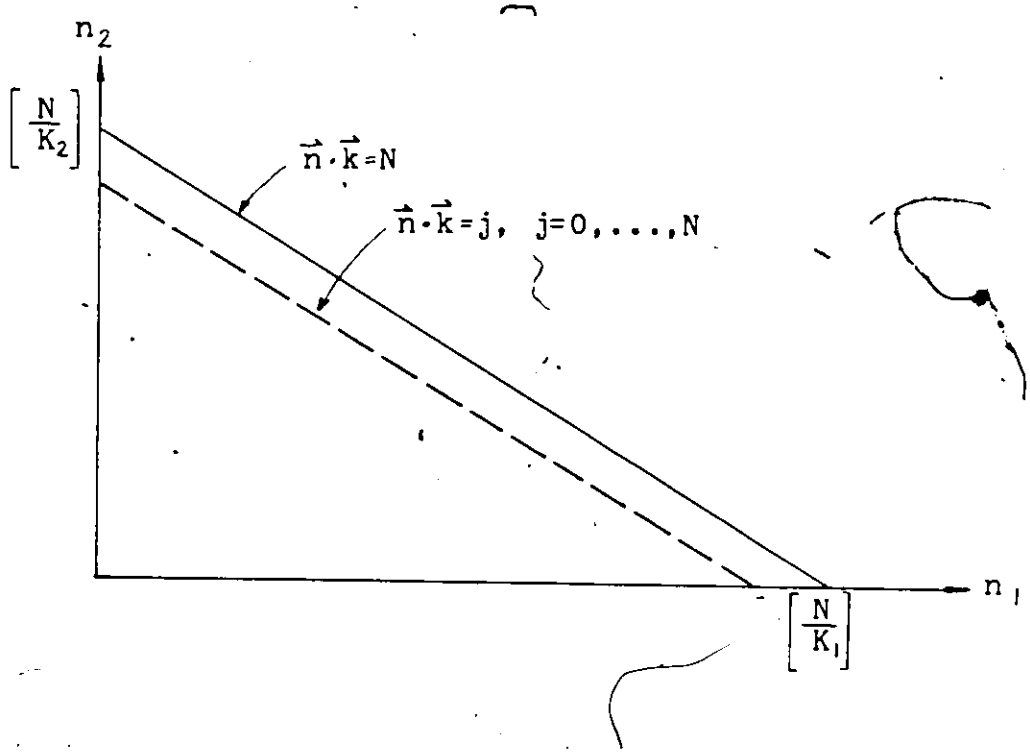


Fig 3.1 Feasible State Space for M=2

The diagonals j represent the total number of resource units occupied, i.e. $j=\vec{n}\cdot\vec{k}=0,1,\ldots,N$. This random variable j has a distribution q(j) so that

$$q(j)=\sum_{\vec{n}:\vec{n}\cdot\vec{K}=j}\prod_{i=1}^{M}V_i(n_i)P(\vec{0}) \qquad (3.2)$$

It can be shown that the following relationship among q(j) exists [4]:-

$$j\,q(j)=\sum_{i=1}^{M}\rho_i K_i q(j-K_i)\ ,j=0,1,\ldots,N \qquad (3.3)$$

where $q(x)=0$ for $x<0$

and $\sum_{j=0}^{N}q(j)=1$ (3.4)

Equation 3.3 defines an one-dimensional recursion which generates q(j) recursively.

Defining $\hat{q}(j)=\frac{q(j)}{q(0)}$; $\hat{q}(0)=1$

Hence $q(j)=q(0)\hat{q}(j)$ $j=1,\ldots,N$

we obtain from (3.3): $\hat{q}(j)=\frac{1}{j}\sum_{i=1}^{M}P_i K_i\hat{q}(j-K_i)$, $j=1,\ldots,N$

From (3.4) $\sum_{j=1}^{N}q(j)=q(0)\sum_{j=1}^{N}\hat{q}(j)=1$

$$\therefore\ q(0)=\left[\sum_{j=1}^{N}\hat{q}(j)\right]^{-1}$$

Having determined q(j) for all j produced two important results:

1. Since $q(0)=P(\vec{0})$, the state distribution of the system, $P(\vec{n})$, as given in (3.1), is determined.
2. More significantly, the system performance of each user group can be obtained directly from q(j):-

   i. Blocking probability, B_i:

since $B_i = \sum_{\{\vec{n}:\vec{n}\cdot\vec{K} > N-K_i\}} P(\vec{n})$, $i=1,\ldots,M$

it can be written as

$$B_i = \sum_{j=0}^{K_i-1} q(N-j) \quad , \quad i=1,\ldots,M \qquad (3.5)$$

ii. Utilization, U:-

From Little's result [3], the mean number of users from group i, $E\{n_i\}$ is

$$E\{n_i\} = \rho_i(1-B_i)$$

$$U = \frac{1}{N}\sum_{i=1}^{M} K_i E\{n_i\}$$

$$= \frac{1}{N}\sum_{i=1}^{M} \rho_i K_i (1-B_i) \quad ; \quad i=1,\ldots,M \qquad (3.6)$$

iii. Throughput, T_i :-

$$T_i = \lambda_i(1-B_i) \qquad (3.7)$$

This section is concluded by noting that the storage requirement of this one-dimensional recursion does not depend on the value of N, the total number of capacity units. To compute q(0), only a single counter to accumulate $\sum_{i=1}^{j} q(i)$, $j=1,\ldots,N$ is needed. To compute B_i, only M counters are required to store $\sum_{j=0}^{K_i-1} q(N-j)$.

Compared with the two-dimensional recursion presented in the next section, this one-dimensional method is superior both in terms of computational speed and storage

requirements. However, it is only applicable for a system with a Complete Sharing policy. In order to extend this method to other policies considered, more research work is needed.

3.2 TWO-DIMENSIONAL RECURSIVE METHOD [1]

The objective of this section is to introduce a two-dimensional recursive algorithm to find $P(\vec{0})$. This is done by constructing an N X M matrix sequentially with the last entry equal to $P(\vec{0})$. In addition, explicit expressions for the blocking probability of each group for each sharing discipline are given so that all quantities of interest can be computed from this matrix without resorting to the underlying distribution. The explicit expressions are shown in 3.2.1 and the algorithm is shown in 3.2.2.

3.2.1 ANALYTICAL RESULTS

As shown in (2.2)

$$P(\vec{0}) = \left\{ \sum_{\vec{n} \in A} \left[\prod_{i=1}^{M} V_i(n_i) \right] \right\}^{-1}$$

Defining the matrix S(n,m) so that for n=N and m=M, $S(N,M) = [P(\vec{0})]^{-1}$. The following shows how all blocking probabilities can be found from this matrix [1].

3.2.1.1 COMPLETE SHARING (CS)

Blocking of an ith group user in this discipline occurs only when the number of units of capacity engaged is greater than $N-K_i$. The blocking set is:-

$$D_i^{CS}(N,M)=\{\vec{n}:\vec{n}=(n_1,\ldots,n_M),\ N-K_i<\sum_{h=1}^{M} n_h K_h \leq N\}$$

obviously

$$D_i^{CS}(N,M)=A(N,M)-A(N-K_i,M)$$

where A is the set of feasible states.

The blocking probability of the ith group B_i can be defined as:

$$B_i=\sum_{\vec{n}\in D_i(N,M)} P(n) \qquad (3.8)$$

Clearly, $$B_i=\frac{S(N,M)-S(N-K_i,M)}{S(N,M)};\ i=1,\ldots,M \qquad (3.9)$$

3.2.1.2 SHARING WITH MAXIMUM ALLOWED (MA)

In this discipline, blocking occurs due to insufficient capacity (the number of units of capacity engaged is greater than $N-K_i$) as well as when $n_i=N_i$ (recall that N_i is the maximum number of users from group i allowed in the system at any one time). Therefore, the blocking states for Sharing with Maximum Allowed for the Mth group,

$$D_M^{MA}(N,M)=\{\vec{n}:\vec{n}=(n_1,\ldots,n_M);N-K_M<\sum_{h=1}^{M} n_h K_h \leq N_i : n_i \leq N_i,$$

$$i=1,\ldots,M\}\cup\{\vec{n}:\vec{n}=(n_1,\ldots,n_M);n_M=N_M;$$

$$\sum_{h=1}^{M-1} n_h K_h \leq N-K_M(N_M+1); n_i < N_i,\ i=1,\ldots,M-1\}$$

One could easily recognize that the blocking states comprise that of Complete Sharing plus the case when the maximum number allowed for the Mth group is reached. From (3.8) it can be shown that [1]:

$$B_M^{MA} = \frac{S(N,M)-S(N-K_M,M)\ S(N-K_M(N_M+1),M-1)V_M(N_M)}{S(N,M)} \quad (3.10)$$

To find B_i^{MA} for $i \neq M$, relabel the indexes, re-evaluate the matrix $S(n,m)$ for $n=1,\ldots,N$ and $m=1,\ldots,M$ and apply (3.10) until all blocking probabilities are obtained.

3.2.1.3 INTERMEDIATE SHARING (IS)

Recall from 2.2.5 that if a maximum N_i is imposed on the number of calls from each group and a number L_i of calls is guaranteed access,

i.e. $N_i < [N/K_i]$ and $L_i > 0$

we have the case of Intermediate Sharing.

A block diagram of such a system is shown in fig 3.2.

With a fixed number of capacity units K_iL_i reserved for each group, (logical reservations, i.e. without specifying which capacity unit is dedicated to which group), the total reserved capacity $\sum_{i=1}^{M} K_iL_i$ forms a dedicated channel. The rest of the system capacity, $\tilde{N}$ (see below) forms a "Shared Channel". New arrivals from any group that has used up its

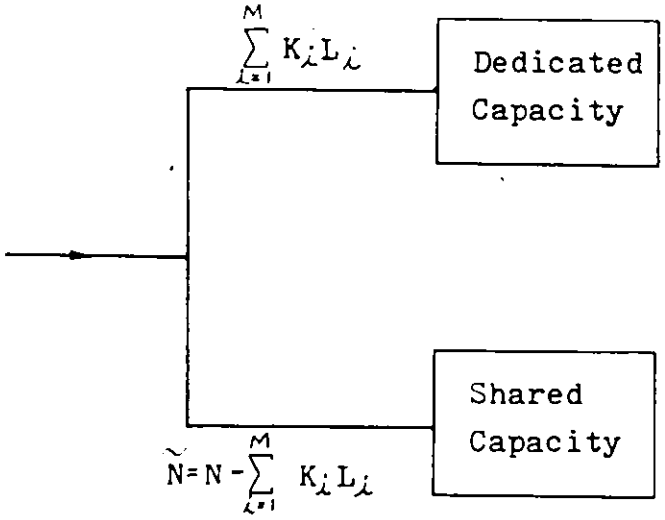


Fig 3.2 Block Diagram of an Intermediate Sharing System

reserved capacity can go on to contend for resources in that channel.

Defining the following:-

$\tilde{N} \triangleq$ Number of capacity units in the shared channel

$$= N - \sum_{i=1}^{M} K_i L_i$$

$\tilde{N}_i \triangleq$ Maximum number of users allowed in the shared channel

$\therefore \; N_i = \tilde{N}_i + L_i$

$\tilde{\bar{n}}_i \triangleq$ Number of users in the Shared Channel from group i

Since new arrivals from group i will enter into the shared channel only when the dedicated resources of this group are used up, so that:

$$\tilde{n}_i = \begin{cases} n_i - L_i & \text{if } n_i > L_i \\ 0 & \text{if } n_i \leq L_i \end{cases}$$

The performance of the overall system can be evaluated by merely considering the shared channel: all states with $n_i < L_i$ for all i produce a single empty state for the shared channel $[\tilde{\bar{n}} = (0, \ldots, 0)]$ and the ith group has any influence on the other groups only when $\tilde{N}_i + L_i \geq n_i > L_i$; consequently blocking of new arrivals is possible only if $n_i \geq L_i$. Using the same approach as in 3.2.1, letting

$$S(N,M) = \sum_{\tilde{\bar{n}} \in A(\tilde{N},M)} \prod_{i=1}^{M} \tilde{V}_i(\tilde{n}_i) \qquad (3.11)$$

where

$$\tilde{V}_i(\tilde{n}_i) = \begin{cases} \sum_{j=0}^{L_i} V_i(j) & \text{if } \tilde{n}_i = 0 \\ V_i(\tilde{n}_i + L_i) & \text{if } \tilde{n}_i > 0 \end{cases} \qquad (3.12)$$

$i = 1, \ldots, M$

and $A(\tilde{N},M) = \{\tilde{\bar{n}} : \tilde{\bar{n}} = (\tilde{n}_1, \ldots \tilde{n}_M); \sum_{i=1}^{M} K_i \tilde{n}_i \leq \tilde{N};$

$$\tilde{n}_i \le \tilde{N}_i, \quad i=1,\ldots,M\} \tag{3.13}$$

for the ith group

$$\hat{S}_i(\tilde{N},M) = \sum_{\tilde{n}\in A(\tilde{N},M)} \prod_{\substack{j=1 \\ j\neq i}}^{M} \tilde{V}_j(\tilde{n}_j)\cdot\hat{V}_i(\tilde{n}_i)$$

$$\hat{V}_i(\tilde{n}_i) = V_i(\tilde{n}_i + L_i) \quad \text{for } i=1,\ldots,M$$

So that the blocking probability for sharing with minimum allocated:

$$B_i^{MN} = \frac{\hat{S}_i(\tilde{N},M) - \hat{S}_i(\tilde{N}-K_i,M)}{\tilde{S}(\tilde{N},M)} \tag{3.14}$$

If the policy of Intermediate Sharing is used, then

$$B_i^{IS} = \frac{\hat{S}_M(\tilde{N},M) - \hat{S}_M(\tilde{N}-K_M,M) + \hat{S}(\tilde{N}-K_M(\tilde{N}_M+1),M+1)\cdot\tilde{V}_M(\tilde{N}_M)}{\tilde{S}(\tilde{N},M)} \tag{3.15}$$

3.2.2 RECURSIVE EVALUATION OF S(N,M)

In the previous section, explicit expressions for finding the performance of a DAMA system are given in terms of S(n,m), the normalization constant of a system with n number of capacity units and m groups of users.

The following expression gives a recursive formula that allows fast computation of that constant. The proof is detailed in Appendix A.

$$S(n,m) = \sum_{h=0}^{b_m(n)} S(n-K_{mh},m-1)V_m(h) \tag{3.16}$$

where $b_m(n) = \min\left[N_m, \left[\frac{n}{K_m}\right]\right]$

and $V_m(h) = \frac{\rho_m^h}{h!}$

Recall that S(n,m) is the reciprocal of the system's empty state probability, the following initial conditions hold:-

$$S(n,0)=1 \quad n=0,\ldots,N$$

$$S(0,m)=1 \quad m=1,\ldots,M$$

This algorithm gives the blocking probability of every group in only MXN^2 operations- fast when compared with the straightforward application of equation 3.1, which requires roughly N^M operations. However, when compared with the one-dimensional algorithm of section 3.1, it is inferior. Unfortunately, this one-dimensional algorithm is only applicable to a Complete Sharing system (see section 3.1).

