THE ALLOCATION OF NON-IDENTICAL MACHINES AMONG NON-IDENTICAL SERVERS

by

Thomas A. Carpenito

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial Engineering and Operations Research

APPROVED:

John A. White, Chairman

Marvin H. Agee

G. Kemble Bennett

May, 1974

Blacksburg, Virginia

ACKNOWLEDGMENTS

The author wishes to express his sincere appreciation to the many people who have generously given advice and assistance through the development and writing of this thesis.

Special recognition is due Dr. John A. White, the author's major advisor, for the time, guidance, encouragement and friendship he provided during the course of this research.

Gratitude is also expressed to the other members of the graduate committee, Dr. G. Kemble Bennett and Dr. Marvin H. Agee for their support, suggestions and understanding during the interim and final stages of the thesis preparation.

Thanks are also extended to for her patience and excellent typing of the manuscript when time was at a premium.

Finally, the author expresses his appreciation to the National Collegiate Athletic Association for the financial support provided.

TABLE OF CONTENTS

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

LIST OF TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

Subject of the Research

The problem of machinery failure and the servicing of these machines is a problem which is of concern to every industry involved in the manufacture of a product. The costs associated with the failure of a machine and its subsequent repair can become a critical factor in the costs of producing an item for sale. Improper handling of this problem can result in many unnecessary expenses for the manufacturer. It is the purpose of this research to develop a procedure which will help in the minimization of these costs.

It is generally recognized that the best method of allocating machines in need of repair is by pooling, such that servers are not assigned specific machines to repair [24]. Instead, the servers are considered to be assigned to a pool, and when a machine requires service, a server from the pool is assigned to the job. In minimizing the costs incurred using this procedure, the objective is to determine the optimal number of servers to assign to the pool if the servers have identical service rates. If the service rates are not the same, then the

optimum assignment is found by determining which servers should be assigned to the pool.

It should be noted that there are instances when it is not feasible to use a service pool. For example, suppose the cost of travel between the pooling area and the machinery is such that it is impractical to locate all the servers in one area. As a further illustration, if one were to ignore the effect of travel time, by either the servers or the customers then it would be optimal to have, say, only one large restroom in the Pentagon [7]. Another instance when pooling is not desirable is if management deems it necessary to have only one person work on a machine in order to pinpoint responsibility for workmanship. There could be other situations, in which it is undesirable to have a service pool. Such a situation is the subject of this research. Given the fact that a service pool can not be used and each server will be responsible for a specific group of machines, it is desired to determine the best allocation of these machines to the servers in order to optimize a given objective function.

Problem Definition

For purposes of this research, it is assumed that the machinery is of two types. The time between failures is assumed to be distributed exponentially for all machines; however, the failure rates differ depending on the type of

machine. Machines of the same type have identical failure rates, but the rate of breakdowns may be different for machines of different types. The time required to service the machines is also assumed to be distributed exponentially, but again with different rates depending on which type of machine is being serviced. Also, the service rates differ between the given operators. As previously stated, the objective of this research is to allocate the non-identical machines to the non-identical servers (different service rates), in such a manner that a given objective function is optimized. To do this, the problem will be formulated as a queuing model. To simplify the discussion, the term customers will be used to denote the machines and servers will refer to the repairmen assigned to the machines.

The system under study can be formulated as a collection of independent single server, finite population subsystems. The number of subsystems in the total system is equal to the number of servers.

Criterion Selection

To assist the production manager in the allocation of machines among servers, it is recommended that a prescriptive model be developed based on the manager's principle of choice. Due to the explicit recognition of risk considered in this research, the expectation principle of choice will be considered. Alternately, an aspiration level or

expectation-variance principle of choice might better model the decision making process of the manager. However, for purposes of this research, an expected cost model will be developed for the total system and machines will be allocated to servers in such a way that expected cost is minimized. Since only steady state results are considered, the decision is most likely being made for the long run. Aspiration level decisions are often used when a specific service level objective is to be met. Situations where aspiration level decisions are appealing include the case where it is too costly to search for the true optimum solution to a problem or when it is too costly to obtain the needed data for an expected cost model. That is, it may be difficult to estimate the cost terms for the expectation model.

Ideally, the best decision rule is the maximization of expected utility. However, such an approach is not felt to be feasible for the present case due to the difficulties in measuring the manager's utility function. Thus, it is assumed that the utility function is linear over the region of interest. Consequently, minimizing expected cost maximizes expected utility. The minimization of expected cost is also a rational objective if the decision is to be based on long run considerations. It should be emphasized that the selection of the type decision model to use should be based on the manager's objectives, rather than the ease of data collection [25].

As previously discussed, the total system under study is composed of a collection of single server, finite population subsystems. The total cost function is therefore the sum of the costs for each server's subsystem. There are three basic costs which are considered in the development of the cost function. One cost is the cost per unit time of a customer waiting for service. A second cost is the cost per unit time of a customer in service. The last cost associated with this formulation of the problem is the cost per unit time of the server. Sometimes the cost of a server is separated into two distinct costs. One portion represents the cost incurred when the server is idle and the other represents the cost of a server being busy. For this model, these costs are considered to be equal, and therefore may be represented by a single cost.

There are two basic problems associated with this formulation. One, of course, is the optimization of the model once the necessary parameters have been specified. The other is the verification of the assumptions of the model and the determization of the correct values for the needed parameters. The solution procedures presented can be used for any queuing model, as long as values of the operating characteristics for the individual subsystems can be determined. In some cases (non-exponential arrivals or services), the determination may require the use of simulation. In order to evaluate the function as we do in this

paper, both the interarrival and service time distributions for each customer must be exponential. To verify that the distributions are indeed exponentially distributed, it is necessary to perform a statistical test such as the Kolmogorov-Smirnov test, to determine if the interarrival and service time distributions are distributed exponentially. Once it has been determined that the distributions are exponential, the mean number of arrivals per unit time and the mean number of services per unit time are determined by simply averaging the data which has been gathered. Determining accurate cost figures for the model may be very difficult. Since this might be a prohibitive factor in the use of the model, a discussion concerning the determination of these costs is appropriate.

Cost Determination

Much of the work done in queuing concerns the use of criteria other than the minimization of expected cost. The rationale for this is that it is impossible to develop accurate cost models due to the difficulty in obtaining exact cost estimates. Hillier [9] disagrees with this attitude stating, "A penetrating analysis by competent people should yield a usable estimate of the cost of waiting." His writings [7], [8], [9] in this area provide a good basis for determining appropriate and accurate cost figures.

In the given model, we are concerned with the costs involving customer service and waiting and server costs. Since the customers are machines in need of repair, this will somewhat simplify the cost determination. It should be noted that the discussion for the determination of waiting cost C_1, also applies to the service cost C_2 [25].

The cost of waiting may be divided into two categories, long run net incremental income foregone and long run net incremental expenses incurred [9]. In determining the net income foregone, one must first determine how much output is lost due to the delay of one item for one unit of time. In ascertaining this, consideration must be given to whether or not the lost production can be made up and, if it can be made up, what incremental expenses are incurred, if any. Since the total lost production is desired, if the waiting of one item produces a chain reaction causing lost output in other areas, this must also be considered.

Once the total net long run output loss has been obtained, the next step is to determine the net income loss associated with the lost output. The value of the lost output can be estimated by the reduction in sales income over the long run. By adding the direct and indirect expenses of each operation to the lost profit, and subtracting the incremental expenses which were not incurred due to the

lost output, one may obtain a good estimate of the value of the lost output per unit time.

Some parts of the second component of waiting cost, long run net incremental expenses incurred, may not be applicable. There may be no cost incurred because of customer dissatisfaction, especially if the product is in great demand on the retail market. If there is a cost, it must be determined by a judgment concerning the value of prompt deliveries. The costs incurred by idle in-process inventories are simply the interest costs of the capital being used by the inventories which have been idled due to the waiting of the customer demanding service. Lastly, net expenses occur due to increased supervision and administrative costs caused by waiting. This is usually a minor cost, considered to be a nuisance cost, and can often be estimated.

The cost of the server is a function of the wages paid to the server. Since in our case, the idle cost is equal to the cost when the server is busy, it is assumed that the compensation is a strict hourly (weekly) wage with no incentive plan. Therefore, the cost of the server is simply the wages paid to him, and is much more easily determined than the previous costs.

Although estimating the pertinent costs seems to be a difficult chore, it certainly is not an impossible one.

As previously stated, Hillier [9] believes the costs can be determined by competent people with a reasonable amount of effort.

For this research, an expected cost model is used. Such a model is believed to be valid in this case due to the fact that if the costs are determined as described in the preceding discussion, good estimates of the value of the costs can be obtained. If the cost function is relatively insensitive to variations in the parameters, the estimates should be satisfactory. This will be discussed in depth in Chapter IV.

Scope and Limitations

In considering the validity of this model for optimally assigning machines to servers, certain constraints must be imposed on the situation. If travel between faulty machinery is not a problem or if it is permissible for any server to service any machine and there are no other constraints on the assignment, then the pooling procedure discussed earlier would be preferred, and the problem should be formulated by considering priorities among customers. If one or both of these constraints does apply, then this model appears justified.

The assumption of exponentially distributed inter-arrival and service times is not as much of a limitation as it may initially appear. Experience has shown that this

assumption is often valid. Of course, before using the model, it is necessary to test to see if the respective distributions are indeed exponentially distributed.

Related Research

Much work has been done concerning the optimal allocation of machines to servers, where the machines have identical breakdown distributions and the servers have identical service distributions. Palm [19] has examined this problem in a paper written in Swedish and later translated into English. In 1950, Ashcroft [1] examined the same problem where the service rates were constant. Fetter [5], Fetter and Galliher [6], King [14], Naor [18], and Mangelsdorf [16] among others, considered the machines assignment problem for the case of identical machines and identical servers.

Queuing problems involving service priorities have received a great deal of attention. Hooke [12], [13], derives limiting properties for a priority queue with an infinite population. Kredentser [15] developed a model for a two-priority queue with infinite populations and the service distribution is the same for both priorities.

Smith [21] developed a model with arrivals generated from several finite populations. The service rate for each type of customer is also different. The difference between

his research and the research for this paper is that Smith allows for no queue build-up. In other words, the number of servers is equal to the sum of the population sizes. In the research presented in this paper each subsystem has only one server. Numerous other papers have been published on priorities in queues; however, a search of the literature indicates the research has been devoted primarily to the study of infinite population problems.

Chapter II

DEVELOPMENT OF THE MODEL

Introduction

This chapter is devoted to the development of the expected cost model described briefly in Chapter I. The major emphasis for the chapter is on a queuing system having arrivals generated from two distinct populations. However, the model for a subsystem having only one population is also needed to minimize the objective function.

The system under study can be viewed as a collection of independently operating single server, finite population subsystems. Using the properties of the Poisson process, the steady state balance equations for each subsystem are developed and a procedure for solving the set of balance equations is presented.

Cost Function

As stated in Chapter I, the cost function for the total system involves the sum of the costs of each subsystem. Using this, the objective function may be written

$$\text{Minimize } f(x_{ij}) = \sum_{j=1}^{c} \{C_{11}L_{q1}^{j}(x_{ij}) + C_{12}\,L_{q2}^{j}(x_{ij}) + C_{21}[L_{1}^{j}(x_{ij}) - L_{q1}^{j}(x_{ij})] + C_{22}[L_{2}^{j}(x_{ij}) - L_{q2}^{j}(x_{ij})] + \delta_j\, C_{3j}\}$$

Subject to

$$\sum_{y=1}^{c} x_{1j} = M$$

$$\sum_{j=1}^{c} x_{2j} = N$$

$$\delta_j = \begin{cases} 0 & \text{if } x_{1j} = x_{2j} = 0 \\ 1 & \text{otherwise} \end{cases}$$

where

C_{1i} = cost per unit time for one customer of type i to wait in queue

C_{2i} = cost per unit time for one customer of type i to be serviced

C_{3j} = cost per unit time of server j

x_{ij} = number of type i customers assigned to server j

$L_i^j(x_{ij})$ = expected number of customers of type i in subsystem j as a function of x_{ij}

$L_{qi}^j(x_{ij})$ = expected number of customers of type i in the queue of subsystem j as a function of x_{ij}

M = population size of type 1 customers

N = population size of type 2 customers

c = number of servers (subsystems)

The variables in the objective function are $L_i^j(x_{ij})$ and $L_{qi}^j(x_{ij})$. A method for determining values for these variables will be discussed later in the chapter.

General Development

Before developing the balance equations for the system, the assumptions and parameters for the model require some discussion. The following assumptions are made:

1. Interarrival time for each customer of type i is exponentially distributed with mean $1/\lambda_i$.
2. The time required for server j to provide service for customer type i is exponentially distributed with mean $1/\mu_{ij}$.
3. If both types of customers are waiting to be served immediately after the server completes a service, the probability that a type 1 customer will be the next to be served is q; the probability that a type 2 customer will be the next into service is 1-q.
4. $M > 0$, and
   $N > 0$.

Let P_{mni} represent the steady state probability that there are m customers of type 1 in the system, n customers of type 2 in the system and customer type i is presently in service, where $m = 0, 1, 2, \ldots, M$, $n = 0, 1, 2, \ldots, N$, and $i = 1, 2$. Let the probability of no customers in the system be denoted P_{00}.

By using the preceding assumptions and notation the steady state balance equations are obtained directly by

using the "rate out equals rate in" approach described by Cooper [3]. The resulting set of state equations is given as follows:

$$(M\lambda_1 + N\lambda_2)\, P_{00} = P_{101}\mu_1 + P_{012}\mu_2$$

$$[\mu_1 + (M-1)\,\lambda_1 + N\lambda_2]\, P_{101} + P_{00}\, M\lambda_1 + P_{201}\mu_1 + P_{112}\mu_2$$

$$[\mu_1 + (M-m)\,\lambda_1 + N\lambda_2]\, P_{m01} = P_{m-1,0,1}\,(M-m)\lambda_1 + P_{m+1,0,1}\mu_1 + P_{m12}\mu_2 \qquad m = 2,3,\ldots,M-1$$

$$[\mu_1 + N\lambda_2]\, P_{M01} = P_{m-1,0,1}\lambda_1 + P_{M12}\mu_2$$

$$[\mu_2 + M\lambda_1 + (N-1)\,\lambda_2]\, P_{012} = P_{00}\, N\lambda_2 + P_{111}\,\mu_1 + P_{022}\mu_2$$

$$[\mu_2 + M\lambda_1 + (N-n)\lambda_2)\, P_{0n2} = P_{0,n-1,2}\,(N-n+1)\lambda_2 + P_{1n1}\mu_1 + P_{0,n+1,2}\mu_2 \qquad n = 2,3,\ldots,N-1$$

$$[\mu_2 + M\lambda_1]\, P_{0N2} = P_{0,N-1,2}\,\lambda_2 + P_{1N1}\mu_1$$

$$[\mu_1 + (M-m)\lambda_1 + (N-n)\lambda_2]\, P_{mn1} = P_{m-1,n,1}\,(M-m+1)\lambda_1 + P_{m,n-1,1}\,(N-n+1)\lambda_2 + P_{m+1,n,1}\,\mu_1\, q + P_{m,n+1,2}\,\mu_2\, q$$

$$m = 1,2,\ldots,M-1$$

$$n = 1,2,\ldots,N-1$$

$$[\mu_2 + (M-m)\lambda_1 + (N-n)\lambda_2]\ P_{mn2} = P_{m-1,n,2}\ (M-m+1)\lambda_1$$

$$+ P_{m,n-1,2}\ (N-n+1)\lambda_2 + P_{m+1,n,1}\ \mu_1\ (1-q)$$

$$+ P_{m,n+1,2}\ \mu_2\ (1-q)$$

$$m = 1,2,\ldots,M-1$$

$$n = 1,2,\ldots,N-1$$

$$[\mu_1 + (N-n)\lambda_2]\ P_{Mn1} = P_{M-1,n,1}\lambda_1 + P_{M,n-1,1}\ (N-n+1)\lambda_2$$

$$+ P_{M,n+1,2}\ \mu_2\ q$$

$$n = 1,2,\ldots,N-1$$

$$[\mu_2 + (N-n)\lambda_2]\ P_{M,n,2} = P_{M-1,n,2}\ \lambda_1 + P_{M,n-1,2}(N-n+1)\lambda_2$$

$$+ P_{M,n+1,2}\mu_2\ (1-q)$$

$$n = 1,2,\ldots,N-1$$

$$[\mu_1 + (M-m)\lambda_1]\ P_{m,N,1} = P_{m-1N1}\ (M-m+1)\lambda_1 + P_{m,N-1,1}\lambda_2$$

$$+ P_{m+1,N,1}\mu_1\ q$$

$$m = 1,2,\ldots,M-1$$

$$[\mu_2 + (M-m)\lambda_1]\ P_{mN2} = P_{m-1,N,2}\ (M-m+1)\lambda_1 + P_{m,N-1,2}\ \lambda_2$$

$$+ P_{m+1,N,1}\ \mu_1\ (1-q)$$

$$m = 1,2,\ldots,M-1$$

$$\mu_1 P_{MN1} = P_{M-1,N,1}\lambda_1 + P_{M,N-1,1}\lambda_2$$

$$\mu_2 P_{MN2} = P_{M-1,N,2}\lambda_1 + P_{M,N-1,2}\lambda_2$$

It should be noted that some expressions may contain terms P_{0n1} or P_{m02} which are not feasible. When evaluating the equations simply let $P_{0n1} = P_{m02} = 0$.

Service Discipline

Before discussing the method used to solve the system of balance equations, it is necessary to discuss a method for choosing the optimal value of q (as discussed in assumption 3). Cox and Smith [4] have shown for the case of an infinite population, that by changing the priority classification of customers with respect to their mean service times and waiting costs, the cost of waiting can be reduced. The rule used is lower values of the ratio of mean service time and the cost of queuing per unit time should have higher priorities. Applying this rule to our model,

$$q = \begin{cases} 1 & \text{if } \frac{1}{\mu_1 C_{11}} < \frac{1}{\mu_2 C_{12}} \\ 0 & \text{if } \frac{1}{\mu_2 C_{12}} < \frac{1}{\mu_1 C_{11}} \end{cases}$$

If $\frac{1}{\mu_1 C_{11}} = \frac{1}{\mu_2 C_{12}}$, then q can equal 0 or 1. By defining q as above, the customer with the highest priority will

always be the next customer into service.

As will be shown subsequently in Chapter III, the nonpreemptive priority rule given above does not hold for the case of a finite population. Rather, the optimum service discipline appears to be a function of various arrival and service rates, as well as the costs involved. In fact, it appears that the optimum service discipline is dependent upon the state of the system at the time a customer is to be selected for service.

Since a determination of the optimum service discipline for the finite population problem exceeds the scope of the present research effort, a heuristic approach will be employed. Namely, a value of 0.50 is assigned to q, given each customer an equal chance of being the next customer into service. Computational experience to support the choice is presented in Chapter IV.

Numerical Solution Procedure

Developing a closed form solution for the state probabilities is, at best, a formidable task due to the complexity of the system. Therefore an iterative technique has been used to evaluate the state probabilities. The numerical solution procedure used to obtain values for the state probabilities is the Gauss-Siedel iteration method as presented by Cooper [3]. Conte [2]refers to this procedure as the method of successive displacements. The only

difference between the methods is that Cooper normalizes his values to form a density function. This method is simply an iterative procedure, where the present values of the state probabilities are substituted into the balance equations to obtain an updated value for each state probability. As soon as a state probability is updated, this value is then used in solving the next balance equation. Normalization of the probabilities may be performed after each iteration or the normalization may be done only after the final iteration.

The use of this procedure is dependent upon the method's convergence. A sufficient condition for convergence of the procedure is for the matrix of the coefficients of the balance equations to be diagonally dominant or for

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{k} a_{ij} \qquad i = 1, 2, \ldots, k$$

where a_{ij} is an element of the matrix of coefficients and k is the number of rows (columns) in the matrix [3].

Cooper [3] points out that the sufficient condition for convergence is not usually met by the matrix formed by the balance equations in queuing models. However, this does suggest that the greater the concentration of non-zero elements along the main diagonal of the matrix of coefficients, the more likely the procedure will converge.

Consider the previously modelled queuing system with population sizes equal to 2 (M = 2, N = 2). The balance equations for this system are:

$$P_{00} = \frac{P_{101}\mu_1 + P_{012}\,\mu_2}{2\lambda_1 + 2\lambda_2}$$

$$P_{101} = \frac{P_{00}\,2\lambda_1 + P_{201}\,\mu_1 + P_{112}\,\mu_2}{\lambda_1 + 2\lambda_2 + \mu_1}$$

$$P_{012} = \frac{P_{00}\,2\lambda_2 + P_{111}\,\mu_1 + P_{022}\,\mu_2}{2\lambda_1 + \lambda_2 + \mu_2}$$

$$P_{111} = \frac{P_{101}2\lambda_2 + P_{211}\,\mu_1\,q + P_{122}\,\mu_2\,q}{\lambda_1 + \lambda_2 + \mu_1}$$

$$P_{112} = \frac{P_{012}2\lambda_1 + P_{211}\,\mu_1\,(1-q) + P_{122}\,\mu_2\,(1-q)}{\lambda_1 + \lambda_2 + \mu_2}$$

$$\vdots$$

$$P_{221} = \frac{P_{121}\,\lambda_1 + P_{211}\,\lambda_2}{\mu_1}$$

$$P_{222} = \frac{P_{122}\,\lambda_1 + P_{212}\,\lambda_2}{\mu_2}$$

Before the procedure can begin, initial probabilities must be assigned for each state. Due to the many different cases which could arise in the formulation of the model, a uniform distribution of the state probabilities is assumed for

the initial assignment. Let $P_{00}(i)$ denote the value of P_{00} at iteration i. The evaluations of the balance equations are

$$P_{00}^{(i+1)} = \frac{P_{101}^{(i)} \mu_1 + P_{012}^{(i)} \mu_2}{2\lambda_1 + 2\lambda_2}$$

$$P_{101}^{(i+1)} = \frac{P_{00}^{(i+1)} 2\lambda_1 + P_{112}^{(i)} \mu_2 + P_{201}^{(i)} \mu_1}{\lambda_1 + 2\lambda_2 + \mu_1}$$

$$P_{012}^{(i+1)} = \frac{P_{00}^{(i+1)} 2\lambda_2 + P_{111}^{(i)} \mu_1 + P_{022}^{(i)} \mu_2}{2\lambda_1 + \lambda_2 + \mu_2}$$

.
.
.
.
.

$$P_{222}^{(i+1)} + \frac{P_{122}^{(i+1)} \lambda_1 + P_{212}^{(i+1)} \lambda_2}{\mu_2}$$

Once the shut-off criterion has been satisfied, normalization of the probabilities is performed. That is

$$\sum_{i=0}^{M} \sum_{j=0}^{N} \sum_{k=1}^{2} P_{ijk} = 1 .$$

To do this let the final value of the state probability P_{ijk} be

$$P_{ijk} = \frac{P_{ijk}^{(\ell)}}{\sum_{i=0}^{M} \sum_{j=0}^{N} \sum_{k=1}^{2} P_{ijk}^{(\ell)}}$$

where $P_{ijk}^{(\ell)}$ is the state probability obtained from the final iteration.

If desired, the normalization procedure may be done after each iteration. This method requires more computer time and appears necessary only if overflow and/or underflow is a problem in computer programming. For this research, the normalization was done only once, after the final iteration.

The shut-off criterion used to stop the procedure was

$$|P_{ijk}^{(\ell+1)} - P_{ijk}^{(\ell)}| < .005 \quad \text{for all } i, j, k$$

or $\ell > 50$, whichever occurred first. Computational experience has shown that the added accuracy which could be obtained with a stricter shut-off criterion is not justified due to the increased computer time required.

Model of Only One Population

Although the major thrust of this research involves the development of the model presented previously, there are occasions in solving the total problem when a simpler and

computationally more efficient model can be used. This occurs when no customers of a particular type are assigned to the server. The model for the resulting system is simply the (M|M|1) : (GD|k|k) model with one population type. Although this is a general case of the preceding model, closed form solutions have been obtained for this system. Since this is a model for which the solution to the state probabilities is commonly known, only the results will be presented in this paper.

Consider the system where the population size of type 1 customers is equal to M and the population size of type 2 customers is equal to 0. The only state probabilities which are feasible are P_{00} and P_{m01} where m = 1, 2, ..., M. The values for these state probabilities are

$$P_{00} = \left(\sum_{m=0}^{M} \frac{M!}{(M-m)!} \left(\frac{\lambda_1}{\mu_1}\right)^m \right)^{-1}$$

$$P_{m01} = \frac{M!}{(M-m)!} \left(\frac{\lambda_1}{\mu_1}\right)^m \Big/ \sum_{m=0}^{M} \frac{M!}{(M-m)!} \left(\frac{\lambda_1}{\mu_1}\right)^m .$$

By a similar process, values for the state probabilities may be obtained for a system with population of type 1 equal to 0 and N in the population of type 2 customers.

Determination of Values for System

After determining the state probabilities, obtaining values for L and L_q is an easy task. This can be done by applying the rules of expected value for a discrete probability mass function. Expressions for L and L_q in terms of the state probabilities are

$$L_1 = \sum_{m=1}^{M} \sum_{n=0}^{N} \sum_{k=1}^{2} m \; P_{mnk}$$

$$L_2 = \sum_{n=1}^{N} \sum_{m=0}^{M} \sum_{k=1}^{2} n \; P_{mnk}$$

$$L_{q1} = \sum_{m=2}^{M} \sum_{n=0}^{N} (m-1) \; P_{mn1} + \sum_{m=1}^{M} \sum_{n=1}^{N} m \; P_{mn2}$$

and

$$L_{q2} = \sum_{n=2}^{N} \sum_{m=0}^{M} (n-1) \; P_{mn2} + \sum_{n=1}^{N} \sum_{m=1}^{M} n \; P_{mn1}$$

Once these values have been obtained, the given cost function may be evaluated.