Some overflow problems may occur when N is large. (recall that $V_m(N)=\frac{\rho_m^N}{N!}$). To address this problem, an appropriate scaling factor, $e^{-\rho_m}$ is required, so that:

$$S'(n,m)=\sum_{h=0}^{b_m(n)} e^{-\rho_m} S(n-K_{mh},m-1)V_m(h) \qquad (3.17)$$

With $b_m(n)$ and $V_m(b)$ defined as before

$$b_m(n)=\min\left[N_m,\frac{n}{K_m}\right] \quad ,V_m(h)=\frac{\rho_m^h}{h!}$$

Substituting all S with S' in equations 3.9, 3.10 and all $\hat{S}$ with $\hat{S}'$ in equations 3.14 and 3.15, the blocking probabilities can be determined.

For a system with N units of capacity and M user groups, the computational requirement involves constructing an N×M matrix, starting with the initial condition S(0,m) equal to 1 for m=1,...,M, filling in the matrix column by column, with the last entry being the normalization constant of that system's state probabilities. From this matrix, the system's various performance parameters can be obtained without resorting to the state probabilities.

3.3 NUMERICAL EXAMPLES

As an example, a system with a maximum of 50 capacity units (N=50) shared by 5 classes of users (M=5) was considered. The traffic intensities of the 5 different user classes were arbitrary chosen at $1.5\beta:3\beta:6\beta:12\beta:15\beta$ where β is a constant.

3.3.1 COMPLETE SHARING

In figure 3.3a and 3.3b, the case of Complete Sharing was assumed. Fig 3.3a shows the blocking probability degradation if N was reduced from the full capacity of 50 units. Four cases of β were considered β=0.9, 1.0, 1.1 and 1.2. In a system design problem where the maximum acceptable blocking probability is specified, a similar figure can be used to determine the minimum number of capacity units required in order to meet the constraint in

blocking probability. Fig 3.3b shows the blocking probabilities if the system's incoming traffic changes but keeping their ratio the same as above. (Blocking probabilities Vs β)

3.3.2 SHARING WITH MAXIMUM ALLOWED

In figures 3.4a to 3.4e, the discipline of Sharing with Maximum Allowed was assumed, and traffic intensity, β =1 was used. For the ith class users, a constraint of maximum number of users from that class, N_i, was imposed while all other user classes did not have that constraint. The blocking probabilities were plotted as a function of N_i. Clearly, a trade-off exists here such that if a user class can tolerate a higher blocking probability, a constraint, N_i, can be imposed so that other classes can be favoured.

The CPU time consumed in collecting data for fig 3.3 and 3.4 was small, (tenths of a second using an AMDAHL 470/V78 main frame computer), thanks to the efficient algorithm shown in section 3.2.

3.3.3 PRE-EMPTIVE PRIORITY

Fig 3.5 shows the blocking probabilities for various traffic intensities (blocking probabilities Vs β) when the pre-emptive priority strategy was adopted. N was chosen to be 50. The order of priority was assigned arbitrarily and

indicated on the figure. (In a real life design problem, the order of priority should be chosen so that the performance specifications are met) In this example, results show that the class that enjoys the highest priority has a blocking probability of $<10^{-6}$ -at the expense of other users as expected. It was also shown that the performance was very sensitive to the value of N. Partial results are shown here in the cases of N=45, 47 and 49 (For graphical simplicity, only the blocking probability of the 2nd priority class was plotted here). The result of the complete sharing case for N=50 was superimposed on the graph, indicating possible trade off.

Due to the lack of an efficient algorithm, the computing time grows rapidly with N (for N=50, CPU time was 7.5 minutes). This is the area where more research is needed to be done in order to make use of the analytical results in practical problems.

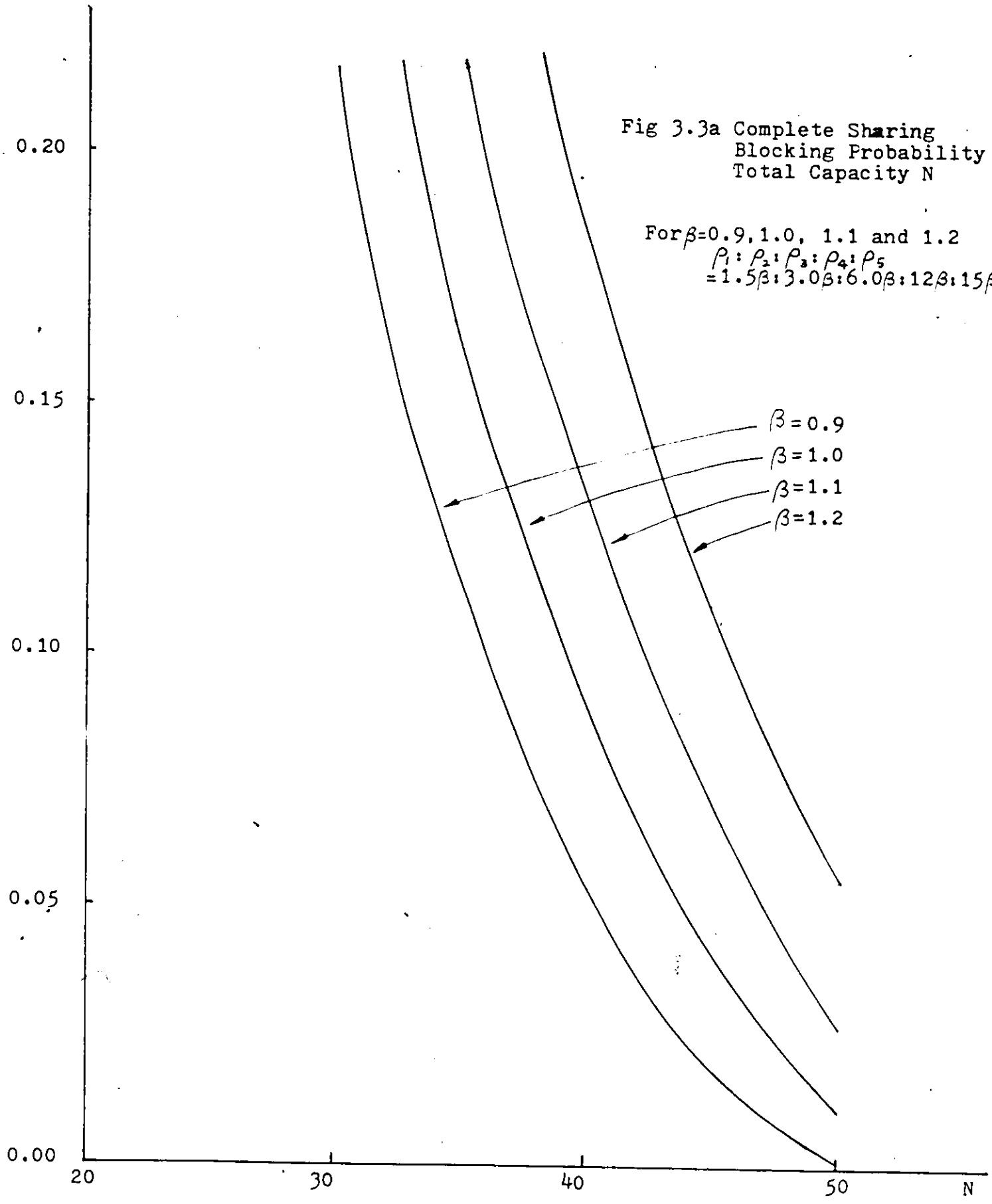


Fig 3.3a Complete Sharing Blocking Probability V's Total Capacity N

For β=0.9, 1.0, 1.1 and 1.2
$\rho_1 : \rho_2 : \rho_3 : \rho_4 : \rho_5$
$= 1.5\beta : 3.0\beta : 6.0\beta : 12\beta : 15\beta$

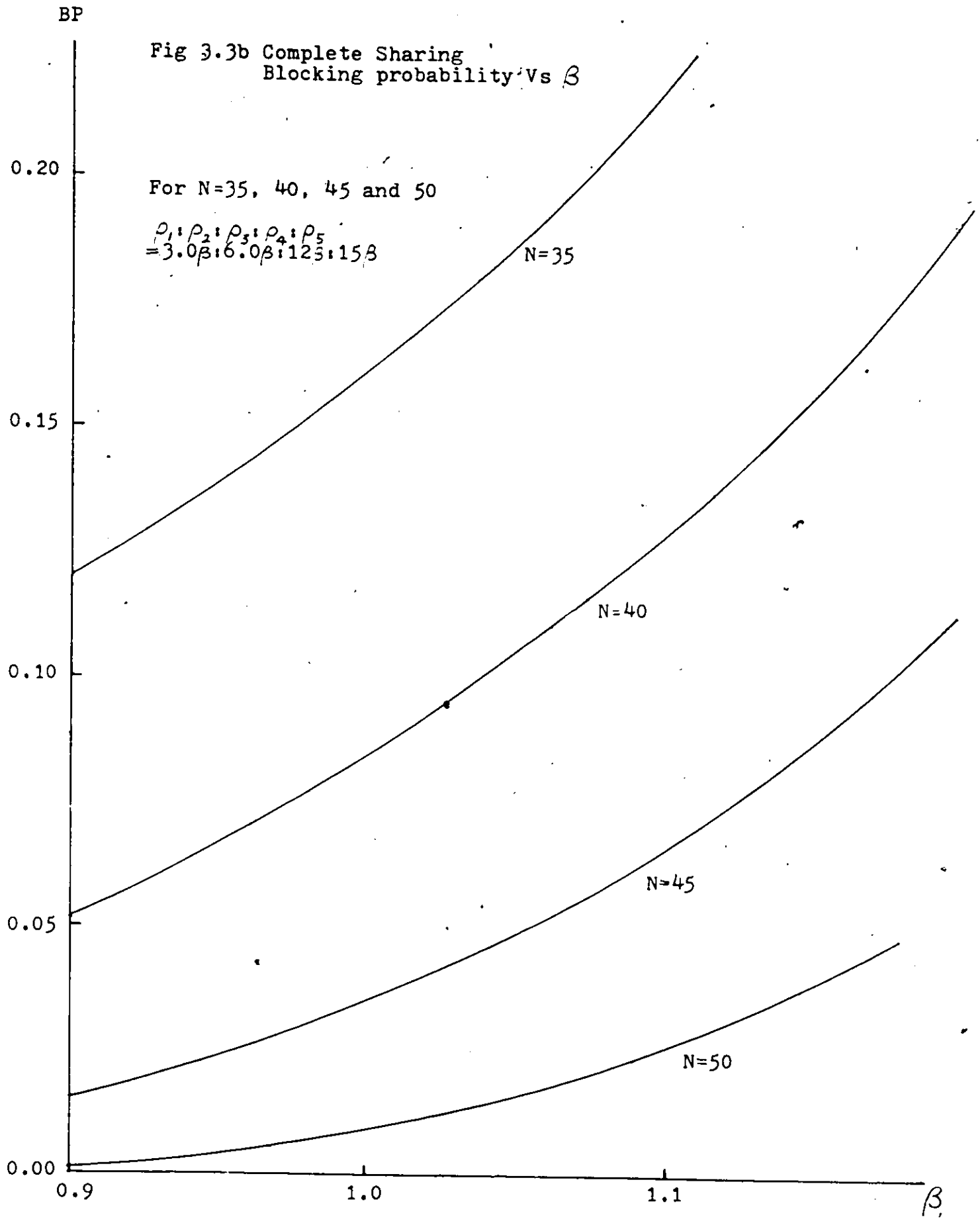


Fig 3.3b Complete Sharing
Blocking probability Vs β

For N=35, 40, 45 and 50

$\rho_1 : \rho_2 : \rho_3 : \rho_4 : \rho_5$
$= 3.0\beta : 6.0\beta : 12\beta : 15\beta$

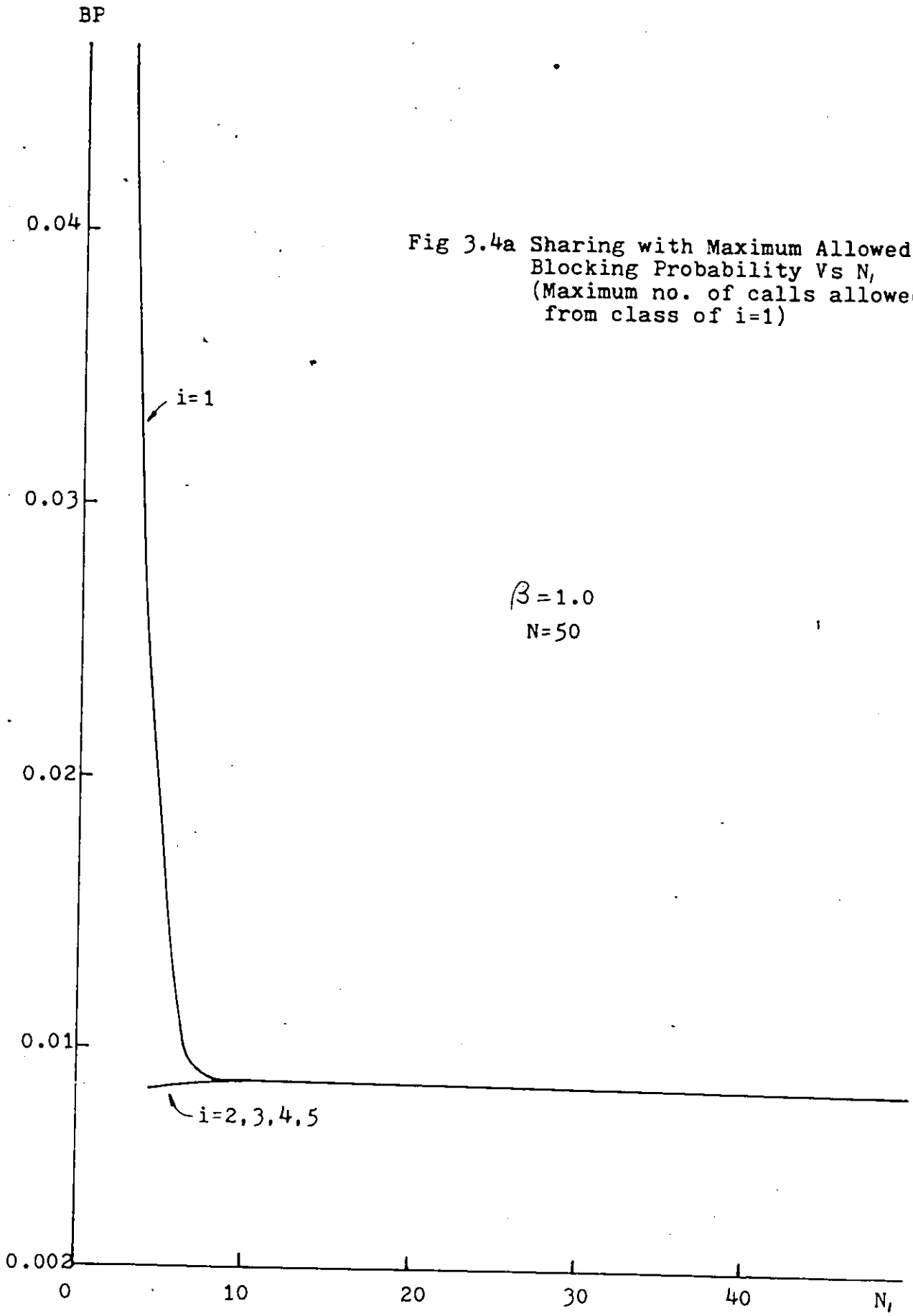


Fig 3.4a Sharing with Maximum Allowed - Blocking Probability Vs N_i (Maximum no. of calls allowed from class of i=1)

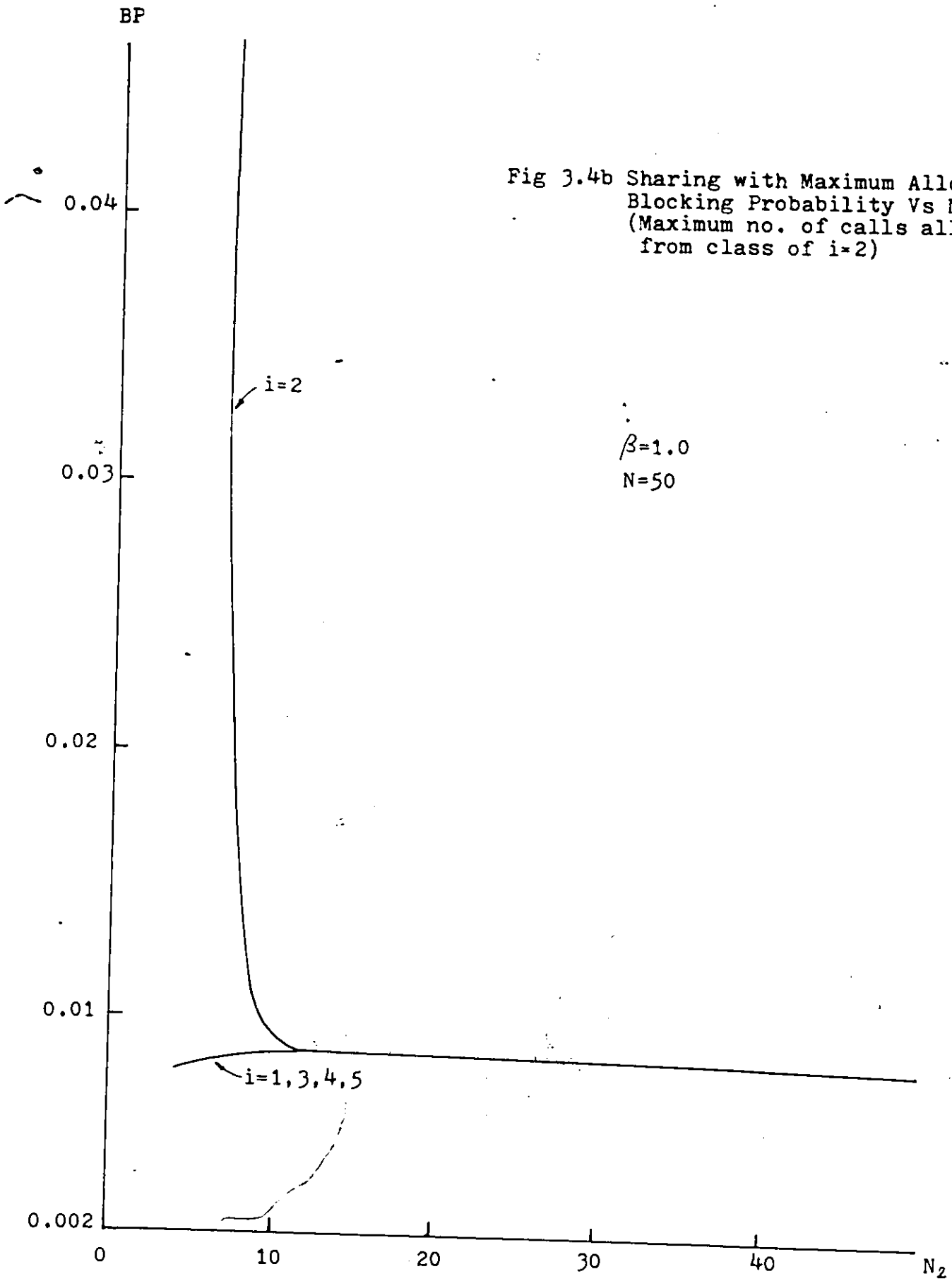


Fig 3.4b Sharing with Maximum Allowed Blocking Probability Vs N_2 (Maximum no. of calls allowed from class of i=2)

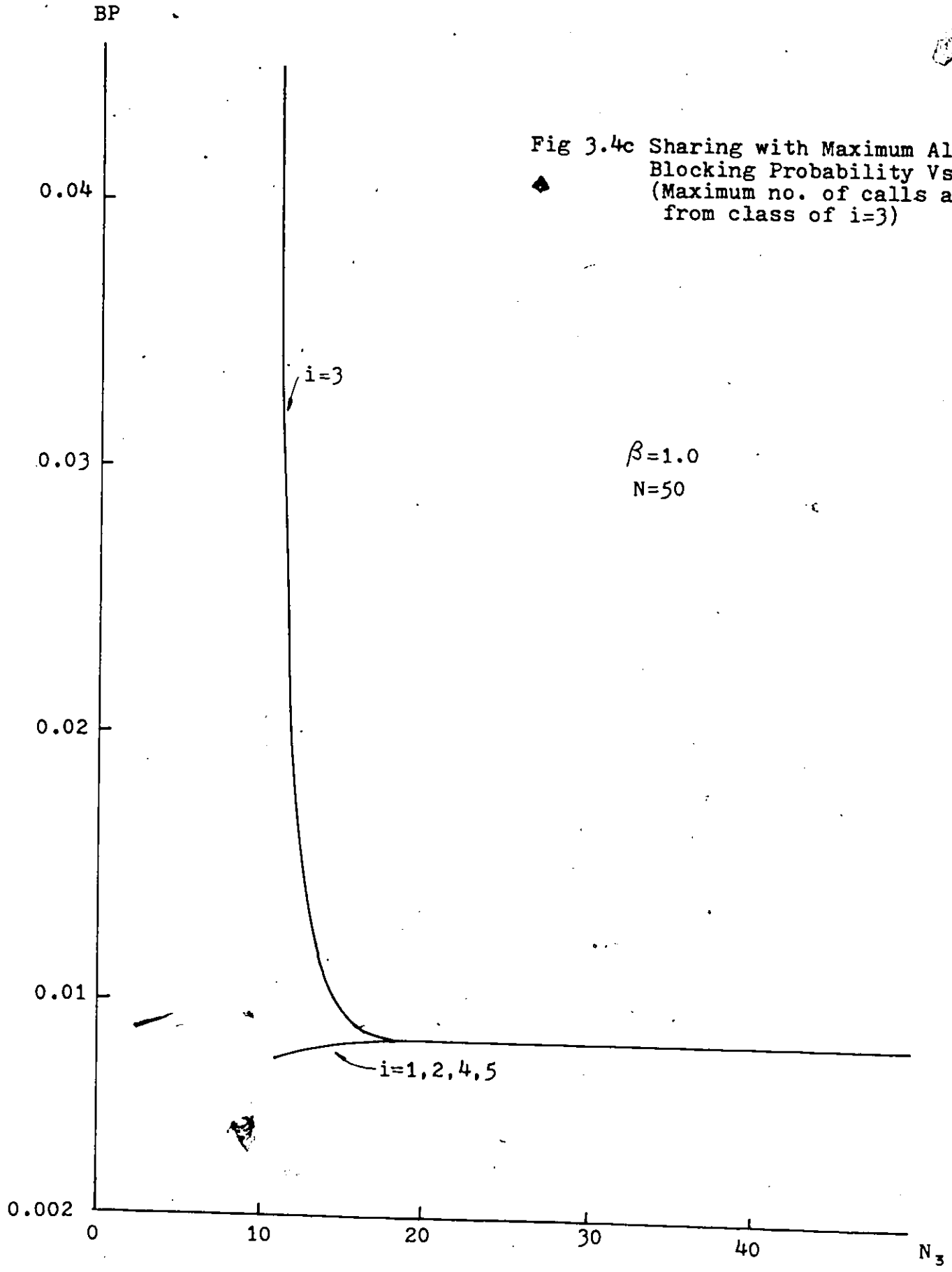


Fig 3.4c Sharing with Maximum Allowed Blocking Probability Vs N_3 (Maximum no. of calls allowed from class of i=3)

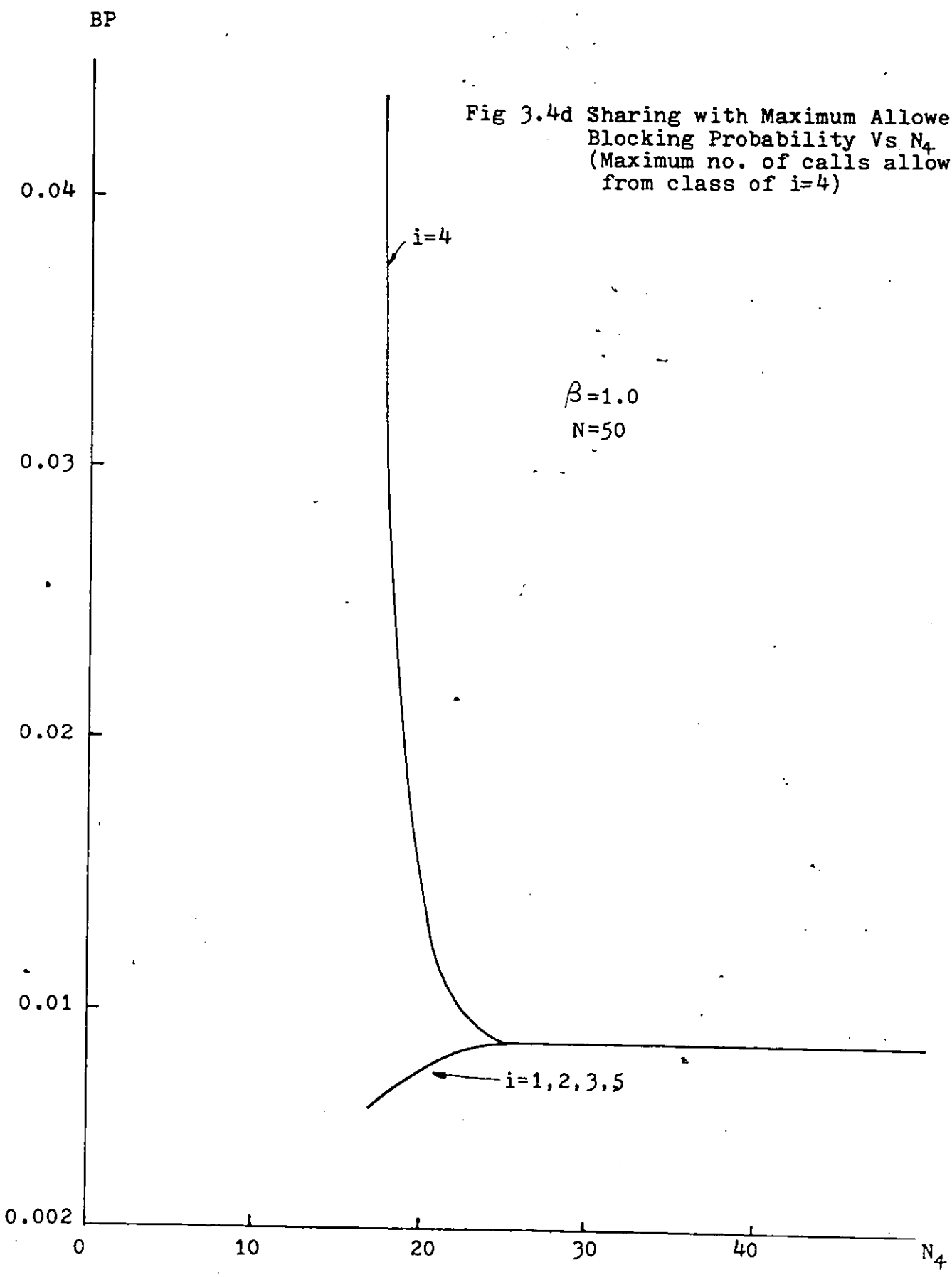


Fig 3.4d Sharing with Maximum Allowed Blocking Probability Vs N_4 (Maximum no. of calls allowed from class of i=4)