Although it is not necessary for the optimization of the given model, there may be some interest in determining other measures of performance for the queuing system. For example, values for the effective arrival rate of each type of customer may be desired. It has been shown, [22], that

the effective arrival rate $\tilde{\lambda}$ may be expressed:

$$\tilde{\lambda} = (L - L_q)\ \mu\ .$$

Applying the above to the model,

$$\tilde{\lambda}_1 = (L_1 - L_{q1})\ \mu_1 \quad \text{and}$$

$$\tilde{\lambda}_2 = (L_2 - L_{q2})\ \mu_2\ .$$

From this the expected waiting time in the system and in the queue can be found for a given customer type. From [22] the expression for the expected waiting time in the system, W, is

$$W = \frac{L}{\tilde{\lambda}}$$

and W_q, the expected waiting time in the queue is

$$W_q = \frac{L_q}{\tilde{\lambda}}\ .$$

Applying the above to our model, we have

$$W_i = \frac{L_i}{\tilde{\lambda}_i} \quad \text{and}$$

$$W_{qi} = \frac{L_{qi}}{\tilde{\lambda}_i}$$

where i = 1, 2.

Although these expressions are not needed for the problem considered in this research, they may be of interest for future research.

Summary

In this chapter the objective function for the proposed problem was defined. The balance equations for the queuing system were developed and a procedure for solving the equations was presented. Finally the expressions for L and L_q were formed, enabling us to evaluate the objective function.

The next logical step is to develop a method of optimizing the objective function. This will be presented in the next chapter.

Chapter III

SOLUTION PROCEDURES

Introduction

The problem addressed in this research is to allocate machines to servers in such a way that an appropriate objective function is optimized. To solve this problem, two procedures have been tried. One involves the use of dynamic programming and the other method is pattern search. Dynamic programming is an exact procedure, meaning an optimal allocation is guaranteed. Conversely, pattern search does not guarantee an optimal allocation. The obvious question now arises, why use a procedure that does not necessarily provide an optimal solution, when another procedure is known to yield the optimum? The answer is that the computational time required for pattern search is much less than the corresponding time needed for dynamic programming. This will be discussed in greater detail in the next chapter.

Dynamic Programming Formulation

The following notation is used in the dynamic programming model:

j = stage variable, the j^{th} server, $j = 1,2,\ldots,c$

λ_i = arrival rate of customer type i

μ_{ij} = service rate of server (stage) j for customer type i

d_{ij} = number of customers of type i assigned to server j

$TC(d_{1j}, d_{2j})$ = cost function as previously defined for server j, where the populations are d_{1j} and d_{2j} respectively

X_{ij} = state variable, total number of customers of type i assigned in stages 1, 2, ..., j

$r_j(X_{1j},X_{2j},d_{1j},d_{2j})$ = individual stage return
$= TC(d_{1j},d_{2j})$

$f^*_j(X_{1j},X_{2j})$ = minimum return through stage j
$= \min_{d_{1j},d_{2j}} f_j(X_{1j},X_{2j},d_{1j},d_{2j})$

The recursive relationship needed to apply dynamic programming to this problem is

$$f_j(X_{1j},X_{2j},d_{1j},d_{2j}) = r_j(X_{1j},X_{2j},d_{1j},d_{2j}) + f^*_{j-1}(X_{1j-1},X_{2j-1})$$

where

$$X_{1j-1} = X_{1j} - d_{1j}$$

$$X_{2j-1} = X_{2j} - d_{2j}$$

It should be noted that

$$d_{1j} \leq X_{1j} \quad \text{and}$$

$$d_{2j} \leq X_{2j} \ .$$

Example Problem

To aid in the understanding of dynamic programming as applied to this problem an example problem will be presented. A problem with small populations sizes was deliberately chosen to show how dynamic programming can be used to determine the optimum allocation of machines to servers. The number of servers (stages) is equal to 3 and each type of machine has a population of size 3. The cost of waiting and service for population type 1 is \$12 per unit time, while the cost of waiting and service for the second group of machines is \$11 per unit time. The fixed cost of the servers is \$8 per unit time for the first and third servers (stages 1 and 3) and \$7 for the second server. The other needed parameters are given in Table 3.1.

Stage 1 Calculations

In stage 1, the decision variables d_{11} and d_{21} are always equal to the state variables, X_{11} and X_{21}. Therefore $f^*(X_{11},X_{21}) = r_1(X_{11},X_{21},d_{11},d_{21})$. The value of the individual stage return, $r_1(X_{11},X_{21},d_{11},d_{21})$, is equal to

$TC(d_{11},d_{21})$. Therefore,

$$r_1(0,0,0,0) = 0.0000$$

$$r_1(0,1,0,1) = 11.8500$$

$$\vdots$$

$$r_1(3,3,3,3) = 54.1983$$

Stage 2 Calculations

The individual stage return function for stage 2, $r_2(X_{12},X_{22},d_{12},d_{22})$, is calculated in the same way as the first stage,

$$r_2(X_{12},X_{22},d_{12},d_{22}) = TC(d_{12},d_{22}) .$$

The relation $f_2(X_{12},X_{22},d_{12},d_{22}) = r_2(X_{12},X_{22},d_{12},d_{22}) + f_1^*(X_{11},X_{21})$

determines the tableau values.

For stage 2

$$f_2(0,0,0,0) = 0.0000 + 0.0000 = 0.0000$$

$$f_2(0,1,0,0) = 0.0000 + 11.8500 = 11.8500$$

$$\vdots$$

$$f_2(3,3,2,3) = 44.3198 + 11.7241 = 56.0439$$

$$f_2(3,3,3,3) = 54.4220 + 0.0000 = 54.4220$$

Stage 3 Calculations

The results for stage 3 are given in Table 3.4. As shown, the minimum cost, 44.7869 is attained when $d_{13} = 0$ and $d_{23} = 3$. By referring to Table 3.3 and using expressions $X_{1j-1} = X_{1j} - d_{1j}$ and $X_{2j-1} = X_{2j} - d_{2j}$, we find the optimal decision variables for stage 2, d_{12} and d_{22}, for $X_{12} = 3$ and $X_{22} = 0$ are $d_{12} = 0$ and $d_{22} = 0$. From this we also get $d_{11} = 3$ and $d_{21} = 0$. Therefore, the optimal allocation of machines is

Server 1 → 3 of type 1, 0 of type 2

Server 2 → 0 of type 1, 0 of type 2

Server 3 → 0 of type 1, 3 of type 2

yielding a cost of 44.7869. This means that server 2 should be placed on another job.

Pattern Search

Determining the optimal allocation of machines to servers using dynamic programming often requires a great deal of computation time and in some cases may require a great deal of storage. Since this time is costly, a pattern search procedure was used to solve the problem under study. Pattern search is an accelerated climbing technique developed by Hooke and Jeeves [11]. The procedure was used in this research for two reasons. First, no evaluations of the

Table 3.1

Parameters for Dynamic Programming Example Problem

	Service Rates μ_{1j}	μ_{2j}	Costs C_{3j}
Server 1	20	13	8
Server 2	15	15	7
Server 3	14	18	8

	Arrival Rate λ_i	Costs C_{1i}	C_{2i}
Machine Type 1	9	12	12
Machine Type 2	7	11	11

Table 3.2

Example Problem - Stage 1

X_{11},X_{21}	$f(X_{11},X_{21},d_{11},d_{21}$	d_{11}^*,d_{21}^*
0,1	0.0000	0,0
0,1	11.8500	0,1
0,2	17.2606	0,2
0,3	24.4319	0,3
1,0	11.7241	1,0
1,1	17.1751	1,1
1,2	24.3923	1,2
1,3	33.2099	1,3
2,0	16.9024	2,0
2,1	24.1927	2,1
2,2	33.1726	2,2
2,3	43.2641	2,3
3,0	23.8188	3,0
3,1	32.9539	3,1
3,2	43.3241	3,2
3,3	54.1983	3,3

Table 3.3

Example Problem - Stage 2

$X_{12}X_{22}/d_{12},d_{22}$	$f(X_{12},X_{22},d_{12},d_{22})$							
	0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3
0,0	0.0000	--	--	--	--	--	--	--
0,1	11.8500	10.5000	--	--	--	--	--	--
0,2	17.2606	22.3500	15.3790	--	--	--	--	--
0,3	24.4319	27.7606	27.2290	21.8894	--	--	--	--
1,0	11.7241	--	--	--	11.5000	--	--	--
1,1	17.1751	22.2241	--	--	23.3500	16.5990	--	--
1,2	24.3923	27.6551	27.1031	--	28.7606	28.4490	23.3082	--
1,3	33.2099	34.8923	32.5541	33.6135	35.9319	33.9596	35.1582	31.6051
2,0	16.9024	--	--	--	23.2241	--	--	--
2,1	24.1927	27.4024	--	--	28.6751	28.3232	--	--
2,2	33.1726	34.6927	32.2814	--	35.8923	33.7741	35.0323	--
2,3	43.2641	43.6726	39.5716	38.7918	44.7099	40.9914	40.4833	43.3292
3,0	23.8188	--	--	--	28.4024	--	--	--
3,1	32.9539	34.3188	--	--	35.6927	33.5014	--	--
3,2	43.3241	43.4539	39.1978	--	44.6727	40.7917	40.2105	--
3,3	54.1983	53.8241	48.3329	45.7082	54.7641	49.7716	47.5008	48.5075

Table 3.3. Example Problem - Stage 2, continued

X_{12},X_{22}/ d_{12},d_{22}	$f(X_{12},X_{22},d_{12},d_{22}$ 2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3	f_2^*	d_{12}^*, d_{22}^*
0,0	--	--	--	--	--	--	--	--	0.0000	0,0
0,1	--	--	--	--	--	--	--	--	10.5000	0,1
0,2	--	--	--	--	--	--	--	--	15,3790	0,2
0,3	--	--	--	--	--	--	--	--	21.8894	0,3
1,0	--	--	--	--	--	--	--	--	11.5000	1,0
1,1	--	--	--	--	--	--	--	--	16.5990	1,1
1,2	--	--	--	--	--	--	--	--	23.3082	1,2
1,3	--	--	--	--	--	--	--	--	31.6051	1,3
2,0	17.8493	--	--	--	--	--	--	--	16.9024	0,0
2,1	29.6993	24.7442	--	--	--	--	--	--	24.1927	0,0
2,2	35.1099	36.8941	33.1331	--	--	--	--	--	33.1331	2,2
2,3	42.2812	42.0047	44.9831	42.7016	--	--	--	--	38.7918	0,3
3,0	29.5434	--	--	--	26.1969	--	--	--	23.8188	0,0
3,1	35.0244	36.4683	--	--	38.0469	34.6668	--	--	32.9539	0,0
3,2	42.2147	41.9193	44.8573		43.4575	46.5168	44.1940	--	39.1978	0,2
3,3	51.0592	49.1365	50.3082	54.4257	50.6288	51.9274	56.0439	54.4220	45.7082	0,3

Table 3.4

Example Problem - Stage 3

$f(X_{13},X_{23},d_{13},d_{23})$

$X_{13},X_{23}/d_{13},d_{23}$	0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3
3,3	45.7082	50.7778	48.2655	44.7869	51.4874	49.6053	47.5591	47.8257

$f(X_{13},X_{23},d_{13},d_{23})$

$X_{13},X_{23}/d_{13},d_{23}$	2,0	2,1	2,2	2,3	3,0	3,1	3,2	3,3	f_3^*	d_{13}^*, d_{23}^*
3,3	50.0737	49.0204	50.0737	53.9528	49.8885	51.2485	55.2776	54.5220	44.7869	0,3

partial derivatives of the objective function need to be performed. Secondly, a problem arises with other techniques with regard to the variables in the objective function. The variables in the expected cost function are not represented explicitly; rather, the variables are reflected implicitly through the operating characteristics, L_i^j and L_{qi}^j. The values of L_i^j and L_{qi}^j result from a given allocation. However, it is difficult to search directly over L_i^j and L_{qi}^j. Pattern search allows the search to be performed using the different allocations as the variables. For these reasons, pattern search was used to optimize the allocation of machines to servers. A detailed discussion of pattern search is provided in [11] and [27].

The variables in the search technique, as applied to this problem, are the number of machines of each type assigned to each server. Again letting x_{ij} represent the number of type i machines assigned to server j, x_{ij} represents the variables in the search. The number of variables to be searched, however, is not 2c, where c is the number of servers available. Instead, the number of variables is $2(c-1)$, since $x_{1c} = M - \sum_{j=1}^{c-1} x_{1j}$ and $x_{2c} = N - \sum_{j=1}^{c-1} x_{2j}$. Thus x_{1c} and x_{2c} are determined by the other variables, x_{1j} and x_{2j}, $j = 1,2,\ldots,c-1$, there is no need to include them in the search. It is important to note, however, that even

though x_{1c} and x_{2c} are not included in the search, the cost resulting from server c must be included in evaluating the expected cost function.

Because pattern search is designed for independent variables, a modification to the search needs to be made. It is possible for x_{1j} and x_{2j}, j = 1, 2, ..., c-1, to be in the feasible range of the problem and still not yield a feasible allocation. This occurs when $x_{1j} \leq M$ and $x_{2j} \leq N$ $\forall_j$, j = 1, 2, ..., c-1, but $\sum_{j=1}^{c-1} x_{1j} > M$ or $\sum_{j=1}^{c-1} x_{2j} > N$. Therefore, the allocation is not feasible and the objective function does not need to be evaluated. If the allocation is not feasible, the objective function is assigned a large value and the search continues, reducing the computational time required.