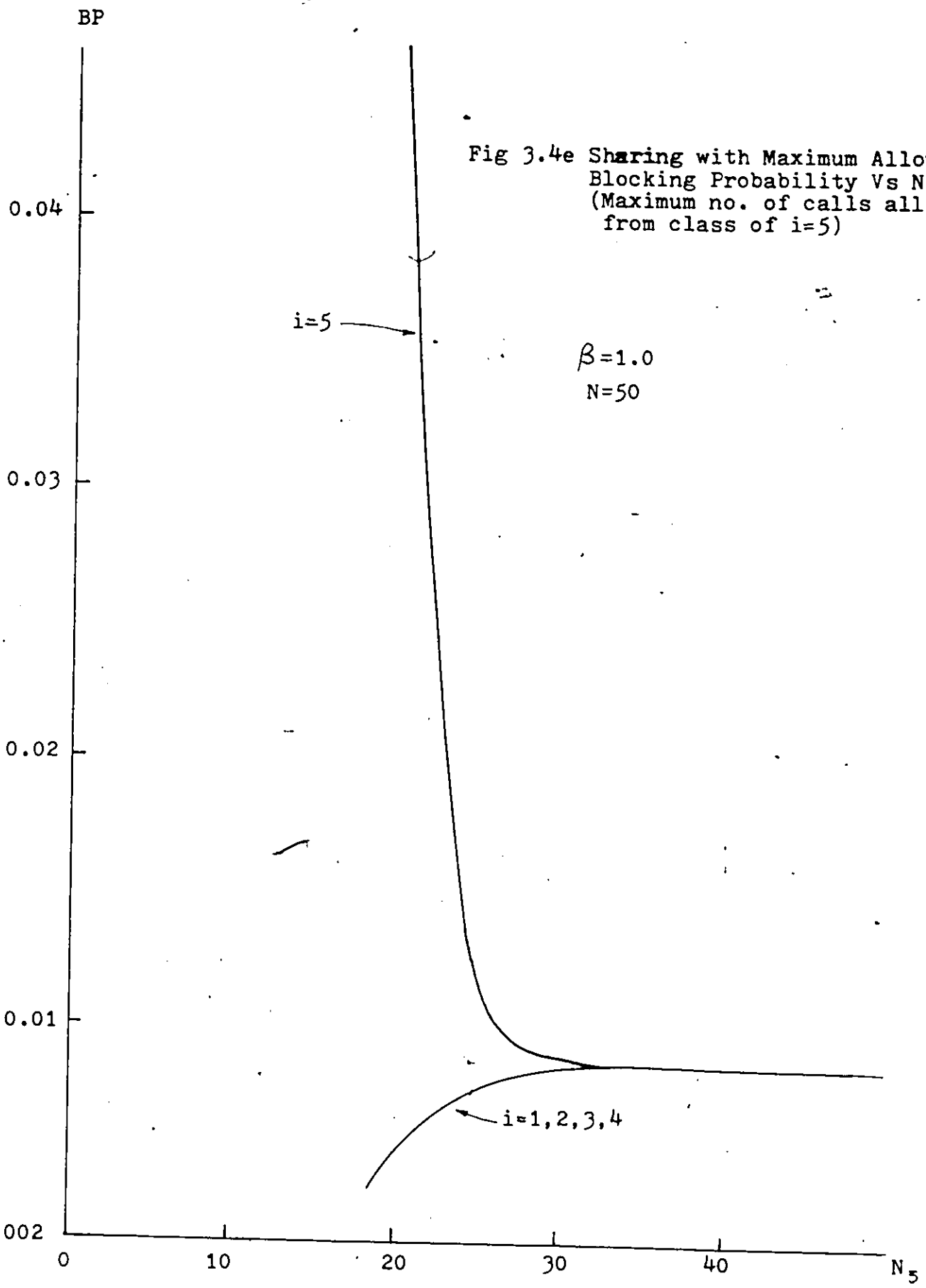
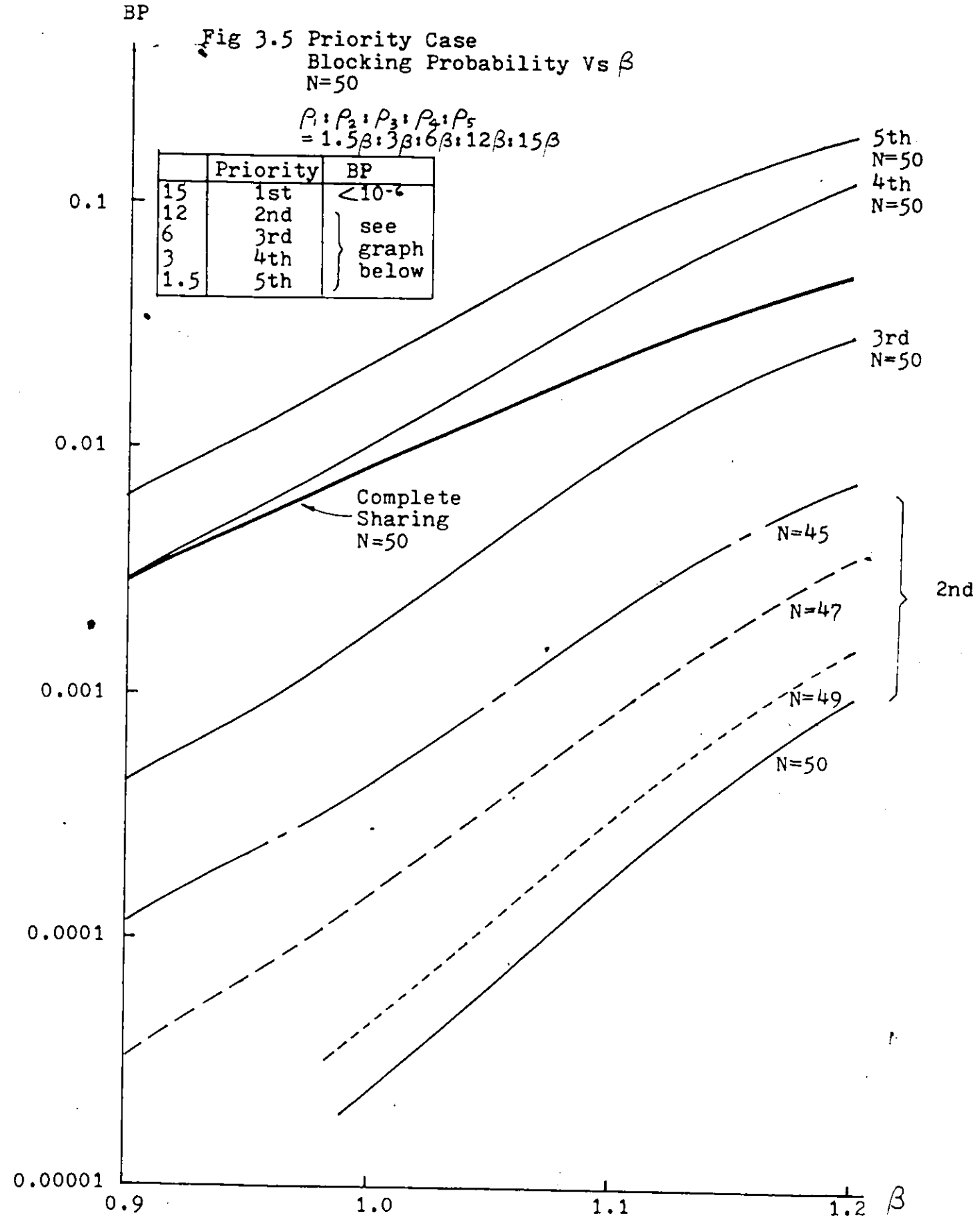


Fig 3.4e Sharing with Maximum Allowed Blocking Probability Vs N_5 (Maximum no. of calls allowed from class of i=5)

Fig 3.5 Priority Case
Blocking Probability Vs β
N=50

$\rho_1 : \rho_2 : \rho_3 : \rho_4 : \rho_5$
$= 1.5\beta : 3\beta : 6\beta : 12\beta : 15\beta$

	Priority	BP
15	1st	$<10^{-6}$
12	2nd	see graph below
6	3rd	
3	4th	
1.5	5th	



3.3.4 SYSTEM UTILIZATION

From the point of view of the system owner, utilization is the main factor in a design problem. Fig 3.6 compares the system utilization for the cases of Complete Sharing, Sharing with Maximum Allowed (for N_5=25) and Complete Partitioning. In Complete Partitioning, the system was divided into 5 subsystems, the capacity ratio of these systems was made to be the same as that of the traffic intensities while keeping the sum of capacity N to be 50. Obviously, this strategy gives a poorer utilization compared with that of the Complete Sharing and Sharing with Maximum Allowed desciplines.

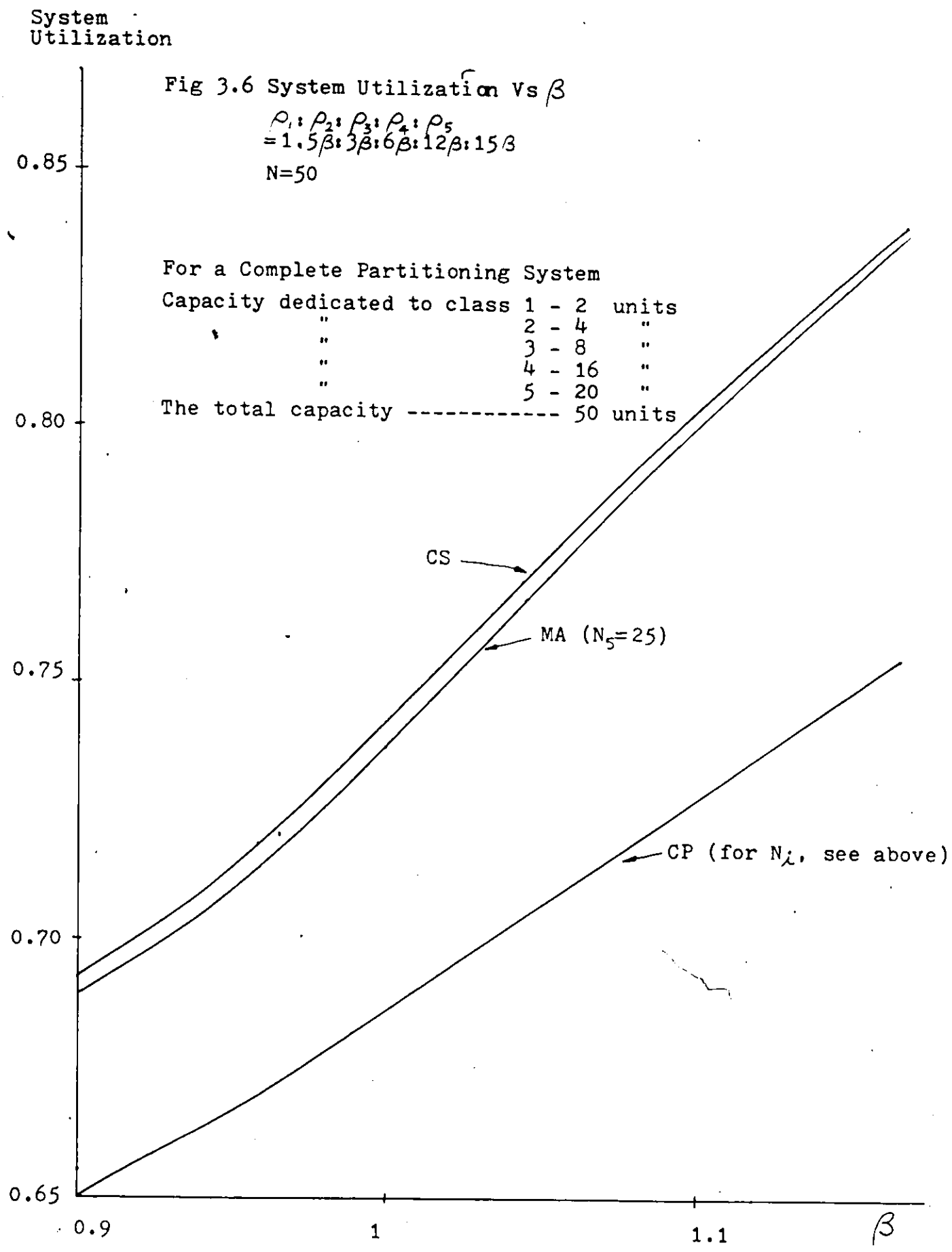


Fig 3.6 System Utilization Vs β

$\rho_1 : \rho_2 : \rho_3 : \rho_4 : \rho_5$
$= 1.5\beta : 3\beta : 6\beta : 12\beta : 15\beta$
N=50

For a Complete Partitioning System

Capacity dedicated to class	1 - 2	units
"	2 - 4	"
"	3 - 8	"
"	4 - 16	"
"	5 - 20	"
The total capacity ----------	50	units

Chapter IV

SYSTEM OPTIMIZATION

In the two preceeding chapters, a Demand Assignment Multiple Access (DAMA) system was considered. Various sharing policies that possess the coordinate convex property, leading to the well known product form solution (3.1) were described. In addition, explicit analytical expressions, as well as efficient algorithms for evaluating the DAMA system performance, were presented.

In a system design problem, however, the question posed to the designer is : given a DAMA system with N units of capacity to be shared by M groups of users, how should the resources be shared in order to fulfill a set of optimization criteria? These criteria can be the maximizations of throughput, utilization or minimization of blocking probability. Moreover, in order to represent the relative importance of different user groups, weighting factors can be introduced so that optimization can be done with respect to weighted blocking, weighted utilization or weighted throughput. An example is that the system owner charges different user groups according to different fee schedules.

This chapter deals with the optimization of a given practical system : The choice of a sharing policy is discussed in section 4.1; determination of the optimal set of parameters in that chosen policy is discussed in section 4.2 . Finally, numerical examples are given in section 4.3 .

4.1 OPTIMAL POLICY

For a DAMA system using any one of the six sharing policies described in Chapter 2, the steady-state probability $P_{\vec{i}}$ has the product-form solution, (3.1) which is repeated here :

$$P(\vec{n}) = P(\vec{0}) \prod_{i=1}^{M} V_i(n_i) \qquad , \vec{n} \in A \qquad (4.1)$$

where $\vec{n} = (n_1, \ldots, n_M)$

$n_i \triangleq$ number of ith group users in the system.

$P(\vec{n}) \triangleq$ Steady state probability

$$V_i(n_i) = \frac{\rho_i^{n_i}}{n_i!} \qquad \text{for an Erlang Model}$$

or $$V_i(n_i) = \binom{R_i}{n_i} \rho_i^{n_i} \qquad \text{for an Engset Model}$$

$$\rho_i = \lambda_i / u_i$$

$A \triangleq$ set of feasible states, depending on the system and policy used.

Using any one of the recursive algorithms presented in the previous chapter, a system's performance can be determined.

There are various measures of performance for each policy. The commonly used ones are throughput, utilization and blocking.

1. Weighted Blocking

The blocking probability of the ith group users is B_i ; their weighted sum can be used to find the blocking in the system

i.e. $\sum_{i=1}^{M} B_i W_i$

The weighting factor, W_i, serves to emphasize the blocking of each type differently. To optimize the system with respect to weighted blocking, the above sum should be minimized.

2. Weighted Throughput

Throughputs can be expressed in terms of blocking, B_i, in the following way

$$T_i = \lambda_i (1 - B_i)$$

The system weighted throughput is

$$\sum_{i=1}^{M} W_i T_i = \sum_{i=1}^{M} \lambda_i W_i (1 - B_i)$$

To maximize the system weighted throughput corresponds to minimizing the sum $\sum_{i=1}^{M} \lambda_i W_i B_i$

3. Weighted Utilization

The mean number of users from group i, $E\{n_i\}$ is

$$E\{n_i\} = \rho_i(1-B_i)$$

The mean number of capacity units used by group i users is

$$K_i E\{n_i\} = K_i \rho_i (1-B_i)$$

Therefore the total weighted utilization of the system is

$$U = \frac{1}{N}\sum_{i=1}^{M} W_i K_i \rho_i (1-B_i)$$

Where N is the total number of units available in the system. To maximize the weighted utilization, the sum $\sum_{i=1}^{M} W_i K_i \rho_i B_i$ is to be minimized.

In summary, the optimization can be done by using any one of the following criteria:

Minimize weighted blocking : minimize $\sum_{i=1}^{M} W_i B_i$

Maximize weighted throughput : minimize $\sum_{i=1}^{M} W_i \lambda_i B_i$

Maximize weighted utilization: minimize $\sum_{i=1}^{M} K_i \rho_i W_i B_i$

All three procedures can be reduced to the minimization of a sum in the form of $\sum C_i B_i$ where C_i are weighting factors.

The optimal policy will be dependent on the weights W_i. Also, for each set of W_i's, optimizing the throughput may not give the optimal blocking nor optimal utilization. However, it can be proved [7] that in the case of M=2 and

M=3, the structure of the optimal policies, using any one of the three criteria, is the same. In particular the optimal policy is that of "Sharing with Maximum Allowed", with the maximum number of the ith group users, N_i, depending on the the weights W_i and the optimization criterion. Therefore, any optimal policy may be implemented by limiting the number of each group to N_i. The proofs for the two dimensional and three dimensional case are beyond the scope of this presentation (see Ref. 7). Fig 4.1 and 4.2 show the optimal policy for the two dimensional and three dimensional cases respectively.

With the policy of "Sharing with Maximum Allowed" proven to be optimal for M=2 & 3 amongst the six sharing policies described in chapter 2, the first task of our system design is accomplished. What remains as the second and last task is the numerical evaluation of the set of optimal system parameters, i.e. the maximum number of ith group users, N_i. This is the subject of the following section.