As mentioned previously, the allocation obtained from pattern search is not always the optimal assignment. Computational experience has shown that the solution is dependent on the starting point. Because of the dependence on the starting point, the user should be cautious of implementing an allocation generated by pattern search if the final solution is at or very near the starting point. If the situation occurs, another starting point should be examined even though it requires additional computation time.

Table 3.5

Parameters for Pattern Search Example Problem

	Service Rate μ_{1j}	Service Rate μ_{2j}	Costs C_{3j}
Server 1	150	120	80
Server 2	130	130	80
Server 3	140	140	80

	Arrival Rate λ_i	Costs C_{1i}	Costs C_{2i}
Machine Type 1	19	90	50
Machine Type 2	13	80	50

M = 10

N = 10

Table 3.6

Results of Example Problem Using Pattern Search

x_{11}	x_{21}	x_{12}	x_{22}	x_{13}	x_{23}	$TC(x_{1j}, x_{2j})$
3	3	3	3	4	4	496.2737
5	3	3	3	2	4	496.0718
5	1	3	3	2	6	488.8645
7	0	3	3	0	7	484.4229

Final Solution

$x_{11} = 7$ $x_{12} = 3$ $x_{13} = 0$

$x_{21} = 0$ $x_{22} = 3$ $x_{23} = 7$

$$f(x_{1j},x_{2j}) = 484.4229$$

True optimum 484.3503

NOTE: Only improved evaluations for search procedure are shown.

Other Solution Procedures

Another solution procedure which could be used for the problem under study is the Kiefer-Wolfowitz procedure as modified by Sacks [20] for the multidimensional case. This technique was developed to optimize systems with variation in the objective function due to randomness, such as in simulation. But it can also be used for the problem under study.

In using the procedure evaluations at 4(c-1) points must be made about an initially selected point (x_{11}, x_{21}, x_{12}, x_{22},...,$x_{1,c-1}$, x_{2c-1}). The evaluations are made at points ($x_{11}^0 + b_1$, $x_{21}^0 \cdots$ $x_{1,c-1}$, $x_{2,c-1}$), ($x_{11}^0 - b_1$, x_{21}^0, $\cdots$ $x_{1,c-1}^0$, $x_{2,c-1}$) .... (x_{11}^0, $x_{21}^0 \cdots$ $x_{1,c-1}^0$, $x_{2,c-1}^0 - b_1$) until all 4(c-1) points have been evaluated. If we let $f(x_{11}, x_{21}, \ldots, x_{2,c-1})$ represent the functional evaluation for the point (x_{11}, x_{21}, $\cdots$ $x_{2,c-1}$) the next center point for the search (x_{11}^1, X_{21}^1, $\cdots$ $x_{2,c-1}^1$) is determined by

$$x_{1j}^{n+1} = \frac{a_n[f(x_{11}^n, \cdots x_{1j}^n + b_n, \cdots x_{1,c-1}^n) - f(x_{11}^n, \cdots x_{1j}^n - b_n, \cdots x_{1,c-1}^n)]}{2b_n}$$

where a_n is a constant of proportionality. Certain restrictions are placed on a_n and b_n for this procedure.

These restrictions are

$$\lim_{n\to\infty} a_n = 0$$

$$\sum_{n=1}^{\infty} a_n = \infty$$

$$\lim_{n\to\infty} b_n = 0$$

$$\sum_{n=1}^{\infty} \left(\frac{a_n}{b_n}\right)^2 < \infty$$

The procedure requires 4(c-1) evaluations at each iteration. This is a large number of evaluations and could possibly result in a great deal of computational time. Detailed discussions of the procedure are found in [10], [20], and [26].

Summary

This chapter described two solution procedures used in the optimization of the proposed problem. The two procedures, dynamic programming and pattern search both have advantages which must be considered before deciding which is the best method to use. A third procedure, Kiefer-Wolfowitz, was presented, although it was not used in the research. The next chapter discusses the computational results for the research and qualitatively compares the two procedures used to solve the problem under study.

Chapter IV

COMPUTATIONAL RESULTS AND SENSITIVITY ANALYSIS

Introduction

This chapter examines the computational results obtained for optimizing the expected cost function. The problem of determining an optimal priority rule is discussed and a counter example is given for the priority rule presented by Cox and Smith [4] for infinite population queues when applied to finite population queues. The two solution procedures presented in Chapter III are compared and, finally, a sensitivity analysis is performed to determine the effect of changes in the arrival and service rates on the objective function. Inferences concerning the shape of the objective function in a region near the optimum are also discussed.

Service Priority Discipline

In Chapter II, a discussion was given concerning the problem of deciding which type of customer to serve next, once a service was completed and both types of customers remained in the subsystem. The rule given by Cox and Smith [4] for the infinite population queue was found not to be optimal for the problem under study. For example, for a subsystem with the following parameters,

$$\lambda_1 = 15 \qquad \lambda_2 = 10$$

$$\mu_1 = 175 \qquad \mu_2 = 100$$

$$M = 12 \qquad N = 8$$

The expected system and queue lengths for given values of q were

$q = 0$, $L_1 = 8.0474$ $L_{q1} = 7.6898$

$L_2 = 1.6284$ $L_{q2} = 0.9911$

$q = 1$, $L_1 = 3.2010$ $L_{q1} = 2.4351$

$L_2 = 5.5048$ $L_{q2} = 5.2755$

If we let

$$C_{11} = C_{12} = 1,$$

$$C_{12} = C_{22} = 1.7, \text{ and}$$

$$C_{31} = C_{32} = 5$$

we obtain objective function values of 15.8156 for q equal to zero and 17.5592 for q equal to one, which says that the better priority rule is for type 2 to be served before type 1. But by applying the rule presented by Cox and Smith [4] we have

$$\frac{1}{\mu_1 C_{11}} = \frac{1}{175(1)} = \frac{1}{175}$$

and

$$\frac{1}{\mu_2 C_{12}} = \frac{1}{100(1.7)} = \frac{1}{170}$$

Therefore, since $\frac{1}{\mu_1 C_{11}} < \frac{1}{\mu_2 C_{12}}$, a customer of type 1 should have priority. Thus the priority rule for infinite population queues does not hold for finite population queues. Furthermore, it is felt that the optimal priority rule is state dependent. However, rather than search for the optimum sequence of values for q, a constant value of q was assigned for purposes of this research. Specifically, q was set equal to one-half throughout the research. This is a compromise because the optimal value of q will be 0 or 1 for a given state of the subsystem. Although q equal to 0.5 might not be the best value neither is it likely to be the worst value.

Computational Results

In Chapter III, the use of dynamic programming and pattern search as solution procedures was discussed. As previously stated dynamic programming is an exact procedure, but pattern search is a heuristic procedure. The major advantage of pattern search is it requires less computational time. To aid in determining which solution procedure is better, fifty sample problems were generated randomly. The

necessary parameters, arrival rates, service rates, costs and population sizes, were generated using a uniform distribution. In all cases 3 servers were used.

As expected, the time required for pattern search was much less than the time needed for dynamic programming to solve the same problems. Pattern search required an average of 11.3 seconds, as compared to an average of 28.1 seconds for dynamic programming using an IBM 370/158 computer. Since pattern search is much faster than the dynamic programming procedure, the next question is, how does the accuracy of the two procedures compare? In the problems solved, pattern search obtained the true optimum 48 per cent of the time. In 88 per cent of the problems, the cost of the allocation generated by pattern search was within 6 per cent of the cost of the allocation produced by dynamic programming. Table 4.1 and Figure 4.1 provide the results obtained using these procedures. In Table 4.1 and Figure 4.1 r represents the ratio of the cost obtained using pattern search to the cost computed using dynamic programming.

In deciding which procedure is better, the decision is a function of the desires of the decision maker and the cost of computation time. If computer costs are not time dependent, computation time is of little significance. This could occur if a company leased a computer at a fixed rate.

Table 4.1

Dynamic Programming vs. Pattern Search

r	Relative Frequency	Cumulative Frequency
1.00	.48	.48
1.00-1.01	.06	.54
1.01-1.02	.08	.62
1.02-1.03	.04	.66
1.03-1.04	.12	.78
1.04-1.05	.08	.86
1.05-1.06	.02	.88
1.06-1.07	.00	.88
1.07-1.08	.02	.90
1.08-1.09	.04	.94
> 1.09	.06	1.00

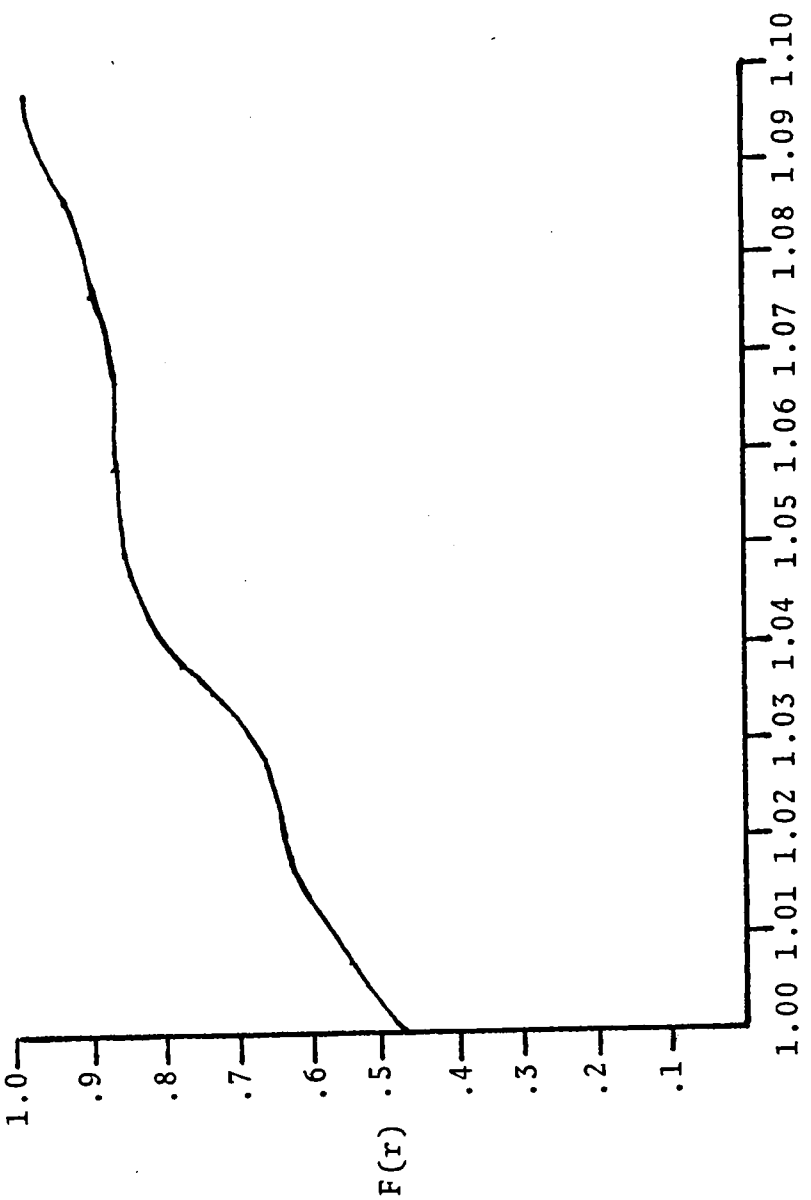

Figure 4.1

Cumulative Frequency Distribution for Comparison of Dynamic Programming and Pattern Search

If computation time is of little importance, then dynamic programming would be the better method. Conversely, if the costs for computation are time dependent, the manager must ask himself how much am I willing to pay for the added accuracy. In many cases, the incremental costs of computation time may be more than the amount saved by using an exact procedure. In this case, pattern search is better.

Sensitivity Analysis

In practice, the values obtained for λ_i and μ_{ij} are often estimates of the true values of λ_i and μ_{ij}. Because these are estimates and not the true values, they will have an effect on the cost function for the model under study. How differing values of λ_i and μ_{ij} affect the assignment of machines to servers and in turn affect the expected cost model is important to a user of this model. If the model is insensitive to these values, comparatively little time needs to be spent gathering data than if the model is very sensitive to changes in λ_i and μ_{ij}.

The measure for the sensitivity analysis will be the ratio of the cost of the assignment using the estimated values to the cost of the allocation using the true values. For discussion purposes, let $\hat{f}(x_{1j}, x_{2j})$ equal the expected cost of the assignment based on the estimated values of the parameters, when the cost of the assignment is evaluated

using the true values of the parameters. Also, let $f^*(x_{1j}, x_{2j})$ equal the optimum expected cost using the true parameter values. The value of importance is the comparison between $\hat{f}(x_{1j}, x_{2j})$ and $f^*(x_{1j}, x_{2j})$ denoted by ϕ, where $\phi = \hat{f}(x_{1j}, x_{2j})/f^*(x_{1j}, x_{2j})$. ϕ represents the penalty incurred by using the estimated values.

Another important aspect of the sensitivity analysis is that it will help in determining if the pattern search solution procedure is a good method, even though it is not an exact procedure. Certainly, if by performing a sensitivity analysis, it can be determined that small deviations from the optimal assignment produce very little change in cost, it is reasonable to assume that the pattern search may be a very good solution procedure.

Conditions of Sensitivity Analysis

Since we are considering the costs incurred by estimating the values of λ_i and μ_{ij}, it is reasonable to think of λ_i and μ_{ij} as random variables having some type of distribution about their true value. The distribution is dependent upon the skill of the person doing the estimating and the amount of data which is collected in developing the estimates.

For purposes of this paper, the value of λ_i will be uniformly varied between $.8\bar{\lambda}_i$ and $1.2\bar{\lambda}_i$, where λ_i is the

estimate of the true arrival rate $\bar{\lambda}_i$. Similarly, μ_{ij} varies uniformly between $.8\bar{\mu}_{ij}$ and $1.2\bar{\mu}_{ij}$. The use of a uniform distribution is motivated from the fact that it will provide an overstatement of the sensitivity of the model. For example, if the model is found to be insensitive to this distribution it should be even more insensitive to, say, a normal distribution.

Sensitivity to Arrival and Service Rate

To determine the sensitivity of the model to the arrival rates and the service rates, 1000 problems were generated from the previously mentioned distributions for λ_i and μ_{ij}. Population sizes of 20 for each type customer were used and 3 servers were available. This was a large enough problem to allow for the changes in λ_i and μ_{ij} to have an affect on the assignments. The values for the other parameters were

$$\lambda_1 = 15$$

$$\lambda_2 = 10$$

$$\mu_{11} = 175 \qquad \mu_{21} = 100$$

$$\mu_{12} = 150 \qquad \mu_{22} = 150$$

$$\mu_{13} = 110 \qquad \mu_{23} = 160$$

$$C_{11} = C_{12} = 7$$
$$C_{21} = C_{22} = 10$$
$$C_{31} = C_{32} = C_{33} = 10$$

Table 4.2 shows the relative and cumulative frequency functions as determined from solving the 1000 problems. The most important result from this analysis is that 94.5 per cent of the allocations have costs within 9 per cent of the cost obtained by using the optimal allocation of machines to servers when using the true parameters.

Interpreting Results

The results of the sensitivity analysis are important in two respects. First, since almost 50 per cent (49.6 per cent) of the allocations had costs which were within 2 per cent of the cost of the optimal assignment and 94.5 per cent had costs within 9 per cent of the cost of the optimal assignment, the model is relatively insensitive to changes in λ_i and μ_{ij}. This is significant in using the model developed in practice, since determining values of λ_i and μ_{ij} will be less of a problem. Certainly it should not be necessary to spend a great deal of money in determining estimates for these values.

The other important result from the analysis concerns the use of an exact procedure versus a heuristic to determine

Table 4.2

Sensitivity to Arrival and Service Rates

$$.8\bar{\lambda}_i \leq \lambda_i \leq 1.2\bar{\lambda}_i$$

$$.8\bar{\mu}_{ij} \leq \mu_{ij} \leq .8\bar{\mu}_{ij}$$

ϕ	Relative Frequency	Cumulative Frequency
1.00	.072	.072
1.00-1.01	.286	.358
1.01-1.02	.138	.496
1.02-1.03	.123	.619
1.03-1.04	.077	.696
1.04-1.05	.106	.802
1.05-1.06	.069	.871
1.06-1.07	.039	.910
1.07-1.08	.026	.936
1.08-1.09	.009	.945
1.09-1.10	.037	.982
1.10-1.11	.011	.993
1.11-1.12	.004	.997
1.12-1.13	.001	.998
> 1.13	.002	1.000

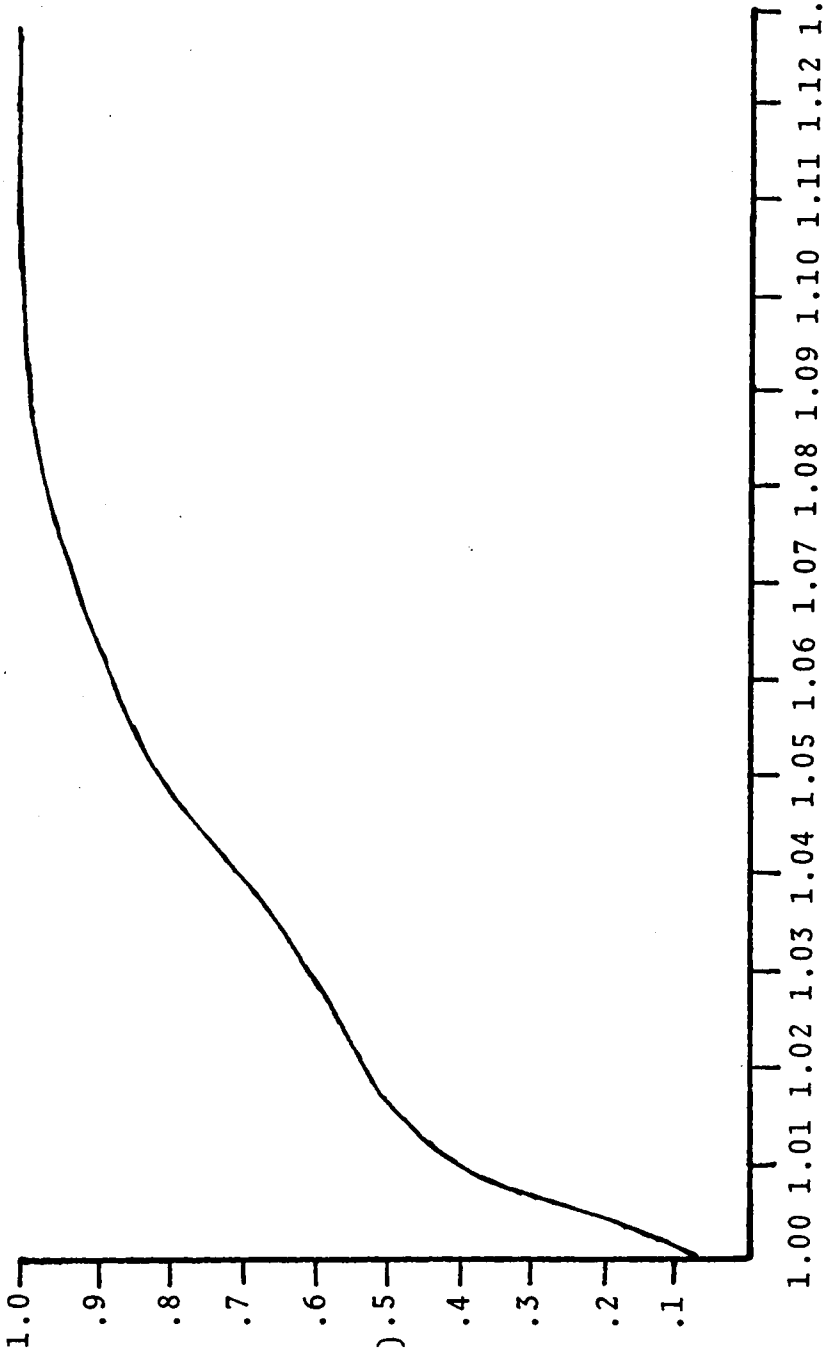

Figure 4.2

Cumulative Frequency Distribution for Variations in Costs of Assignments due to Random Assignments of λ_i and μ_{ij}.

the desired allocation. The results show that the expected cost function is basically flat in the region near the optimum and that very little savings may be obtained by using an exact procedure. In the case of dynamic programming versus pattern search, the use of the exact procedure may not be justified. Since pattern search requires more computation time, the cost of the additional time could be more than the savings in using the exact procedure.

Summary

The computational results for the research were presented in this chapter. A rationale for the priority rule used in the research was also discussed. Finally, the sensitivity of the objective function to changes in the arrival and service rates was determined providing an answer as to why a heuristic procedure, such as pattern search, provides such accurate results.

Chapter V

SUMMARY, RESULTS AND RECOMMENDATIONS FOR FURTHER RESEARCH

Summary

The objectives of this research included the development of an expected cost model for the problem of allocating machines to servers for service required due to machine failure, as well as the development of a solution procedure to optimize the expected cost model.

In accomplishing the first objective, developing an expected cost model, the costs incurred due to machine failure and the ensuing service were analyzed. A method of determining values for the variables in the objective function, L_i^j and L_{qi}^j, was then developed. This determination was accomplished by developing the needed steady state balance equations for the single server, finite population subsystems with two distinct populations. The solution to the state probabilities was found by use of an iterative procedure and the needed parameters, L_i^j and L_{qi}^j, were then evaluated.

After obtaining a method of evaluating the expected cost function, a procedure for minimizing the cost was developed. Two solution procedures were examined. The first, dynamic programming, is an exact procedure. Although this procedure guaranteed an optimal allocation of

machines to servers, the amount of computational time required to produce the optimal assignment was very large. Therefore, a second procedure was examined. Pattern search was used as an alternative means of optimizing the given objective function. This procedure, although not exact, produced results which were very close to the results obtained by using dynamic programming, but required less computational time.

Finally the effect of varying the arrival and service rates was studied. A sensitivity analysis was performed by allowing the arrival and service rates for the problem to vary simultaneously. A uniform probability distribution was used to generate the values for both the arrival and service rates. This analysis was used not only to analyze the effect of the rates, but also to help justify the use of a heuristic procedure, such as pattern search, to optimize the objective function. Upon completing these tests, the research objectives were satisfied.

Results

The first result obtained was the development of a model describing the expected costs incurred due to machine failure and the consequent repair. Solution procedures to optimize the model were developed and programmed in Fortran IV and are presented in Appendix I and II.

The second result involved the conclusion provided by computational analysis. First, the objective function was found to be very flat in the region of the optimum assignment. Therefore, small fluctuations in assignments from the optimum produce very little change in the objective function. Also the optimum assignments were found to be very intuitive. This, plus the fact that the objective function is very flat near the optimum, leads one to believe that a person with experience in making the assignments probably could do an adequate job without the use of a sophisticated solution procedure. This would be especially true if results were available for a variety of problems. Therefore, in using this model in practice, it might be better to provide a person with information concerning the optimal assignments for a wide range of problems and allow him to make the assignment of machines to servers by intuition for a specific problem. This would be done in lieu of using one of the solution procedures every time an assignment was required.

Finally, it was found that the rule for priorities of service presented by Cox and Smith [4] for the infinite population system was not optimal for the finite population queuing system. It is a subjective belief that the optimal rule may be state dependent.

Recommendations for Further Research

Several possible extensions to this research have been recognized. These extensions are listed below.

1. Extend the current model to include costs associated with a server travelling to a machine to provide service. Include in this extension a comparison of pooling of service and allocating specific machines to servers. Also develop a procedure for determining which method is optimal for a given problem.

2. Develop a model for the problem having more than two arrival populations with non-identical arrival and service rates.

3. Develop a method for obtaining an optimal priority rule which would determine which customer type should be served next upon completion of a service.

4. Extend the model to optimize the objective function under certain constraints, such as a constraint on the waiting time of a customer.

5. Extend the model to include not only the allocation of machines to servers but also to determine the location of the machines in order to minimize a cost function which includes waiting, service and travel costs.

6. Extend the model to optimize the given objective where the interarrival and service time distributions are

non-Poisson. This could be accomplished by using a search procedure such as the Kiefer-Wolfowitz to optimize the cost function, where the needed variables are obtained by simulation.

7. Develop a series of graphs showing the values of L_i^j and L_{qi}^j for different arrival and service rates and different population sizes.

8. Implement the model suggested in the research effort. By using actual operating information the model could be tested and refined to more accurately represent an actual operating system.

BIBLIOGRAPHY

1. Ashcroft, H., "The Productivity of Several Machines Under the Care of One Operator," Royal Statistical Society Journal, Vol. 12 (1950), p. 145.

2. Conte, S. D., Elementary Numerical Analysis, McGraw-Hill Book Company, New York (1965).

3. Cooper, R. B., Introduction to Queuing Theory, The Macmillan Company, New York (1972).

4. Cox, D. R. and Smith, W. L., Queues, Methuen and Co., Ltd., London (1961).

5. Fetter, R. N., "The Assignment of Operators to Service Automatic Machines," The Journal of Industrial Engineering, Vol. VI, No. 5 (Sept.-Oct., 1955), pp. 22-30.

6. _____, and Galliher, H. P., "Waiting Line Models in Materials Handling," The Journal of Industrial Engineering, Vol. IX, No. 3 (May-June, 1958), pp. 202-208.

7. Hillier, F. S., "Economic Models for Industrial Waiting Line Problems," Management Science, Vol. 10, No. 1 (Oct., 1963), pp. 119-130.

8. _____, "The Application of Industrial Waiting Line Theory to Industrial Problems," The Journal of Industrial Engineering, Vol. XV, No. 1 (Jan.-Feb., 1964), pp. 3-8.

9. _____, "Cost Models for the Application of Priority Waiting Line Theory to Industrial Problems," The Journal of Industrial Engineering, Vol. XVI, No. 3 (May-June, 1965).

10. Hersh, M., "Allocation Processes with Variable Channel Queues," Management Science, Vol. 20, No. 2, (Oct., 1973), pp. 203-213.

11. Hooke, J. A. and Jeeves, T. A., "'Direct Search' Solution of Numerical and Statistical Problems," Journal of the Association for Computing Machinery, Vol. 8, No. 2 (April, 1961), pp. 212-229.

12. Hooke, J. A., "A Priority Queue with Low Priority Arrivals," Operations Research, Vol. 20, No. 2 (March-April, 1972), pp. 373-380.

13. _____, "Some Heavy-Traffic Limit Theorems for a Priority Queue with General Arrivals," Operations Research, Vol. 20, No. 2 (March-April, 1972), pp. 381-388.