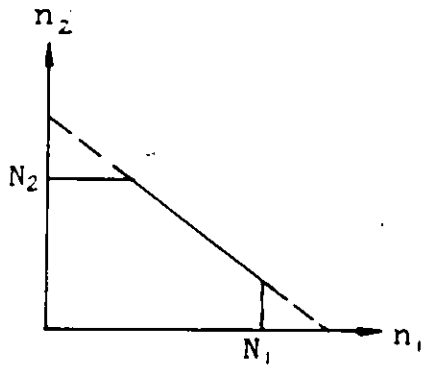


Fig 4.1 Optimal policy for a Two-Dimensional System

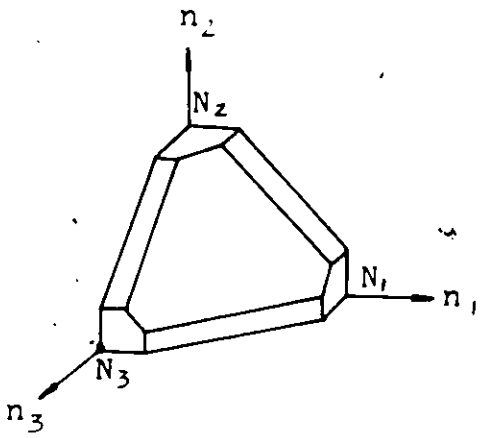


Fig 4.2 Optimal Policy for a Three-Dimensional System

4.2 PATTERN SEARCH TECHNIQUE

Since the policy of "Sharing with Maximum Allowed " is theoretically determined to be optimal (for M=2 & 3), what is left in a system design problem is to numerically determine the maximum number of users from each group that gives the best weighted performance. This is to be done with respect to one of the optimization criteria: weighting blocking, throughput or utilization.

As indicated in section 4.1, the optimization involves minimizing a sum in the form of $\sum_{i=1}^{M} C_i B_i$. C_i being a weighting factor for group i. (3.10) shows the expression for evaluating B_i for this policy. In general, the goal is to minimize the objective function $f(N_1, N_2, \ldots N_M) = \sum_{i=1}^{M} C_i B_i$ (B_i is function of N_i's) and determine the corresponding set $(N_1, N_2, \ldots, N_M)$.

The pattern search technique [8] was chosen for this task for the following reasons:

(a) It is well adapted to a computer since our task tends to use repeated identical arithmetic operations.

(b) It is logicaly simple - which simplifies computer programming.

(c) It permits integral increments to be taken - suitable for this problem since all N_i's must be non-negative integers.

Pattern search is a direct search routine for minimizing a function $f(N_1, N_2, \ldots, N_M)$. The arguments N_i are varied one by one until the minimum of f is obtained. The pattern search routine determines the sequence of values for N_i's. An independent routine computes the functional values of f.

The successive values of N_i's can be interpreted as points in a M-dimensional space. The procedure of going from a given point to the following point is called a move. A move is termed a success if the value of f decreases, otherwise, it is a failure. The pattern search routine makes two types of moves. The first type is exploratory moves designed to acquire knowledge concerning the behaviour of the function f. At each move only the value of a single coordinate is changed. The second type of move is pattern moves, designed to utilize the information acquired in the exploratory moves. The point from which a pattern move is made is designated a base point.

To begin the pattern search procedure, an arbitrary point is chosen at which f is evaluated. This is called the initial base point, B_o. Exploratory moves are then conducted. A single coordinate of the point is varied, either increasing or decreasing the coordinate by a prescribed step size, to see whether a successful move can be made. If a success is obtained, the altered value of the coordinate is retained; otherwise, the original value is

restored. Such exploratory moves are made for each coordinate, and the final point reached becomes a new base point B_1. The first pattern move is then carried out by duplicating the combined moves from the previous base point. That is, all coordinates are changed by an amount equal to the difference between the present base point and the previous base point (i.e. from B_0 to B_1). The intuitive basis for this type of move is the presumption that whatever constituted a successful set of moves in the past is likely again to prove successful. That leads to P_1. From P_1 exploratory moves are then conducted in an attempt to further improve the results.

If the pattern move fails, that is, if no new base points can be found, the simplest way of continuing the search would be to begin over again from the most recently obtained base point with a series of exploratory moves, and thus establish a new pattern.

The method for terminating the search is also to be addressed. For any given step size, the search procedure will reach an impasse when - the pattern move having failed to determine a new base point - all the exploratory moves from the base point failed. To further continue the search, it is necessary to reduce the prescribed step size and repeat exploratory moves. The search is then terminated when a prescribed number of step size reductions is reached.

In the pattern search procedure, the major reduction in f is produced by the pattern move. Some reduction is made by the exploratory moves, but their primary function is to supply information for the improvement of the pattern move.

Pattern search has been proved to be particularly successful in the search of the $\{N_i\}$ set - here the problem is to locate minima on hypersurfaces which contain "sharp surfaces". An exact description of a pattern search routine is given in Appendix B.

4.3 NUMERICAL EXAMPLES

In order to illustrate how the preceeding concepts are applied to a practical system design problem, two examples are considered. Fochini and Gopinath proved [7] that for a system with up to 3 user groups contending for the resources, the policy of "Sharing with Maximum Allowed" gives the optimal results if any one of the three performance criteria, namely, weighted blocking, weighted throughput or weight utilization is used as optimization goal. These two researchers then conjectured, without proof, that this is also true for any arbitrary dimensions.

In the examples here, a four-dimensional system is considered. (The optimization package developed here can handle systems of any dimension, limited by, obviously, the available computing time, since the amount of computation

involved is proportional to the system dimension, M). In each case, the input parameters are as follows:

$N \triangleq$ number of capacity units available

$M \triangleq$ number of user groups

$K_i \triangleq$ number of capacity units required to serve each ith user

$\rho_i \triangleq$ traffic intensity of the ith group user

$w_i \triangleq$ weight factor of the ith group user

The procedure is as follows:-

1. input system parameters
2. input initial guess for the maximum number of the ith group users allowed $(N_1, \ldots, N_M)$
3. using (3.16) to construct an $N \times M$ matrix $S(n,m)$ so that $n=0,\ldots,N$, $m=0,\ldots,M$ (recall that $S(N,M)$ is the inverse of $P(\vec{0})$, the probability that the system is empty)
4. from (3.10), elaluate the values of the objective function
5. use pattern search to determine the direction of search for minimization of the objective function. That involves changing one of the coordinates in the vector $(N_1, \ldots, N_M)$ and repeat steps 3 and 4 until it is satisfied that the minimum has been found. The termination of this search is described in detail in section 4.2.

The output of step (5) is then a vector of "optimal" N_i's and the corresponding system weighted blocking, weighted throughput or weighted utilization.

Note that since the pattern search technique does not guarantee that the solution is the absolute minimum, the search should be repeated several times, each time starting with a different initial guess. If the results are similar (depending on the tolerance required, a study on the sensitivity of the objective function f was conducted and is presented at the end of this chaper), one can be reasonably confident that the outcome is the required solution.

Appendix C describes step (1) through (5) in a flow chart form.

4.3.1 OPTIMIZATION RESULTS

Two numerical examples are considered here, both are four-dimensional (M=4) with N=200. The first one has balanced traffic. By balanced traffic, we refer to the fact that the products $K_i\rho_i$ for all i are identical. The second one has non-balanced traffic, i.e. $K_i\rho_i$ are not equal for all i.

For each of the two examples, the optimal vector $(N_1,\ldots,N_4)$ and the corresponding system performance is determined. (with respect to each of the three criteria). The results are summerized below:-

Example 1:- N=200, M=4

Input

i	1	2	3	4
K_i	40	20	8	4
ρ_i	1.0	2.0	5.0	10.0
W_i	0.1	0.2	0.3	0.5

Output

Criterion	f	N_1	N_2	N_3	N_4
Weighted blocking	0.084	2	9	19	31
Weighted throughput	0.201	1	4	21	49
Weighted Utilization	4.930	2	8	18	30

f $\triangleq$ objective function

It is clear from the results that optimizing the system in term of weighted blocking does not give optimal results as far as weighted throughput or weighted utilization is concerned. To optimize weighted blocking, the $\{N_i\}$ vector should be (2,9,19,31) while if optimizing the weighted throughput is desired, the $\{N_i\}$ vector should be (1,4,21,49).

Example 2:- M=4, N=200

Input

i	1	2	3	4
K_i	40	16	16	4
ρ_i	0.1	1.0	5	10
W_i	0.9	0.05	0.01	0.04

Output

Criterion	f	N_1	N_2	N_3	N_4
Weighted blocking	0.0107	4	6	1	24
Weighted throughput	0.022	4	9	5	40
Weighted utilization	31.5	4	11	4	27

f $\triangleq$ objective function

Again, the same observation made in example 1 applies here. Therefore, in a practical design problem, when more than one criterion is deemed to be significant, a compromise is required. Judgement of this nature can only be passed on a per system basis.

Obviously, cross comparisons between e.g. 1 and 2 should not be made since they are two entirely different systems. As for computational efforts, for these particular examples, the package developed for this work used approximately 60 iterations for each optimization. In terms of CPU time, it required about 10 seconds for each optimization. The requirement increases with M and N.

4.3.2 SENSITIVITY STUDIES RESULTS

To conclude this numerical exercise, one question remains to be answered: how much tolerance in the resultant optimal N_i's do we need? This is an important question to ask since it determines how long the pattern search should be continued before a solution is declared acceptable.

Sensitivity studies of this type were conducted on both of the examples here based on two parameters, namely the objective function f with respect to the N_i and the objective function with respect to ρ_i:-

1. At each of the optimal points $[N_i]$, find the required change in each of the coordinates in order to produce, say, a 5% change in the objective function. e.g. take the weighted blocking case in example 1, the optimal $[N_i]$ is (2,9,19,31). To determine how sensitive f is to N_4, N_4 is varied from 31 until f is increased by 5%. It was found that N_4 can be changed by as much as 45% before the objective function increases by 5%. This is repeated for all i's. The results are summarized below.

Example 1:- Balanced Traffic

ΔN_i = required percentage change in N_i to give 5% change in f

Criterion	ΔN_1	ΔN_2	ΔN_3	ΔN_4
Weighted blocking	50%	56%	47%	45%
Weighted throughput	100%	25%	48%	60%
Weighted utilization	50%	75%	67%	47%

The results show that the allowed changes in N_i in this particular example range from 25 to 100% before the objective function increases by 5%. The indication here is that f is not very sensitive to $[N_i]$. Therefore, no significant advantage is gained by allowing the pattern search procedure to proceed too long once little improvement is observed in f, i.e., instead of using 60 iterations (as indicated in section 4.3.1) the search could have been terminated earlier without much degradation in the final results. This is significant when a large system is under consideration.

A similar observation can be made on the example 2 as indicated by the following results:-

Example 2:- Unbalanced Traffic

Criterion	ΔN_1	ΔN_2	ΔN_3	ΔN_4
Weighted blocking	25%	33%	100%	33%
Weighted throughput	50%	67%	20%	53%
Weighted utilization	75%	97%	75%	63%

2. The same procedure as that in 1 is repeated but instead of changing N_i, the traffic intensity ρ_i is changed so that at the optimal point $[N_i]$, the change in each of the ρ_i required in order to increase the f by 5% is determined. Again, the choice of 5% is arbitrary. In most systems, the traffic pattern is assumed to be known. This study shows how sensitive the system is in case of variation of this information.

Example 1:- Balanced Traffic

$\Delta\rho_i \triangleq$ required percentage changes in ρ_i to give a 5% change in f

Criterion	$\Delta\rho_1$	$\Delta\rho_2$	$\Delta\rho_3$	$\Delta\rho_4$
Weighted blocking	8%	6%	6%	6%
Weighted throughput	9%	4%	4%	4%
Weighted utilization	20%	40%	20%	12%

Example 2:- Unbalanced Traffic

$\Delta\rho_i \triangleq$ required percentage change in ρ_i to give a 5% change in f

Criterion	$\Delta\rho_1$	$\Delta\rho_2$	$\Delta\rho_3$	$\Delta\rho_4$
Weighted blocking	10%	10%	40%	5%
Weighted throughput	10%	10%	2%	5%
Weighted utilization	10%	30%	94%	20%

The results indicated that the allowed change in ρ_i's is again of relatively large amount althrough it

is not as much as that of N_i's. One can conclude that the systems under consideration in these two examples are considerably more sensitive to ρ_i than N_i. However, there is no indication that a general conclusion of this nature can be made on all DAMA system using the policy of sharing with Maximum Allowed.

Chapter V

CONCLUSIONS

A demand assignment multiple access system (DAMA) with a pool of resources shared by several user groups was considered here. Sharing policies of such resources that are simple to implement and require no detailed measurement of arriving traffic (i.e. accuracy of statistical parameters such as ρ is not important) are of considerable practical interest. Six of these policies, namely, Complete Sharing, Complete Partitioning, Sharing with Maximum Allowed, Sharing with Minimum Allocated, Intermediate Sharing and Preemptive Priority Sharing were studied in detail. All of these policies possess the property of coordinate convexity, giving rise to the well known product form solution for the steady-state probabilities.

To allow use of these theoretical results in systems of realistic size, efficient algorithms were presented. From that, commonly considered system performance measures can be determined with a reasonable amount of effort. Moreover, the contribution of the two-dimensional algorithm was shown to be more significant when it was adapted in system designs.

In a system design problem, the designer is required to choose a sharing policy and then find the optimal system parameters numerically. For such optimization, three of the commonly used criteria, namely, weighted blocking, weighted throughput and weighted utilization were considered. Theoretical results have been presented in the literature indicating that, for two and three dimensional systems, the optimal policy for ~~any of the~~ three criteria is that of Sharing with Maximum Allowed. For the determination of the maximum number of users allowed, the pattern search technique was presented. By adopting the pattern search procedure, in conjunction with the two dimensional recursive algorithm, an optimization computer package was developed. Using this package, the optimal results of any such DAMA system of any dimension can be obtained.

Finally, two numerical examples were given. In each case, the optimization was repeated for each of the three criteria discussed. Sensitivity studies were also carried out to determine how sensitive the objective function were with respect to both the variation of N_i from its optimal value and to the variation of ρ_i. It was concluded that for these two examples they are quite insensitive.

It is hoped that these results will be useful to the designers of DAMA Satellite Systems, such as the Canadian Mobile Satellite (MSAT).

APPENDIX A[1] - RECURSIVE EVALUATION OF S(N,M)

We remember that

$$S(N,M) = \sum_{A} \prod_{i=1}^{M} V_i(n_i) \qquad (1)$$

and

$$A = \left\{ n : n\ (n_1,\ldots,n_M);\ \sum_{i=1} K_i n_i \leq N \text{ and } n_i \leq N_i,\ i=1,\ldots,M \right\} \qquad (2)$$

The following quantity has to be defined:

$$G(n,m) = \sum_{A'} \prod_{i=1}^{m} V_i(n_i) \qquad (3)$$

where

$$A' = \left\{ n : n\ (n_1,\ldots,n_M), \sum_{i=1}^{m} K_i n_i = n \text{ and } n_i \leq N \right\} \qquad (4)$$

Since $A = U_{l=0}^{m} A'$ and the A' sets are disjoint, the following holds

$$S(n,m) = \sum_{l=0}^{n} G(l,m) \qquad (5)$$

It is easily shown that

$$G(n,m) = \sum_{h=0}^{b_m(n)} G(n-K_m h, m-1) \cdot V_m(h) \qquad (6)$$

with

$$b_m(n) = \min\left(N_m, \left[\frac{n}{K_m}\right]\right) \qquad (7)$$

In order to show (6) we define

$$Q_i(z) = \sum_{h=0}^{N_i} V_i(h) z^h \qquad (8)$$

and

$$G_m(z) = \prod_{i=1}^{m} Q_i(z^{K_i}) \qquad (9)$$

In a way similar to (9) it is not difficult to see that

$$G(n,m) = \frac{1}{n!} \frac{\partial^n}{\partial z^n} G_m(z) \Big|_{z=0} \qquad (10)$$

(1) Quoted from [1], page 1754.

that is, $G(n,m)$ is the coefficient of the term z^n in $G_m(z)$; in fact, all the factors such that $\sum_{i=1}^{m} n_i \cdot K_i = n$ (and $n_i \leq N_i$) contribute to this term. The $G_m(z)$ is the z-transform of the sequence $G(n,m)$. Analytically,

$$G_m(z) = \sum_{n=0}^{\infty} G(n,m) z^n \tag{11}$$

Note that $G(n,m)=0$ if $n > \sum_{i=1}^{m} N_i K_i$. From (9), the following equation holds:

$$G_m(z) = G_{m-1}(z) Q_m(z^{K_m}) \tag{12}$$

Performing the inverse z-transform of (12), (6) is obtained.