14. King, J. R., "On the Optimum Size of Workforce Engaged in the Servicing of Automatic Machines," The International Journal of Production Research, Vol. 8, No. 3 (1970), pp. 207-220.

15. Kredentser, B. P., "On the Optimal Loading of a Two-Priority Service System with Waiting," Avotamatika I Telemkhanika, Vol. 9 (1970), pp. 145-150.

16. Mangelsdorf, T. M., "Waiting Line Theory Applied to Manufacturing Problems," S. M. Thesis, M.I.T., (1955), Reprinted in Analysis of Industrial Operations, edited by Bowman, E. H. and Fetter, R. B., Richard D, Irwin, Homewood, Illinois (1959).

17. Morris, W. T., The Analysis of Management Decisions, Richard D. Irwin, Inc., Homewood, Illinois, (1964).

18. Naor, P., "On Machine Interference," Journal of Royal Statistical Society, Series B (1956), pp. 280-287.

19. Palm, D. C., "The Assignment of Workers in Servicing Automatic Machines," The Journal of Industrial Engineering, Vol. IX, No. 1 (Jan.-Feb., 1958), pp. 28-42.

20. Sacks, J., "Asymptotic Distribution of Stochastic Approximation Procedures," Annals of Mathematical Statistics, Vol. 29 (1958), pp. 397-398.

21. Smith, W., "The Infinitely-Many Server Queue with Semi-Markovian Arrivals and Customer Dependent Exponential Service Times," Operations Research, Vol. 20, No. 4, (July-August, 1972), pp. 907-912.

22. Stidham, S., "A Last Word on $L = \lambda W$," Operations Research, Vol. 22, No. 2 (March-April, 1974) pp. 417-421.

23. Taha, H. A., Operations Research, The Macmillan Co., New York (1971).

24. _____, "A Case Study Comparison of Independent Channels Versus a Combined Pool," The Journal of Industrial Engineering, Vol. XIX, No. 3 (March, 1968), pp. 137-143.

25. White, J. A., Schmidt, J. W., Bennett, G. K., The Analysis of Queuing Systems, Academic Press (to be published).

26. Wilde, D. J., Optimum Seeking Methods, Prentice-Hall, Inc., Englewood Cliffs, N. J., (1964).

27. _____, and Beightler, C. S., Foundations of Optimization, Prentice-Hall, Inc., Englewood Cliffs, N. J., (1967).

APPENDIX I

Appendix I concerns the limitations and the use of the dynamic programming procedure developed to optimize the problem under study. There are two basic limitations to the program. First, the maximum number of servers to be allocated machines is nine. Also the maximum population size for each customer type is twenty-four. These limitations can be overcome by increasing the array sizes in the dimension statements. The storage needed for this program is approximately 110 K. The user should specify a region of 130 K for use on the Fortran G compiler for an IBM 370/158 system.

The following statements explain the use of the program.

CARD	FORMAT	DESCRIPTION
1	I5	Number of problems to be solved
2	I5	Number of servers (c)
3	2I5	Number of customers in each population; Population 1 always precedes population 2 (M,N)
4	2F5.0	Arrival rates for each population (λ_1,λ_2)
5 thru 4+c	2F5.0	Service rates for each server for each population; Service rates for server 1 are always on separate card from other servers (μ_{1j},μ_{2j})

CARD	FORMAT	DESCRIPTION
5+c	cF5.0	Fixed costs for each server (C_{31}, C_{32}, ... C_{3c})
6+c	4F5.0	Costs of servicing each population and costs of waiting for each population (C_{21}, C_{22}, C_{11}, C_{12})

Repeat steps 2 thru 6+c for multiple problems, where c is the number of servers for each problem.

To also aid in using the program, a list of the main variables is included.

VARIABLE NAME	USE
STAGE (I,J,K)	Minimum return value for stage K and state I-1 and J-1
ISAVE (I,J,K)	Optimum decision variable for customer type 1 for stage K and state I-1 and J-1.
JSAVE (I,J,K)	Optimum decision variable for customer type 2 for stage K and state I-1 and J-1.
VAL (I,J)	Value of $r(X_{1j}, X_{2j}, d_{1j}, d_{2j})$ for the stage presently being considered where I-1 = d_{1j} and J-1 = d_{2j}.
ALAM (I)	Arrival rate for customer type I.
AMU (J,I)	Service rate of server J for customer type I.
NSERV	Number of servers being considered.
MMAX	Total number of customers of type 1.
NMAX	Total number of customers of type 2.

VARIABLE NAME	USE
CMAN (J)	Fixed cost of server J, C_{3j}.
CSERV (I)	Cost of servicing customer I, C_{2i}.
CQUE (I)	Cost of waiting for customer I, C_{1i}.
P (I,J,K)	Probability of I-1 customers of type 1 and J-1 customers of type 2 in the system and customer K is presently in service.
MEAN (I,1)	Expected number of type I customers in the system.
MEAN (I,2)	Expected number of type I customers in the queue.
NPROB	Number of problems to be solved.

```
      COMMON/BLK/NSERV
      COMMON AMU,ALAM
      DIMENSION VAL(25,25)
      DIMENSION STAGE(25,25,9),AMU(9,2),ALAM(2),ISAVE(25,25,9),JSAVE(25,
     125,9)
C     ***************************************************************
C     READ THE NUMBER OF PROBLEMS,NUMBER OF SERVERS,ARRIVAL RATE FOR EACH
C     POPULATION AND NUMBER IN EACH POPULATION
C     ***************************************************************
      READ 100,NPROB
      DO 1000 KXY=1,NPROB
      READ 100,NSERV
      READ 100,MMAX,NMAX
      READ 101,ALAM(1),ALAM(2)
      WRITE(6,100) NSERV
      WRITE(6,100) MMAX,NMAX
      WRITE(6,304) ALAM(1),ALAM(2)
C     ***************************************************************
C     READ SERVICE RATES FOR EACH SERVER
C     ***************************************************************
      DO 15 I=1,NSERV
      READ 101,AMU(I,1),AMU(I,2)
   15 WRITE(6,304) AMU(I,1),AMU(I,2)
      CALL SUBSYS(M,N,YY,0)
      MM=MMAX+1
      NN=NMAX+1
C     ***************************************************************
C     CALCULATE THE STAGE RETURN VALUES FOR EACH STATE IN STAGE ONE
C     ***************************************************************
      DO 10 I=1,MM
      DO 10 J=1,NN
      MS=I-1
```

```
      NS=J-1
      CALL SUBSYS(MS,NS,YY,1)
      ISAVE(I,J,1)=MS
      JSAVE(I,J,1)=NS
   10 STAGE(I,J,1)=YY
      NSERV1=NSERV-1
      IF(NSERV.EQ.2) GO TO 60
C  ***************************************************************************
C  THIS DO LOOP PERFORMS THE NEEDED CALCULATIONS FOR STAGES 2 THRU C-1
C  ***************************************************************************
      DO 51 IA=2,NSERV1
      IB=IA-1
C  ***************************************************************************
C  CALCULATE THE RETURN VALUES FOR ALL POSSIBLE COMBINATIONS
C  OF THE DECISIONS VARIABLES
C  ***************************************************************************
      DO 50 II=1,MM
      DO 50 JJ=1,NN
      MS=II-1
      NS=JJ-1
      CALL SUBSYS(MS,NS,YY,IA)
   50 VAL(II,JJ)=YY
C  ***************************************************************************
C  CALCULATE THE MINIMUM STAGE RETURN VALUE FOR EACH STATE
C  ***************************************************************************
      DO 11 I=1,MM
      DO 11 J=1,NN
      OP=10.**10
      DO 12 II=1,I
      DO 12 JJ=1,J
      III=I-II+1
      JJJ=J-JJ+1
```

```
      CHECK=VAL(II,JJ)+STAGE(III,JJJ,IB)
      IF(CHECK.GT.OP) GO TO 12
      OP=CHECK
      IC=II-1
      JC=JJ-1
      ID=III-1
      JD=JJJ-1
   12 CONTINUE
C  ************************************************************************
C  STORE THE OPTIMUM DECISION VARIABLES FOR EACH STATE
C  ************************************************************************
      ISAVE(I,J,IA)=IC
      JSAVE(I,J,IA)=JC
   53 STAGE(I,J,IA)=OP
   11 CONTINUE
   51 CONTINUE
C  ************************************************************************
C  CALCULATE THE MINIMUM STAGE RETURN VALUE FOR THE FINAL STAGE
C  ************************************************************************
   60 OP=10.**10
      DO 14 I=1,MM
      DO 14 J=1,NN
      JM=NN-J+1
      IM=MM-I+1
      MS=I-1
      NS=J-1
      CALL SUBSYS(MS,NS,YY,NSERV)
      II=MM-MS
      JJ=NN-NS
      CHECK=YY+STAGE(II,JJ,NSERV1)
      WRITE(6,302) MS,NS,CHECK
      IF(CHECK.GT.OP) GO TO 14
```

```
      OP=CHECK
C  ***************************************************************************
C  STORE THE OPTIMUM DECISION VARIABLES FOR THE FINAL STAGE
C  ***************************************************************************
      KI=MS
      KJ=NS
      LI=IM
      LJ=JM
   14 CONTINUE
      OP1=OP
C  ***************************************************************************
C  PRINT THE OPTIMAL ASSIGNMENTS FOR EACH SERVER AND THE OPTIMAL
C  OBJECTIVE FUNCTION VALUE
C  ***************************************************************************
      WRITE(6,301) NSERV,KI,KJ
      DO 54 K=1,NSERV1
      KK=NSERV-K
      I=MM-KI
      J=NN-KJ
      KI=KI+ISAVE(I,J,KK)
      KJ=KJ+JSAVE(I,J,KK)
   54 WRITE(6,301) KK,ISAVE(I,J,KK),JSAVE(I,J,KK)
   20 WRITE(6,300) OP1
 1000 CONTINUE
      STOP
  304 FORMAT(' ',2X,2F6.1)
  302 FORMAT(' ',I4,2X,I4,2X,F12.4)
  300 FORMAT(' ','OPTIMUM = ',2X,F12.4)
  301 FORMAT(' ','ASSIGN TO SERVER ',2X,I3,1X,I2,1X,'OF TYPE 1 AND ',2X,
     *I2,1X,' OF TYPE 2')
  101 FORMAT(4F5.0)
  100 FORMAT(4I5)
```

END

```
      SUBROUTINE SUBSYS(M,N,YY,LM)
      REAL*4 MEAN(2,2)
      COMMON/BLK/NSERV
      COMMON BMU,ALAM
      DIMENSION P(25,25,2)
      DIMENSION CMAN( 9),CSERV(2),CQUE(2),BMU(9,2),ALAM(2),AMU(2)
      IF(LM.GT.0) GO TO 65
C     ***************************************************************
C     READ THE COSTS FOR THE PROBLEM
C     ***************************************************************
      READ 400,(CMAN(I),I=1,NSERV)
      WRITE(6,400) (CMAN(I),I=1,NSERV)
      READ 400,CSERV(1),CSERV(2),CQUE(1),CQUE(2)
      WRITE(6,400) CSERV(1),CSERV(2),CQUE(1),CQUE(2)
      GO TO 85
   65 MMM=0
      NNN=0
      YY=0.
C     ***************************************************************
C     SET THE SERVICE RATES FOR THE SERVER BEING CALCULATED
C     ***************************************************************
      AMU(1)=BMU(LM,1)
      AMU(2)=BMU(LM,2)
C     ***************************************************************
C     CHECK TO SEE IF BOTH POPULATION SIZES ARE EQUAL TO 0 OR IF ONE
C     POPULATION SIZE EQUALS 0
C     ***************************************************************
   88 IF(M.EQ.0.AND.N.EQ.0) GO TO 86
      IF(M.EQ.0) GO TO 76
      IF(N.EQ.0) GO TO 77
      AN=N
      AM=M
```

```
      IT=0
      NN=N+1
      MM=M+1
C     ***************************************************************************
C     SET INITIAL PROBABILITIES FOR EACH STATE
C     ***************************************************************************
      DO 12 I=1,MM
      DO 12 J=1,NN
      DO 12 K=1,2
   12 P(I,J,K)=1./(AM*AN*2.)
C     ***************************************************************************
C     SET INFEASIBLE PROBABILITIES TO 0
C     ***************************************************************************
      DO 10 I=1,MM
   10 P(I,1,2)=0.
      DO 11 I=2,NN
   11 P(1,I,1)=0.
   51 CONTINUE
C     ***************************************************************************
C     CALCULATE STATE PROBABILITY FOR P(0,0) AND CALCULATE THE DIFFERENCE
C     BETWEEN VALUE FOR ITERATION J+1 AND J
C     ***************************************************************************
      CHECK=P(1,1,1)
      P(1,1,1)=(P(2,1,1)*AMU(1)+P(1,2,2)*AMU(2))/(AM*ALAM(1)+AN*ALAM(2))
      SUM=ABS(CHECK-P(1,1,1))
      P(1,1,2)=P(1,1,1)
C     ***************************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(I,0,1) WHERE I=1,.....M AND COMPUTE
C     THE DIFFERENCE BETWEEN VALUE FOR ITERATION J+1 AND J
C     ***************************************************************************
   55 IF(M.EQ.1) GO TO 53
      DO 1 I=2,M
```

```
      AI=I-1
      II=I+1
      III=I-1
      CHECK=P(I,1,1)
      P(I,1,1)=(P(II,1,1)*AMU(1)+P(I,2,2)*AMU(2)+P(III,1,1)*ALAM(1)*
     1(AM-AI+1.))/((AM-AI)*ALAM(1)+AN*ALAM(2)+AMU(1))
      SU=ABS(CHECK-P(I,1,1))
      IF(SU.GT.SUM) SUM=SU
    1 CONTINUE
   53 I=M+1
      CHECK=P(I,1,1)
      P(I,1,1)=(P(I,2,2)*AMU(2)+P(M,1,1)*ALAM(1))/(AN*ALAM(2)+AMU(1))
      SU=ABS(CHECK-P(I,1,1))
      IF(SU.GT.SUM) SUM=SU
C     ************************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(0,I,2) WHERE I=1,.....N AND COMPUTE
C     THE DIFFERENCE BETWEEN THE VALUE FOR ITERATION J+1 AND J
C     ************************************************************************
      IF(N.EQ.1) GO TO 52
      DO 2 I=2,N
      AI=I-1
      II=I+1
      III=I-1
      CHECK=P(1,I,2)
      P(1,I,2)=(P(2,I,1)*AMU(1)+P(1,II,2)*AMU(2)+P(1,III,2)*ALAM(2)*
     1(AN-AI+1.))/(AM*ALAM(1)+(AN-AI)*ALAM(2)+AMU(2))
      SU=ABS(CHECK-P(1,I,2))
      IF(SU.GT.SUM) SUM=SU
    2 CONTINUE
   52 I=N+1
      CHECK=P(1,I,2)
      P(1,I,2)=(P(2,I,1)*AMU(1)+P(1,N,2)*ALAM(2))/(AM*ALAM(1)+AMU(2))
```

```
      SU=ABS(CHECK-P(1,I,2))
      IF(SU.GT.SUM) SUM=SU
C     ***************************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(I,J,K) WHERE I=1,....M, J=1,....N,
C     AND K=1,2  AND COMPUTE THE DIFFERENCE BETWEEN THE VALUES FOR
C     ITERATION J+1 AND J
C     ***************************************************************************
      PR=.5
      DO 4 I=2,MM
      DO 4 J=2,NN
      DO 4 K=1,2
      CHECK=P(I,J,K)
      AI=I-1
      AJ=J-1
      II=I+1
      III=I-1
      JJ=J+1
      JJJ=J-1
      IF(J.EQ.NN) JJ=1
      IF(I.EQ.MM) II=1
      P(I,J,K)=(P(II,J,1)*AMU(1)*PR+P(I,JJ,2)*AMU(2)*PR+P(III,J,K)*(AM-
     1AI+1.)*ALAM(1)+P(I,JJJ,K)*(AN-AJ+1.)*ALAM(2))/((AM-AI)*ALAM(1)+
     2(AN-AJ)*ALAM(2)+AMU(K))
      SU=ABS(CHECK-P(I,J,K))
      IF(SU.GT.SUM) SUM=SU
    4 CONTINUE
C     ***************************************************************************
C     CHECK TO SEE IF SHUT-OFF CRITERION IS MET
C     ***************************************************************************
      IT=IT+1
      IF(IT.GT.50) GO TO 50
      IF(SUM.LE..0005) GO TO 50
```

```
      GO TO 51
   50 CONTINUE
      P(1,1,2)=0.
C     ***************************************************************************
C     NORMALIZE THE STATE PROBABILITIES
C     ***************************************************************************
      ADD=0.
      DO 7 I=1,MM
      DO 7 J=1,NN
      DO 7 K=1,2
    7 ADD=P(I,J,K)+ADD
      DO 9 I=1,MM
      DO 9 J=1,NN
      DO 9 K=1,2
    9 P(I,J,K)=P(I,J,K)/ADD
C     ***************************************************************************
C     CALCULATE THE EXPECTED SYSTEM AND QUEUE LENGTHS FOR EACH POPULATION
C     ***************************************************************************
      DO 46 I=1,2
      DO 46 J=1,2
   46 MEAN(I,J)=0.
      DO 47 I=2,MM
      DO 47 J=1,NN
      DO 47 K=1,2
      AI=I-1
      AII=I-2
      IF(K.EQ.2) AII=I-1
      MEAN(1,1)=MEAN(1,1)+AI*P(I,J,K)
   47 MEAN(1,2)=MEAN(1,2)+AII*P(I,J,K)
      DO 48 I=1,MM
      DO 48 J=2,NN
      DO 48 K=1,2
```

```
      AJ=J-1
      AJJ=J-2
      IF(K.EQ.1) AJJ=J-1
      MEAN(2,1)=MEAN(2,1)+AJ*P(I,J,K)
   48 MEAN(2,2)=MEAN(2,2)+AJJ*P(I,J,K)
      GO TO 850
C  ***************************************************************************
C  CALCULATE STATE PROBABILITIES FOR AN ASSIGNMENT WITH ONLY ONE POPULATION
C  ***************************************************************************
   76 AC=N
      L1=N+1
      LL=2
      MEAN(1,1)=0.
      MEAN(1,2)=0.
      GO TO 78
   77 AC=M
      L1=M+1
      LL=1
      MEAN(2,1)=0.
      MEAN(2,2)=0.
   78 P(1,1,1)=1.
      DO 79 J=2,L1
      AD=J-1
      JJ=J-1
   79 P(1,1,1)=P(1,1,1)+EXP(ALGAMA(AC+1.)-ALGAMA(AC-AD+1.))*(ALAM(LL)/
     1AMU(LL))**JJ
      P(1,1,1)=1./P(1,1,1)
      DO 68 J=2,L1
      AD=J-1
      JJ=J-1
   68 P(J,1,1)=EXP(ALGAMA(AC+1.)-ALGAMA(AC-AD+1.))*(ALAM(LL)/AMU(LL))**
     1JJ*P(1,1,1)
```

```
C  ***********************************************************************
C  CALCULATE THE EXPECTED SYSTEM AND QUEUE LENGTHS FOR SYSTEM WITH
C  ONE POPULATION
C  ***********************************************************************
      MEAN(LL,1)=0.
      MEAN(LL,2)=0.
      DO 69 J=2,L1
      AJ=J-1
      AJJ=J-2
      MEAN(LL,1)=MEAN(LL,1)+AJ*P(J,1,1)
   69 MEAN(LL,2)=MEAN(LL,2)+AJJ*P(J,1,1)
C  ***********************************************************************
C  EVALUATE THE OBJECTIVE FUNCTION
C  ***********************************************************************
  850 YY=YY+CSERV(1)*(MEAN(1,1)-MEAN(1,2))+CQUE(1)*MEAN(1,2)
     1+CSERV(2)*(MEAN(2,1)-MEAN(2,2))+CQUE(2)*MEAN(2,2)+CMAN(LM)
   86 CONTINUE
   85 RETURN
  400 FORMAT(10F5.0)
  401 FORMAT(2I5)
      END
```

APPENDIX II

Appendix II concerns the limitation and the use of the pattern search procedure which has been programmed to solve the problem under study. As with the dynamic programming procedure the limitations result from two constraints. The maximum number of servers to be allocated machines is twenty and the maximum population sizes are twenty-four for each population. These limitations may be overcome by increasing the array sizes. The user should specify a region size of 50 K when running on the Fortran G compiler for an IBM 370/158 system.

The following list explains the data cards needed to use the program.

CARD	FORMAT	DESCRIPTION
1	I5	Number of problems to be solved.
2	I5	Print control; 0 if only improved evaluations are printed, 1 if all evaluations are printed.
3	2I5	Number of variables, 2(c-1); and the maximum number of iterations desired in the search.
4	I5	Number of servers (c)
5	2I5	Number of customers in each population (M,N).
6	2F5.0	Arrival Rates of populations 1 and 2 (λ_1,λ_2)

CARD	FORMAT	DESCRIPTION
7 thru 6+c	2F5.0	Service rates of each server for each population (μ_{1j},μ_{2j})
7+c	cF5.0	Fixed costs of each server (C_{31}, C_{32}, ... C_{3c})
8+c	4F5.0	Costs of servicing each population and costs of waiting for each population (C_{21}, C_{22}, C_{11}, C_{12}).
9+c thru 7+2c	2F5.0	Initial assignment of machines of type 1 and type 2 to the first c-1 servers, (x_{11}, x_{12}, ..., $x_{c-1,1}$, $x_{c-1,2}$)

Repeat steps 3 thru 7+2c for multiple problems, where c is the number of servers for each problem.

To aid in using the problem, a list of the main variables is included.

VARIABLE NAME	USE
TB (I)	Trial base for variable I.
BASE (I)	Base for variable I.
YMAX	Best value of the objective value currently found.
BEST (I)	Best value of variable I currently found.
XMIN (I)	Minimum value that variable I may be assigned.
XMAX (I)	Maximum value that variable I may be assigned.
DELT (I)	Current value of the step size for variable I.
DELTM (I)	Minimum step size for variable I.