Now, we show that $S(n,m)$ satisfies a recursive relation similar to (6). From (5) and (6)

$$S(n,m) = \sum_{l=0}^{n} \sum_{h=0}^{b_m(l)} G(l-K_m h,\ m-1) V_m(h) \tag{13}$$

Taking into account that

$$G(l,\ m-1) = 0 \quad \text{if } l<0 \tag{14}$$

we have

$$S(n,m) = \sum_{l=0}^{n} \sum_{h=0}^{b_m(n)} G(l-K_m h, m-1) V_m(h) \tag{15}$$

Now the exchange of the summation indexes is possible. Then letting $r = l - K_m h$ and remembering (14)

$$S(n,m) = \sum_{h=0}^{b_m(h)} \left(\sum_{r=0}^{n-K_m h} G(r, m-1) \right) V_m(h) \tag{16}$$

Finally from (5)

$$S(n,m) = \sum_{h=0}^{b_m(n)} S(n-K_m h, m-1) V_m(h) \tag{17}$$

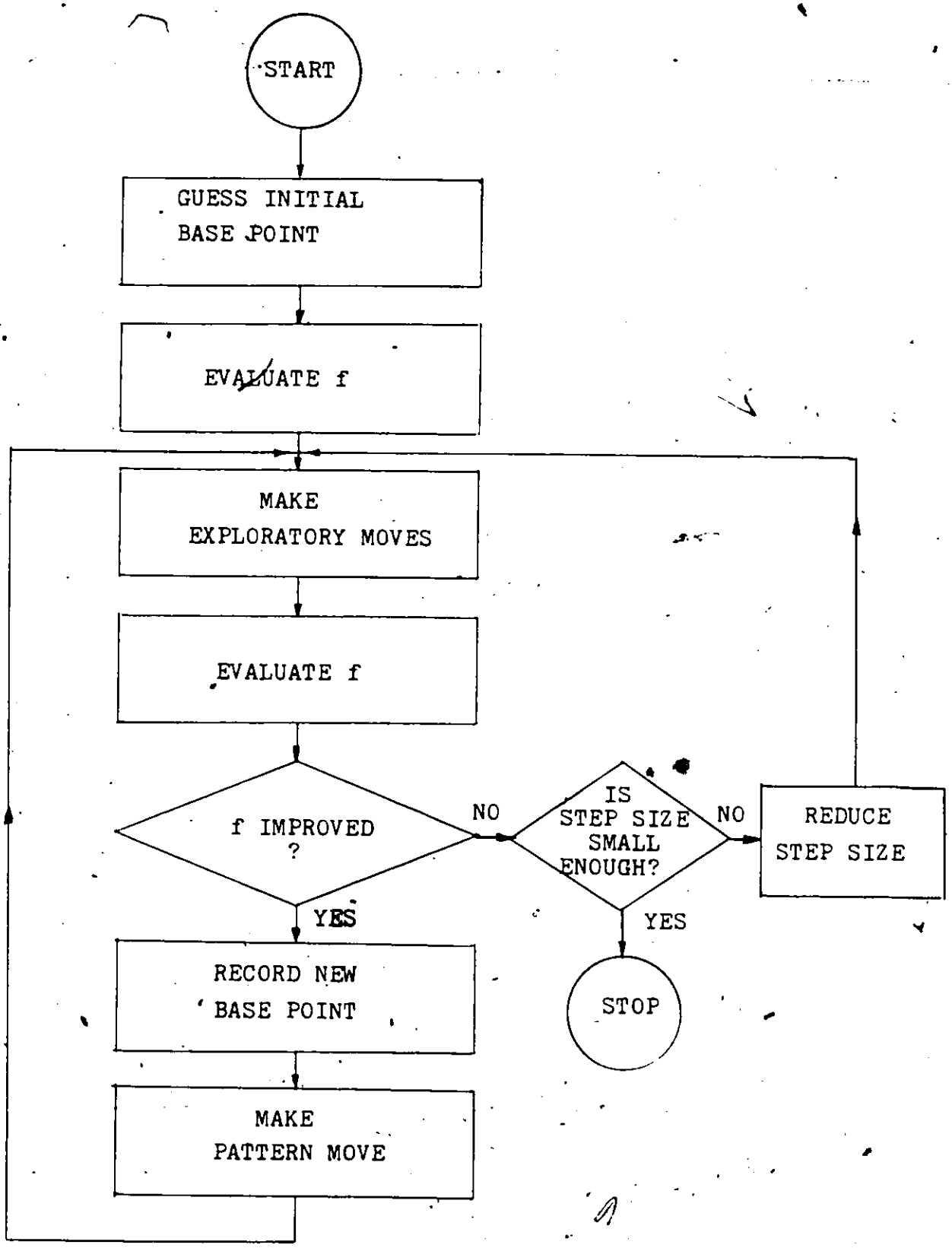
The above equation is the recursive formula that allows a fast computation of the normalization constant $S(N,M)$.

The intermediate results $S(n;m)(n<N, m<M)$ are the normalization constants of reduced systems with only n ECU's and m groups. For a numerical evaluation of (12), $S(n,0)$ and $S(o,m)$ must be specified.

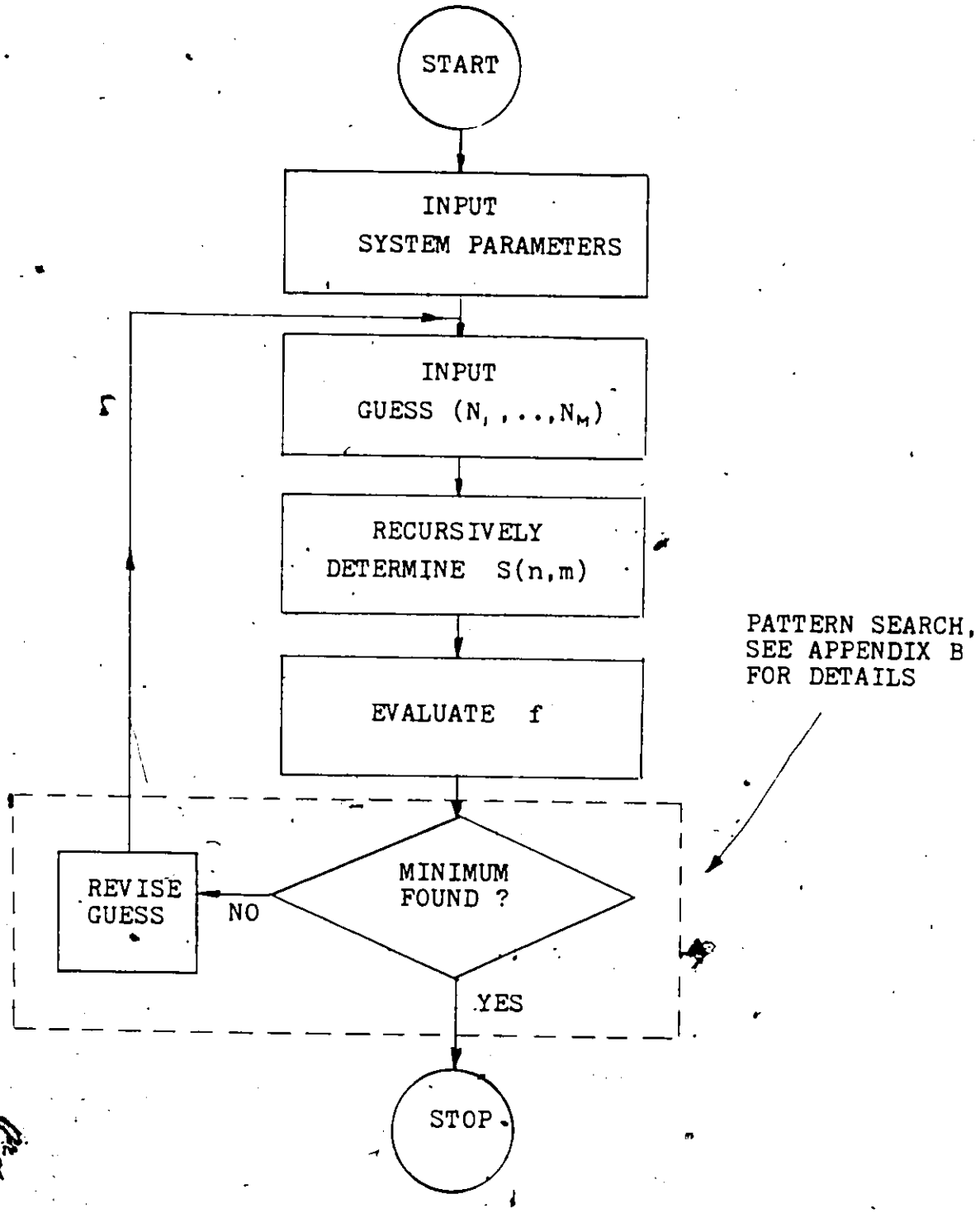
Remembering the physical meaning of such values, the following initial conditions hold:

$$S(n,0)=1 \qquad n=0,..,N \tag{18}$$
$$S(0,m)=1 \qquad m=1,..,M$$

APPENDIX B - FLOW CHART FOR PATTERN SEARCH



APPENDIX C - OPTIMIZATION PROCEDURE



APPENDIX D - COMPUTER PROGRAM

(1) **Optimization of DAMA Systems**
Sharing Policy : Sharing with Maximum Allowed
Criterion : Minimize Weighted Blocking

```
      DIMENSION R(10),C(10),NX(10),NXB(10),NXBC(10),NXB(10),NXC(10),
     +V(300,10),NBP(10),S(300,2),PPR(10),W(10)
      INTEGER R,PP,C,HALF,HALF2,HALV1,HALV2,HALV3,BBC,BBOF,PP,PP
C
C     INPUT SYSTEM PARAMETERS
C
      READ ,(3,10)MM,NN
      WRITE (6,10)MM,NN
10    FORMAT(2I5)
      READ (3,11)(K(M),M=1,MM)
      WRITE (6,11)(K(M),M=1,MM)
11    FORMAT(10I5)
      READ (3,12)(C(M),M=1,MM)
      WRITE (6,12)(C(M),M=1,MM)
12    FORMAT(10F7.2)
      READ (3,12)(W(M),M=1,MM)
      WRITE(6,12)(W(M),M=1,MM)
      READ(3,14)(NX(M),M=1,MM)
      WRITE (6,14)(NX(M),M=1,MM)
      READ(3,14)(NXB(M),M=1,MM)
      WRITE (6,14)(NXB(M),M=1,MM)
14    FORMAT(10I5)
      READ(3,13)HALF,HALF2,BBOF
      WRITE (6,13)HALF,HALF2,BBOF
13    FORMAT(3I5)
      IT=0
      HALV3=0
      CALL OBJ(MM,NN,K,C,W,NX,IT,PPR,F)
      FO=F
121   DO 1020 M=1,MM
      NXP(M)=0
      NXBO(M)=NXB(M)
      NXO(M)=NN/K(M)
1020  CONTINUE
      BPO=0
      PM=0
      BP=0
106   HALV1=0
C
C     STOP AL IF HALF IS EXCEEDED
C
105   IF(HALV1.GT.HALF) GOTO 107
      BPO=BPO+1
      IF(BBC.GT.BBOF)GOTO 120
```

```
C
C       LOCATE INITIAL BASE POINT
C
        IN=0
        BPO=BPO+1
C       WRITE(6,30)
30      FORMAT(/'**********************************************')
C       WRITE (6,16)BPO
18      FORMAT('**** BPO = ',I5)
        DO 1021 M=1,MM
C       WRITE(6,20)M
20      FORMAT(10X,'M = ',I5/)
        IF(NXB(M).EQ.0)GOTO 109
        NX(M)=NX(M)-NXB(M)
        IF(NX(M).LT.0)GOTO 117
        CALL OBJ(MM,NN,M,O,P,NX,IT,FMN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)+NXB(M)
        IF(FO.GE.F)GOTO 1021
117     NX(M)=NX(M)+2*NXB(M)
        IF(NX(M).GT.NXO(M))GOTO 108
        CALL OBJ(MM,NN,M,O,M,NX,IT,FMN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)-NXB(M)
        IF(FO.GE.F)GOTO 1021
108     NX(M)=NX(M)-NXB(M)
109     IN=IN+1
1021    CONTINUE
C
C       IF INITIAL BASE POINT IS POINT,PROCEED WITH PATTERN MOVE
C
        IF(IN.LT.MM) GOTO 100
C
C       ELSE REDUCE STEP SIZE AND TRY AGAIN
C
        DO 1025 M=1,MM
        NXB(M)=NXB(M)/2
1025    CONTINUE
        HALV1=HALV1+1
        GOTO 105
C
C       RECORD CURRENT BP
C
100     DO 1029 M=1,MM
        NBP(M)=NX(M)
1029    CONTINUE
C
C       PROCEED WITH PATTERN MOVE
C
        PM=PM+1
C       WRITE (6,16)PM
16      FORMAT(5X,'**** PM = ',I5/)
        INDE=0
        DO 1022 M=1,MM
        NX(M)=NX(M)-NXB(M)
        IF((NX(M).GT.NXO(M)).OR.(NX(M).LT.0))INDE=1
1022    CONTINUE
```

```
C       WRITE(6,29)(NX(M),M=1,MM)
29      FORMAT(5I5)
C
C       RESET STEP SIZE TO INITIAL VALUE
C
        DO 1027 M=1,MM
        NXB(M)=NXEC(M)
1027    CONTINUE
C
C       IF PM MOVED NX OUT OF BOUNDS, DISCARD PM
C
        IF(INDE.EQ.1)GOTO 116
C
C       ELSE LOCATE NEW BP
C
        BP=0
        HALV2=0
103     IND=0
        BP=BP+1
C       WRITE(6,19)BP
19      FORMAT(10X,'**** BP = ',I5/)
        DO 1023 M=1,MM
C       WRITE(6,21)M
21      FORMAT(10X,'M = ',I5/)
        IF(NXB(M).EQ.0)GOTO 111
        NX(M)=NX(M)-NXB(M)
        IF(NX(M).LE.0)GOTO 116
        INDEX=0
        DO 1032 I=1,MM
        IF(NX(I).EQ.NBE(I))INDEX=INDEX+1
1032    CONTINUE
        IF(INDEX.EQ.MM)GOTO 118
        CALL OBJ(MM,NN,K,D,W,NX,IT,FMM,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)+NXB(M)
        IF(FO.GE.F)GOTO 1023
118     NX(M)=NX(M)+2*NXB(M)
        IF(NX(M).GT.NXO(M))GOTO 110
        INDEX=0
        DO 1033 I=1,MM
        IF(NX(I).EQ.NBE(I))INDEX=INDEX+1
1033    CONTINUE
        IF(INDEX.EQ.MM)GOTO 110
        CALL OBJ(MM,NN,K,D,W,NX,IT,FMM,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)-NXB(M)
        IF(FO.GE.F)GOTO 1023
110     NX(M)=NX(M)-NXB(M)
111     IND=IND+1
1023    CONTINUE
C
C       IF NEW BP IS FOUND, PROCEED WITH PATTERN MOVE
C
```

```
      IF(IND.LT.MM)GOTO 100
C
C     ELSE REDUCE STEP SIZE AND TRY AGAIN
C
      HALV2=HALV2+1
      IF(HALV2.GT.HALM2)GOTO 104
      DO 1024 M=1,MM
      NXB(M)=NXB(M)/2
1024  CONTINUE
      GOTO 103
C
C     IF NEW BP IS NOT FOUND AFTER STEP SIZE IS REDUCED REPEATEDLY,
C     DISCARD LAST PATTERN MOVE, RESTORE INITIAL STEP SIZE AND
C     RETURN TO PREVIOUS BP
C
104   DO 1025 M=1,MM
      NXB(M)=NXBO(M)
1025  CONTINUE
116   DO 1026 M=1,MM
      NX(M)=NX(M)+NXB(M)
      NXP(M)=0
1026  CONTINUE
C
C     RE INITIATE PATTERN SEARCH
C
      GOTO 106
C
C     RE INITIATE PATTERN SEARCH WITH REDUCED STEP SIZE
C
120   HALV3=HALV3+1
      IF(HALV3.GT.HALM)GOTO 107
      DO 1035 M=1,MM
      NXB(M)=NXB(M)/2
1035  CONTINUE
      GOTO 121
C
C     RECORD OPTIMIZATION RESULT
C
107   WRITE (6,15)
15    FORMAT(//5X,'******** PATTERN SEARCH ENDS ********'/)
      CALL OBJ(MM,NA,M,Q,W,NX,IT,FMN,F)
      WRITE(6,32)IT
32    FORMAT('NO. OF ITERATIONS = ',I5/)
      DO 1034 M=1,MM
      WRITE(6,33)M,FMN(M),M,NX(M),M,NXB(M)
33    FORMAT(5X,'FMN(',I2,')=',F6.6,5X,'NX(',I2,')=',I5,
     +7X,'NXB(',I2,')=',I5)
1034  CONTINUE
      WRITE(6,34)F
34    FORMAT(/5X,'OBJECTIVE FUNCTION = ',F8.3/)
      STOP
      END
```

```
C
C
C
C     ***************************************************************
C
C     SUBROUTINE TO DETERMINE VALUE OF OBJECTIVE FUNCTION
C
C     ***************************************************************
C
      SUBROUTINE OBJ(MM,NN,M,C,K,NX,IT,FUN,F)
C
C
      DIMENSION K(10),Q(10),NX(10),NXE(10),NXE(10),
     +V(300,10),S(300,2),FMN(10),N(10)
      INTEGER M,EM,C,FALM,FALV1,FALV2
      IT=IT+1
C     WRITE(6,28)IT
28    FORMAT(/'ITERATION ',I3/)
C     WRITE(6,91)(NX(M),M=1,MM)
91    FORMAT(4015)
C
C     FIND V(M,M)
C
      DO 1040 M=1,MM
      N3=NX(M)+1
      V(1,M)=1.0
      DO 1041 M=2,N3
      V(M,M)=1.0
      N4=M-1
      DO 1042 I=1,N4
      V(M,M)=V(M,M)*Q(M)/I
1042  CONTINUE
1041  CONTINUE
1040  CONTINUE
C
C     INPUT INITIAL CONDITION
C
      N1=NN+1
      DO 1006 C=1,MM
         DO 1001 N=1,N1
         S(N,1)=1.0
1001     CONTINUE
         MX=1
         MY=2
         DO 1002 IXX=1,MM
            M=C+IXX
            IF(M.GT.MM)M=M-MM
            MI=MX
            MX=MY
            MY=MI
```

```
C
C     FIND S(N,M)
C
            S(1,MX)=1.0/(2.718**O(C))
            DO 1005 N=2,N1
                  S(N,MX)=0.0
                  JJ=(N-1)/K(M)
                  JI=NX(M)
                  EM=(FINO(JI,JJ))+1
                  DO 1010 B=1,EM
                        N2=N-(K(M)*(B-1))
                        X1=S(N2,MY)*V(B,M)
                        S(N,MX)=S(N,MX)+X1
1010                    CONTINUE
1005              CONTINUE
1002        CONTINUE
C
C     FIND BLOCKING PROBABILITY, PBN(P)
C
            ND=NX(C)+1
            I1=N1-(K(C)*ND)
            I2=N1-K(C)
            PBN(C)=(S(N1,MX)-S(I2,MX)+(S(I1,MY)*V(ND,C)))/S(N1,MX)
17          FORMAT(20X,'PBN(',I2,')=',F8.6)
C           WRITE (6,17)C,PBN(C)
C
C     RELABEL INDEX AND REPEAT
C
1006        CONTINUE
C
C     FIND TOTAL WEIGHTED BLOCKING
C
            F=0.0
            DO 1030 M=1,MM
            F=W(M)*PBN(M)+F
1030  CONTINUE
C     WRITE(6,27)F
27    FORMAT(/10X,'OBJECTIVE FUNCTION (B) = ',F8.3/)
      RETURN
      END
```

(2) **Optimization of DAMA Systems**
Sharing Policy : Sharing with Maximum Allowed
Criterion : Maximize Weighted Throughput

```
      DIMENSION K(10),Q(10),NX(10),NXB(10),NXBO(10),NXP(10),NXO(10),
     +V(300,10),NBP(10),S(300,2),BBN(10),W(10)
      INTEGER M,BN,C,HALV1,HALV2,HALV3,BPO,BFOM,BP,PM
C
C     INPUT SYSTEM PARAMETERS
C
      READ (3,10)MM,NN
      WRITE (6,10)MM,NN
10    FORMAT(2I5)
      READ (3,11)(K(M),M=1,MM)
      WRITE (6,11)(K(M),M=1,MM)
11    FORMAT(10I5)
      READ (3,12)(Q(M),M=1,MM)
      WRITE (6,12)(Q(M),M=1,MM)
12    FORMAT(10F7.2)
      READ (3,12)(W(M),M=1,MM)
      WRITE(6,12)(W(M),M=1,MM)
      READ(3,14)(NX(M),M=1,MM)
      WRITE (6,14)(NX(M),M=1,MM)
      READ(3,14)(NXB(M),M=1,MM)
      WRITE (6,14)(NXB(M),M=1,MM)
14    FORMAT(10I5)
      READ(3,13)HALM,HALM2,BFOM
      WRITE (6,13)HALM,HALM2,BFOM
13    FORMAT(3I5)
      IT=0
      HALV3=0
      CALL OBJ(MM,NN,N,C,M,NX,IT,T,IMN,F)
      FO=F
121   DO 1020 M=1,MM
      NXP(M)=0
      NXO(M)=NN/K(M)
      NXBO(M)=NXB(M)
1020  CONTINUE
      BPO=0
      PM=0
      BP=0
106   HALV1=0
C
C     STOP AT IF HALM IS EXCEEDED
C
105   IF(HALV1.GT.HALM) GOTO 107
      BPO=BPO+1
      IF(BPO.GT.BFOM)GOTO 120
```

```
C
C     LOCATE INITIAL BASE POINT
C
      IN=0
      BPO=BPO+1
C     WRITE(6,30)
30    FORMAT(/'*********************************************')
C     WRITE (6,18)BPO
18    FORMAT('**** BPO = ',I5)
      DO 1021 M=1,MM
C     WRITE(6,20)M
20    FORMAT(10X,'M = ',I5/)
      IF(NXB(M).EQ.0)GOTO 109
      NX(M)=NX(M)-NXB(M)
      IF(NX(M).LE.0)GOTO 117
      CALL OBJ(MM,NN,P,C,B,NX,IT,T,FMN,F)
      IF(FO.GE.F)FO=F
      IF(FO.GE.F)NXB(M)=NXB(M)+NXB(M)
      IF(FO.GE.F)GOTO 1021
117   NX(M)=NX(M)+2*NXB(M)
      IF(NX(M).GT.NXO(M))GOTO 106
      CALL OBJ(MM,NN,P,C,B,NX,IT,T,FMN,F)
      IF(FO.GE.F)FO=F
      IF(FO.GE.F)NXB(M)=NXB(M)-NXB(M)
      IF(FO.GE.F)GOTO 1021
108   NX(M)=NX(M)-NXB(M)
109   IN=IN+1
1021  CONTINUE
C
C     IF INITIAL BASE POINT IS FOUND,PROCEED WITH PATTERN MOVE
C
      IF(IN.LT.MM) GOTO 100
C
C     ELSE REDUCE STEP SIZE AND TRY AGAIN
C
      DO 1025 M=1,MM
      NXB(M)=NXB(M)/2
1025  CONTINUE
      HALV1=HALV1+1
      GOTO 105
C
C     RECORD CURRENT BP
C
100   DO 1029 M=1,MM
      NBP(M)=NX(M)
1029  CONTINUE
C
C     PROCEED WITH PATTERN MOVE
C
      PM=PM+1
C     WRITE (6,16)PM
16    FORMAT(5X,'**** PM = ',I5/)
      INDB=0
      DO 1022 M=1,MM
      NX(M)=NX(M)-NXP(M)
      IF((NX(M).GT.NXO(M)).OR.(NX(M).LE.0))INDB=1
1022  CONTINUE
```

```
C       WRITE(6,29)(NX(I),I=1,NN)
29      FORMAT(5I5)
C
C       RESET STEP SIZE TO INITIAL VALUE
C
        DO 1027 I=1,NN
        NXB(I)=NXB0(I)
1027    CONTINUE
C
C       IF PM MOVED NX OUT OF BOUND, DISCARD PM
C
        IF(IADE.EQ.1)GOTO 116
C
C       ELSE LOCATE NEW BP
C
        BP=0
        HALV2=0
103     IND=0
        BP=BP+1
C       WRITE(6,19)BP
19      FORMAT(10X,'**** BP = ',I5/)
        DO 1023 I=1,NN
C       WRITE(6,21)I
21      FORMAT(10X,'I = ',I5/)
        IF(NXB(I).EQ.0)GOTO 111
        NX(I)=NX(I)-NXB(I)
        IF(NX(I).LE.0)GOTO 118
        INDEX=0
        DO 1032 J=1,NN
        IF(NX(J).EQ.NBE(J))INDEX=INDEX+1
1032    CONTINUE
        IF(INDEX.EQ.NN)GOTO 118
        CALL OBJ(NN,NN,K,G,F,NX,IT,J,FNN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(I)=NXB(I)+NXB(I)
        IF(FO.GE.F)GOTO 1023
118     NX(I)=NX(I)+2*NXB(I)
        IF(NX(I).GT.NXO(I))GOTO 110
        INDEX=0
        DO 1033 J=1,NN
        IF(NX(J).EQ.NBE(J))INDEX=INDEX+1
1033    CONTINUE
        IF(INDEX.EQ.NN)GOTO 110
        CALL OBJ(NN,NN,K,G,F,NX,IT,J,FNN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(I)=NXB(I)-NXB(I)
        IF(FO.GE.F)GOTO 1023
110     NX(I)=NX(I)-NXB(I)
111     IND=IND+1
1023    CONTINUE
C
C       IF NEW BP IS FOUND, PROCEED WITH PATTERN MOVE
C
        IF(IND.LT.NN)GOTO 100
```

```
C
C     ELSE REDUCE STEP SIZE AND TRY AGAIN
C
      HALV2=HALV2+1
      IF(HALV2.GT.HALF2)GOTO 104
      DO 1024 M=1,MM
      NXB(M)=NXB(M)/2
1024  CONTINUE
      GOTO 103
C
C     IF NEW BP IS NOT FOUND AFTER STEP SIZE IS REDUCED REPEATEDLY,
C     DISCARD LAST PATTERN MOVE, RESTORE INITIAL STEP SIZE AND
C     RETURN TO PREVIOUS BP
C
104   DO 1026 M=1,MM
      NXB(M)=NXBO(M)
1028  CONTINUE
116   DO 1026 M=1,MM
      NX(M)=NX(M)+NXP(M)
      NXP(M)=0
1026  CONTINUE
C
C     RE INITIATE PATTERN SEARCH
C
      GOTO 106
C
C     RE INITIATE PATTERN SEARCH WITH REDUCED STEP SIZE
C
120   HALV3=HALV3+1
      IF(HALV3.GT.HALF)GOTO 107
      DO 1035 M=1,MM
      NXB(M)=NXB(M)/2
1035  CONTINUE
      GOTO 121
C
C     RECORD OPTIMIZATION RESULT
C
107   WRITE (6,15)
15    FORMAT(//'******** NO. OF STEP-SIZE-HALVINGS EXCEEDED ********'/)
      CALL OBJ(MM,NN,M,C,P,NX,IT,T,FMN,F)
      WRITE(6,32)IT
32    FORMAT('NO. OF ITERATIONS = ',I5/)
      DO 1034 M=1,MM
      WRITE(6,33)M,FMN(M),M,NX(M),M,NXB(M)
33    FORMAT(5X,'FMN(',I2,')=',F8.6,5X,'NX(',I2,')=',I5,
     +7X,'NXB(',I2,')=',I5)
1034  CONTINUE
      WRITE(6,35)T
35    FORMAT(/5X,'SYSTEM WEIGHTED THROUGHPUT * D = ',E 8.3)
      WRITE(6,34)F
34    FORMAT(/5X,'OBJECTIVE FUNCTION = ',E8.3/)
      STOP
      END
```

```
C
C
C
C     ***************************************************************
C
C     SUBROUTINE TO DETERMINE VALUE OF OBJECTIVE FUNCTION
C
C     ***************************************************************
C
      SUBROUTINE OBJ(PM,MM,R,Q,P,NX,IT,T,PMN,F)
C
C
      DIMENSION R(10),Q(10),NX(10),NXB(10),NXE(10),
     *V(300,10),S(300,2),PMN(10),P(10)
      INTEGER M,PM,C,HALF,HALV1,HALV2
      IT=IT+1
C     WRITE(6,28)IT
28    FORMAT(/'ITERATION ',I3/)
C     WRITE(6,91)(NX(M),M=1,MM)
91    FORMAT(1015)
C
C     FIND V(M,M)
C
      DO 1040 M=1,MM
      N3=NX(M)+1
      V(1,M)=1.0
      DO 1041 M=2,N3
      V(M,M)=1.0
      N4=M-1
      DO 1042 I=1,N4
      V(M,M)=V(M,M)*Q(M)/I
1042  CONTINUE
1041  CONTINUE
1040  CONTINUE
C
C     INPUT INITIAL CONDITION
C
      N1=NN+1
      DO 1006 C=1,MM
         DO 1001 M=1,N1
         S(M,1)=1.0
1001     CONTINUE
         MX=1
         MY=2
         DO 1002 IXX=1,MM
            M=C+IXX
            IF(M.GT.MM)M=M-MM
            MI=MX
            MX=MY
            MY=MI
```

```
C
C     FIND S(N,M)
C
            S(1,MX)=1.0/(2.718**C(C))
            DO 1005 N=2,N1
                  S(N,MX)=0.0
                  JJ=(N-1)/K(M)
                  JI=MX(M)
                  EN=(FINO(JI,JJ))+1
                  DO 1010 H=1,EN
                        N2=N-(K(M)*(H-1))
                        X1=S(N2,MY)*V(H,M)
                        S(N,MX)=S(N,MX)+X1
1010                    CONTINUE
1005              CONTINUE
1002        CONTINUE
C
C     FIND BLOCKING PROBABILITY, PBN(M)
C
           ND=MX(C)+1
           I1=N1-(K(C)*ND)
           I2=N1-K(C)
           PBN(C)=(S(N1,MX)-S(I2,MX)+(S(I1,MY)*V(ND,C)))/S(N1,MX)
17       FORMAT(20X,' PBN(',I2,')=',F8.6)
C        WRITE (6,17)C,,PN(C)
C
C     RELABEL INDEX AND REPEAT
C
1006  CONTINUE
C
C     FIND WEIGHTED THROUGHPUT, T
C
      T=0.0
      DO 1030 M=1,MM
      T=W(M)*Q(M)*(1-PBN(M))+T
1030  CONTINUE
C     WRITE(6,23)T
23    FORMAT(/10X,'SYSTEM WEIGHTED THROUGHPUT * U = ',F6.3/)
C
C     FIND OBJECTIVE FUNCTION, F
C
      F=0.0
      DO 1031 M=1,MM
      F=W(M)*Q(M)*PBN(M)+F
1031  CONTINUE
C     WRITE(6,27)F
27    FORMAT(/10X,'OBJECTIVE FUNCTION = ',E8.3/)
      RETURN
      END
```

(3) **Optimization of DAMA Systems**
Sharing Policy : Sharing with Maximum Allowed
Criterion : Maximize Weighted Utilization

```
      DIMENSION K(10),Q(10),NX(10),NXE(10),NXBO(10),NXP(10),NXO(10),
     +V(300,10),PN(10),NBP(10),S(300,2),PMN(10),W(10)
      INTEGER M,PP,C,HALM,HALM2,HALV1,HALV2,HALV3,EPO,BIO,BP,PM
C
C     INPUT SYSTEM PARAMETERS
C
      READ (3,10)MM,NN
      WRITE (6,10)MM,NN
10    FORMAT(2I5)
      READ (3,11)(K(M),M=1,MM)
      WRITE (6,11)(K(M),M=1,MM)
11    FORMAT(10I5)
      READ (3,12)(Q(M),M=1,MM)
      WRITE (6,12)(Q(M),M=1,MM)
12    FORMAT(10F7.2)
      READ (3,12)(W(M),M=1,MM)
      WRITE(6,12)(W(M),M=1,MM)
      READ(3,14)(NX(M),M=1,MM)
      WRITE (6,14)(NX(M),M=1,MM)
      READ(3,14)(NXE(M),M=1,MM)
      WRITE (6,14)(NXE(M),M=1,MM)
14    FORMAT(10I5)
      READ(3,13)HALM,HALM2,EPOM
      WRITE (6,13)HALM,HALM2,EPOM
13    FORMAT(3I5)
      IT=0
      HALV3=0
      CALL OBJ(MM,NN,K,C,W,NX,IT,T,PMN,AN,F)
      FO=F
121   DO 1020 M=1,MM
      NXP(M)=0
      NXO(M)=NN/K(M)
      NXBO(M)=NXE(M)
1020  CONTINUE
      BPO=0
      PM=0
      BP=0
106   HALV1=0
C
C     STOP A1 IF HALM IS EXCEEDED
C
105   IF(HALV1.GT.HALM) GOTO 107
      BPO=BPO+1
      IF(BPO.GT.EPOM)GOTO 120
```

```
C
C       LOCATE INITIAL BASE POINT
C
        IN=0
        BPO=BPO+1
C       WRITE(6,30)
30      FORMAT(/'*****************************************')
C       WRITE (6,18)BPO
18      FORMAT('**** BPO = ',I5)
        DO 1021 M=1,MM
C       WRITE(6,20)M
20      FORMAT(10X,'M = ',I5/)
        IF(NXB(M).EQ.0)GOTO 109
        NX(M)=NX(M)-NXB(M)
        IF(NX(M).LE.0)GOTO 117
        CALL OBJ(MM,NN,K,C,B,NX,IT,U,FMN,AN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)+NXB(M)
        IF(FO.GE.F)GOTO 1021
117     NX(M)=NX(M)+2*NXB(M)
        IF(NX(M).GT.NXO(M))GOTO 108
        CALL OBJ(MM,NN,K,C,B,NX,IT,U,FMN,AN,F)
        IF(FO.GE.F)FO=F
        IF(FO.GE.F)NXB(M)=NXB(M)-NXB(M)
        IF(FO.GE.F)GOTO 1021
108     NX(M)=NX(M)-NXB(M)
109     IN=IN+1
1021    CONTINUE
C
C       IF INITIAL BASE POINT IS FOUND,PROCEED WITH PATTERN MOVE
C
        IF(IN.LT.MM) GOTO 100
C
C       ELSE REDUCE STEP SIZE AND TRY AGAIN
C
        DO 1025 M=1,MM
        NXB(M)=NXB(M)/2
1025    CONTINUE
        HALV1=HALV1+1
        GOTO 105
C
C       RECORD CURRENT BP
C
100     DO 1029 M=1,MM
        NBP(M)=NX(M)
1029    CONTINUE
C
C       PROCEED WITH PATTERN MOVE
C
        PM=PM+1
C       WRITE (6,16)PM
16      FORMAT(5X,'**** PM = ',I5/)
        INDE=0
        DO 1022 M=1,MM
        NX(M)=NX(M)-NXB(M)
        IF((NX(M).GT.NXO(M)).OR.(NX(M).LE.0))INDE=1
1022    CONTINUE
```

```
C     WRITE(6,29)(NX(M),M=1,MM)
29    FORMAT(5I5)
C
C     RESET STEP SIZE TO INITIAL VALUE
C
      DO 1027 M=1,MM
      NXB(M)=NXBC(M)
1027  CONTINUE
C
C     IF PM MOVES NX OUT OF BOUND, DISCARD PM
C
      IF(INDE.EQ.1)GOTO 116
C
C     ELSE LOCATE NEW BP
C
      BP=0
      HALV2=0
103   IND=0
      BP=BP+1
C     WRITE(6,19)BP
19    FORMAT(10X,'**** BP = ',I5/)
      DO 1023 M=1,MM
C     WRITE(6,21)M
21    FORMAT(10X,'M = ',I5/)
      IF(NXB(M).EQ.0)GOTO 111
      NX(M)=NX(M)-NXB(M)
      IF(NX(M).LE.0)GOTO 116
      INDEX=0
      DO 1032 I=1,MM
      IF(NX(I).EQ.NBP(I))INDEX=INDEX+1
1032  CONTINUE
      IF(INDEX.EQ.MM)GOTO 116
      CALL OBJ(MM,NN,K,O,W,NX,IT,U,PMN,AN,F)
      IF(FO.GE.F)FO=F
      IF(FO.GE.F)NXP(M)=NXB(M)+NXB(M)
      IF(FO.GE.F)GOTO 1023
116   NX(M)=NX(M)+2*NXB(M)
      IF(NX(M).GT.NXO(M))GOTO 110
      INDEX=0
      DO 1033 I=1,MM
      IF(NX(I).EQ.NBP(I))INDEX=INDEX+1
1033  CONTINUE
      IF(INDEX.EQ.MM)GOTO 110
      CALL OBJ(MM,NN,K,O,W,NX,IT,U,PMN,AN,F)
      IF(FO.GE.F)FO=F
      IF(FO.GE.F)NXP(M)=NXB(M)-NXB(M)
      IF(FO.GE.F)GOTO 1023
110   NX(M)=NX(M)-NXB(M)
111   IND=IND+1
1023  CONTINUE
C
C     IF NEW BP IS FOUND, PROCEED WITH PATTERN MOVE
C
      IF(IND.LT.MM)GOTO 100
```

```
C
C     ELSE REDUCE STEP SIZE AND TRY AGAIN
C
      HALV2=HALV2+1
      IF(HALV2.GT.HALM2)GOTO 104
      DO 1024 M=1,MM
      NXB(M)=NXB(M)/2
1024  CONTINUE
      GOTO 103
C
C     IF NEW BP IS NOT FOUND AFTER STEP SIZE IS REDUCED REPEATEDLY,
C     DISCARD LAST PATTERN MOVE, RESTORE INITIAL STEP SIZE AND
C     RETURN TO PREVIOUS BP
C
104   DO 1028 M=1,MM
      NXB(M)=NXBO(M)
1028  CONTINUE
116   DO 1026 M=1,MM
      NX(M)=NX(M)+NXP(M)
      NXP(M)=0
1026  CONTINUE
C
C     RE INITIATE PATTERN SEARCH
C
      GOTO 106
C
C     RE INITIATE PATTERN SEARCH WITH REDUCED STEP SIZE
C
120   HALV3=HALV3+1
      IF(HALV3.GT.HALM)GOTO 107
      DO 1035 M=1,MM
      NXB(M)=NXB(M)/2
1035  CONTINUE
      GOTO 121
C
C     RECORD OPTIMIZATION RESULT
C
107   WRITE (6,15)
15    FORMAT(//'********* NO. OF STEP-SIZE-HALVINGS EXCEEDED *********'/)
      CALL OBJ(PP,NN,K,C,W,NX,IT,U,PPN,AN,F)
      WRITE(6,32)IT
32    FORMAT('NO. OF ITERATIONS = ',I5/)
      DO 1034 M=1,MM
      WRITE(6,33)M,PMN(M),M,AN(M),M,NX(M),M,NXB(M)
33    FORMAT(2X,'PMN(',I2,')=',F8.6,5X,'AN(',I2,')=',F5.2,
     +5X,'NX(',I2,')=',I3,5X,'NXB',I2,')=',I3)
1034  CONTINUE
      WRITE(6,35)U
35    FORMAT(/5X,'SYSTEM WEIGHTED UTILIZATION = ',F8.3)
      WRITE(6,34)F
34    FORMAT(/5X,'OBJECTIVE FUNCTION = ',E8.3/)
      STOP
      END
```

```
C     ***************************************************************
C
C     SUBROUTINE TO DETERMINE VALUE OF OBJECTIVE FUNCTION
C
C     ***************************************************************
C
      SUBROUTINE OBJ(MM,NN,R,Q,W,NX,IT,U,PMN,AN,F)
C
C
      DIMENSION R(10),Q(10),NX(10),NXB(10),NXP(10),AA(10),
     +V(300,10),S(300,2),PMN(10),P(10)
      INTEGER B,BM,C,HALM,MAIV1,MAIV2
      IT=IT+1
C     WRITE(6,28)IT
28    FORMAT(/'ITERATION ',I3/)
C     WRITE(6,91)(NX(M),M=1,MM)
91    FORMAT(10I5)
C
C     FIND V(B,M)
C
      DO 1040 M=1,MM
      N3=NX(M)+1
      V(1,M)=1.0
      DO 1041 B=2,N3
      V(B,M)=1.0
      N4=B-1
      DO 1042 I=1,N4
      V(B,M)=V(B,M)*Q(M)/I
1042  CONTINUE
1041  CONTINUE
1040  CONTINUE
C
C     INPUT INITIAL CONDITION
C
      N1=NN+1
      DO 1006 C=1,MM
         DO 1001 N=1,N1
         S(N,1)=1.0
1001     CONTINUE
         MX=1
         MY=2
         DO 1002 IXX=1,MM
            M=C+IXX
            IF(M.GT.MM)M=M-MM
            MI=MX
            MX=MY
            MY=MI
C
C     FIND S(N,M)
C
            S(1,MX)=1.0/(2.718**Q(C))
            DO 1005 N=2,N1
                  S(N,MX)=0.0
                  JJ=(N-1)/R(M)
```

```
                  JI=NX(M)
                  BM=(PINO(JI,JJ))+1
                  DO 1010 H=1,BM
                       N2=N-(K(M)*(H-1))
                       X1=S(N2,MY)*V(H,L)
                       S(N,MX)=S(N,MX)+X1
1010                   CONTINUE
1005              CONTINUE
1002         CONTINUE
C
C      FIND BLOCKING PROBABILITY, PBN(M)
C
           ND=NX(C)+1
           I1=N1-(K(C)*ND)
           I2=N1-K(C)
           PBN(C)=(S(N1,MX)-S(I2,MX)+(S(I1,MY)*V(ND,C)))/S(N1,MX)
C
C      FIND AVERAGE NO. OF CAPACITY UNITS ENGAGED BY GROUP C
C
       AN(C)=0.0
       I4=NX(C)
       DO 1036 LD=1,I4
       I3=N1-(K(C)*LD)
       I5=LD+1
       AN(C)=S(I3,MY)*LD*V(I5,C)+AN(C)
1036   CONTINUE
       AN(C)=K(C)*AN(C)/S(N1,MX)
C      WRITE(6,22)C,PBN(C),C,AN(C)
22     FORMAT(10X,'PBN(',I3,')=',F6.5,5X,'AN(',I2,')=',F8.5)
C
C      RELABLE INDEX AND REPEAT
C
1006   CONTINUE
       AVN=0.0
       DO 1037 M=1,MM
       AVN=W(M)*AN(M)+AVN
1037   CONTINUE
C
C      FIND SYSTEM WEIGHTED UTILIZATION
C
       U=AVN/NN
C      WRITE(6,31)U
31     FORMAT(10X,'SYSTEM WEIGHTED UTILIZATION = ',F8.3)
C
C      FIND OBJECTIVE FUNCTION
C
       F=1/U
C      WRITE(6,27)F
27     FORMAT(/10X,'OBJECTIVE FUNCTION = ',F8.3/)
       RETURN
       END
```

REFERENCES

1. G. Barberis and R. Brignlol, "Capacity Allocation in a DAMA Satellite System", IEEE Trans. Comm., Vol Com-30, No.7, July 1982, pp.1750-1757.
2. J. Aein, "A Multi-user-class, Blocked-calls-cleared Demand Access Model", Trans. on Comm., Vol. Com-26, No.3, March 78, pp.378-385.
3. J. Aein, O. Kosovych, "Satellite Capacity Allocation", Proceedings of the IEEE, Vol.65, No. 3, March 1977, pp.332-342.
4. R. B. Cooper, "Introduction to Queueing Network", Macmillan Co., N.Y., 1972.
5. Constantine Karmokolias, "A Performance Model for Multi-receiver Terminals", to be published.
6. J. S. Kaufman, "Blocking in a Shared Resource Environment", IEEE Trans.Comm., Vol. Com-29, No.10, Oct. 1980, pp.1474-1481.
7. G. J. Foschini, B. Gopinath, "Sharing Memory Optimally", IEEE Trans. Comm., Vol. Com-31, No.3, March 1983, pp.352-360.
8. R. Hooke, T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", J. Assn. Comp., Vol. 8, April 1961, pp.212-229.

9. H. Kobayashi, "Modelling and Analysis: An Introduction to System Performance Evaluation Methodology, Reading MA: Addison- Wesley, 1978, pp.168-174.