VARIABLE NAME	USE
DELTU (I)	Maximum step size for variable I
ALAM (I)	Arrival rate for customer type I
BMU (J,I)	Service rate of server J for customer type I.
NSERV	Number of servers.
MMAX	Total number of customers of type 1.
NMAX	Total number of customers of type 2.
NPROB	Number of problems to be solved.
CMAN (J)	Fixed cost of server J, C_{3j}.
CSERV (I)	Cost of servicing customer I, C_{2i}.
CQUE (I)	Cost of waiting for customer I, C_{1i}.
P (I,J,K)	Probability of I-1 customers of type 1 and J-1 customer of type 2 in the system and customer K is presently in service.
MEAN (I,1)	Expected number of type I customers in the system.
MEAN (I,2)	Expected number of type I customers in the queue.

```
      DIMENSION BEST(45)
      DIMENSION BASE(45),XMIN(45),XMAX(45),DELT(45),RF(45),TB(45)
      DIMENSION DELTM(45),DELTU(45),PB(45)
      COMMON/BL/MMAX,NMAX
      COMMON/BLCKL/RF
C     ***************************************************************
C     READ THE NUMBER OF PROBLEMS TO BE SOLVED
C     ***************************************************************
      READ(5,9090) NPROB
      WRITE(6,9C90)NPROB
C     ***************************************************************
C     READ PRINT CONTROL (IF = 1, WILL PRINT EACH FUNCTIONAL
C     EVALUATION;  IF = 0, WILL PRINT ONLY IMPROVED EVALUATIONS
C     ***************************************************************
      READ(5,9090) IPRINT
      WRITE(6,9090)IPRINT
      DO 200 IJ=1,NPROB
C     ***************************************************************
C     READ THE NUMBER OF VARIABLES IN THE SEARCH AND THE MAXIMUM NUMBER
C     OF ITERATIONS
C     ***************************************************************
      READ(5,9000) MX,NS
      WRITE(6,9000)MX,NS
      DO 401 I=1,MX,2
      XMIN(I)=0.
      XMIN(I+1)=0.
      DELT(I)=2.
      DELT(I+1)=2.
      DELTM(I)=1.
      DELTM(I+1)=1.
      DELTU(I)=4.
      DELTU(I+1)=4.
```

```
  401 CONTINUE
      NITER=0
      IND=0
      CALL SUBSYS(IND,YY)
      IND=1
C  ***************************************************************************
C  READ THE STARTING VALUES, THE MINIUMUM AND MAXIMUM VALUES OF THE
C  VARIABLES, THE STARTING STEP SIZE AND THE MINIMUM AND MAXIMUM STEP SIZES
C  ***************************************************************************
      DO 402 I=1,MX,2
      READ(5,9010) BASE(I),BASE(I+1)
      XMAX(I)=MMAX
      XMAX(I+1)=NMAX
      WRITE(6,9010)BASE(I),XMIN(I),XMAX(I),DELT(I),DELTM(I),DELTU(I)
  402 CONTINUE
      DO 30 I=1,MX
      TB(I)=BASE(I)
      RF(I)=BASE(I)
      PB(I)=BASE(I)
   30 CONTINUE
  310 IC=1
      YMAX=-10.**20
      JJ=1
      II = 1
   50 II=1
C  ***************************************************************************
C  EVALUATE THE FUNCTION AT THE TRIAL BASE
C  ***************************************************************************
      DO 60 I=1,MX
      RF(I) = TB(I)
   60 CONTINUE
      CALL SUBSYS(IND,YY)
```

```
      YY=-YY
      IF(IPRINT.LE.0) GO TO 230
      WRITE(6,9000) IC
      WRITE(6,9020) YY,(RF(L2),L2=1,MX)
  230 IF(YY.LE.YMAX) GO TO 70
      IF(IPRINT.GT.0) GO TO 240
      WRITE(6,9000) IC
      WRITE(6,9020) YY,(RF(L2),L2=1,MX)
  240 NITER=0
      DO 280 I=1,MX
  280 BEST(I)=RF(I)
      YMAX=YY
      IF(JJ.NE.1) II=2
C  ***************************************************************
C  PERTURB THE VARIABLE IN THE POSITIVE DIRECTION TO DETERMINE THE NEW
C  TRIAL BASE
C  ***************************************************************
   70 DO 90 I=1,MX
      RF(I)=TB(I)+DELT(I)
   75 CONTINUE
      IF(RF(I).GT.XMAX(I)) RF(I)=XMAX(I)
      CALL SUBSYS(IND,YY)
      YY=-YY
      IF(IPRINT .LE.0) GO TO 250
      WRITE(6,9000) IC
      WRITE(6,9020) YY,(RF(L2),L2=1,MX)
  250 IF(YY.GT.YMAX) GO TO 80
C  ***************************************************************
C  PERTURB THE VARIABLE IN THE NEGATIVE DIRECTION TO DETERMINE
C  THE NEW TRIAL BASE
C  ***************************************************************
      RF(I)=TB(I)-DELT(I)
```

```
  255 CONTINUE
      IF(RF(I).LT.XMIN(I)) RF(I)=XMIN(I)
      CALL SUBSYS(IND,YY)
      YY=-YY
      IF(IPRINT.LE.0) GO TO 260
      WRITE(6,9000) IC
      WRITE(6,9020) YY,(RF(L2),L2=1,MX)
  260 IF(YY.GT.YMAX) GO TO 80
      RF(I)=TB(I)
      GO TO 90
   80 YMAX=YY
      DO 320 K=1,MX
  320 BEST(K)=RF(K)
      IF(IPRINT.GT.0) GO TO 270
      WRITE(6,9000) IC
      WRITE(6,9020) YY,(RF(L2),L2=1,MX)
   99 FORMAT(4F14.2)
  270 NITER=0
      II=2
   90 CONTINUE
      GO TO (120,100),II
  100 JJ=2
C  ***************************************************************************
C  DETERMINE A NEW BASE AND CONTINUE SEARCH
C  ***************************************************************************
      DO 110 I=1,MX
      PB(I)=BASE(I)
      BASE(I)=RF(I)
      TB(I)=2.*BASE(I)-PB(I)
      IF(TB(I).LT.XMIN(I)) TB(I)=XMIN(I)
      IF(TB(I).GT.XMAX(I)) TB(I)=XMAX(I)
  110 CONTINUE
```

```
      GO TO 50
C  ***********************************************************************
C  CHECK TO SEE IF MAXIMUM NUMBER OF ITERATIONS HAS BEEN EXCEEDED
C  ***********************************************************************
  120 IF(IC.GT.NS) GO TO 180
      IC=IC+1
C  ***********************************************************************
C  REDUCE STEP SIZE BY ONE-HALF AND CONTINUE SERACH
C  ***********************************************************************
      DO 170 I=1,MX
      DELT(I)=DELT(I)/2.
      IF(DELT(I).LT.DELTM(I)) GO TO 210
      TB(I)=BASE(I)
      PB(I)=BASE(I)
      RF(I)=BASE(I)
  170 CONTINUE
      GO TO 70
  210 NITER=NITER+1
      IF(NITER.GT.1) GO TO 180
C  ***********************************************************************
C  SET STEP SIZE TO MAXIMUM AND CONTINUE SEARCH
C  ***********************************************************************
      DO 220 I=1,MX
      DELT(I)=DELTU(I)
      BASE(I)=TB(I)
  220 RF(I)=BASE(I)
      GO TO 50
  180 WRITE(6,9030)
      IC=IC-1
C  ***********************************************************************
C  PRINT OBJECTIVE FUNCTION VALUE AND THE OPTIMAL ASSIGNMENTS
C  ***********************************************************************
```

```
      WRITE(6,9080) YMAX
      CHECK1=0.
      CHECK2=0.
      J=0
      DO 600 I=1,MX,2
      J=J+1
      CHECK1=CHECK1+BEST(I)
      CHECK2=CHECK2+BEST(I+1)
  600 WRITE(6,9300) J,BEST(I),BEST(I+1)
      CHECK1=MMAX-CHECK1
      CHECK2=NMAX-CHECK2
      J=J+1
      WRITE(6,9300) J,CJ
      WRITE(6,9300) J,CHECK1,CHECK2
      WRITE(6,9100) IC
      DO 300 I=1,MX
  300 RF(I)=BEST(I)
  200 CONTINUE
      STOP
 9300 FORMAT(' ' ,'ASSIGN TO SERVER ',1X,I2,1X,F4.0,' OF TYPE 1 AND  '
     1,1X,F4.0,'  OF TYPE 2')
 9000 FORMAT(2I5)
 9010 FORMAT(2F5.0)
 9020 FORMAT(1X,7(3X,F14.4))
 9030 FORMAT(1H0,'FINAL SOLUTION',//)
 9080 FORMAT(1X,'OBJECTIVE FUNCTION = ',E14.7,/,1X,'DECISION VARIABLES '
     1,//)
 9090 FORMAT(I5)
 9100 FORMAT(1X,'ITERATIONS = ',I5)
      END
```

```
      SUBROUTINE SUBSYS(IND,YY)
      REAL*4 MEAN(2,2)
      COMMON/BL/MMAX,NMAX
      COMMON/ABC/PROB
      COMMON/BLOKL/RF
      DIMENSION CMAN(22),CSERV(2),CQUE(2),BMU(22,2),RF(45)
      DIMENSION P(25,25,2)
      DIMENSION ALAM(2),AMU(2)
      IF(IND.GT.0) GO TO 65
C     ****************************************************************
C     READ POPULATION SIZES,NUMBER OF SERVERS,ARRIVAL RATES, SERVICE RATES
C     ****************************************************************
      READ 401, NSERV
      WRITE(6,401) NSERV
      READ 401,MMAX,NMAX
       WRITE(6,401) MMAX,NMAX
      READ 400,ALAM(1),ALAM(2)
       WRITE(6,400) ALAM(1),ALAM(2)
      DO 66 I=1,NSERV
      READ 400,BMU(I,1),BMU(I,2)
   66 WRITE(6,400) BMU(I,1),BMU(I,2)
      READ 400,(CMAN(I),I=1,NSERV)
      WRITE(6,400) (CMAN(I),I=1,NSERV)
      READ 400,CSERV(1),CSERV(2),CQUE(1),CQUE(2)
      WRITE(6,400) CSERV(1),CSERV(2),CQUE(1),CQUE(2)
      GO TO 85
   65 MMM=0
      NNN=0
      YY=0.
      CH2=0.
      CH1=0.
      NSERV1=NSERV-1
```

```
      DO 55 I=1,NSERV1
      I1=2*I-1
      I2=2*I
      CH1=CH1+RF(I1)
   55 CH2=CH2+RF(I2)
      IF(CH1.GT.MMAX) GO TO 56
      IF(CH2.GT.NMAX) GO TO 56
      DO 86 LM=1,NSERV
      LN=LM+NSERV
C     ***************************************************************
C     SET SERVICE RATE FOR THE CORRECT SERVER
C     ***************************************************************
      AMU(1)=BMU(LM,1)
      AMU(2)=BMU(LM,2)
      IF(LM.EQ.NSERV) GO TO 87
      LK=2*LM-1
      LN=2*LM
      M=RF(LK)
      MMM=MMM+M
      N=RF(LN)
      NNN=NNN+N
      GO TO 88
   87 M=MMAX-MMM
      N=NMAX-NNN
      IF(M.LE.0) M=0
      IF(N.LE.0) N=0
C     ***************************************************************
C     CHECK TO SEE IF BOTH POPULATION SIZES EQUAL 0 OR IF ONE POPULATION
C     SIZE EQUALS 0
C     ***************************************************************
   88 IF(M.EQ.0.AND.N.EQ.0) GO TO 86
      IF(M.EQ.0) GO TO 76
```

```
      IF(N.EQ.0) GO TO 77
      AN=N
      AM=M
      IT=0
      NN=N+1
      MM=M+1
C     ****************************************************************************
C     SET INITIAL PROBABILITIES FOR EACH STATE
C     ****************************************************************************
      DO 12 I=1,MM
      DO 12 J=1,NN
      DO 12 K=1,2
   12 P(I,J,K)=1./(AM*AN*2.)
      DO 10 I=1,MM
   10 P(I,1,2)=0.
      DO 11 I=2,NN
   11 P(1,I,1)=0.
   51 CONTINUE
C     ****************************************************************************
C     CALCULATE STATE PROBABILITY FOR P(0,0) AND CALCULATE THE DIFFERENCE
C     BETWEEN VALUE FOR ITERATION J+1 AND J
C     ****************************************************************************
      CHECK=P(1,1,1)
      P(1,1,1)=(P(2,1,1)*AMU(1)+P(1,2,2)*AMU(2))/(AM*ALAM(1)+AN*ALAM(2))
      SUM=ABS(CHECK-P(1,1,1))
      P(1,1,2)=P(1,1,1)
C     ****************************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(I,0,1) WHERE I=1,.....M AND COMPUTE
C     THE DIFFERENCE BETWEEN VALUE FOR ITERATION J+1 AND J
C     ****************************************************************************
   54 IF(M.EQ.1) GO TO 53
      DO 1 I=2,M
```

```
      AI=I-1
      II=I+1
      III=I-1
      CHECK=P(I,1,1)
      P(I,1,1)=(P(II,1,1)*AMU(1)+P(I,2,2)*AMU(2)+P(III,1,1)*ALAM(1)*
     1(AM-AI+1.))/((AM-AI)*ALAM(1)+AN*ALAM(2)+AMU(1))
      SU=ABS(CHECK-P(I,1,1))
      IF(SU.GT.SUM) SUM=SU
    1 CONTINUE
   53 I=M+1
      CHECK=P(I,1,1)
      P(I,1,1)=(P(I,2,2)*AMU(2)+P(M,1,1)*ALAM(1))/(AN*ALAM(2)+AMU(1))
      SU=ABS(CHECK-P(I,1,1))
      IF(SU.GT.SUM) SUM=SU
C     ************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(0,I,2) WHERE I=1,.....N AND COMPUTE
C     THE DIFFERENCE BETWEEN THE VALUE FOR ITERATION J+1 AND J
C     ************************************************************
      IF(N.EQ.1) GO TO 52
      DO 2 I=2,N
      AI=I-1
      II=I+1
      III=I-1
      CHECK=P(1,I,2)
      P(1,I,2)=(P(2,I,1)*AMU(1)+P(1,II,2)*AMU(2)+P(1,III,2)*ALAM(2)*
     1(AN-AI+1.))/(AM*ALAM(1)+(AN-AI)*ALAM(2)+AMU(2))
      SU=ABS(CHECK-P(1,I,2))
      IF(SU.GT.SUM) SUM=SU
    2 CONTINUE
   52 I=N+1
      CHECK=P(1,I,2)
      P(1,I,2)=(P(2,I,1)*AMU(1)+P(1,N,2)*ALAM(2))/(AM*ALAM(1)+AMU(2))
```

```
      SU=ABS(CHECK-P(1,I,2))
      IF(SU.GT.SUM) SUM=SU
C     ***************************************************************************
C     CALCULATE STATE PROBABILITIES FOR P(I,J,K) WHERE I=1,....M, J=1,....N,
C     AND K=1,2  AND COMPUTE THE DIFFERENCE BETWEEN THE VALUES FOR
C     ITERATION J+1 AND J
C     ***************************************************
      PR=.5
      DO 4 I=2,MM
      DO 4 J=2,NN
      DO 4 K=1,2
      CHECK=P(I,J,K)
      AI=I-1
      AJ=J-1
      II=I+1
      III=I-1
      JJ=J+1
      JJJ=J-1
      IF(J.EQ.NN) JJ=1
      IF(I.EQ.MM) II=1
      P(I,J,K)=(P(II,J,1)*AMU(1)*PR+P(I,JJ,2)*AMU(2)*PR+P(III,J,K)*(AM-
     1AI+1.)*ALAM(1)+P(I,JJJ,K)*(AN-AJ+1.)*ALAM(2))/((AM-AI)*ALAM(1)+
     2(AN-AJ)*ALAM(2)+AMU(K))
      SU=ABS(CHECK-P(I,J,K))
      IF(SU.GT.SUM) SUM=SU
    4 CONTINUE
      IT=IT+1
C     ***************************************************************************
C     CHECK TO SEE IF SHUT-OFF CRITERION IS MET
C     ***************************************************************************
      IF(IT.GT.50) GO TO 50
      IF(SUM.LE..0005) GO TO 50
```

```
      GO TO 51
   50 CONTINUE
      P(1,1,2)=0.
C  ***************************************************************************
C  NORMALIZE THE STATE PROBABILITIES
C  ***************************************************************************
      ADD=0.
      DO 7 I=1,MM
      DO 7 J=1,NN
      DO 7 K=1,2
    7 ADD=P(I,J,K)+ADD
      DO 9 I=1,MM
      DO 9 J=1,NN
      DO 9 K=1,2
    9 P(I,J,K)=P(I,J,K)/ADD
C  ***************************************************************************
C  CALCULATE THE EXPECTED SYSTEM AND QUEUE LENGTHS FOR EACH POPULATION
C  ***************************************************************************
      DO 46 I=1,2
      DO 46 J=1,2
   46 MEAN(I,J)=0.
      DO 47 I=2,MM
      DO 47 J=1,NN
      DO 47 K=1,2
      AI=I-1
      AII=I-2
      IF(K.EQ.2) AII=I-1
      MEAN(1,1)=MEAN(1,1)+AI*P(I,J,K)
   47 MEAN(1,2)=MEAN(1,2)+AII*P(I,J,K)
      DO 48 I=1,MM
      DO 48 J=2,NN
      DO 48 K=1,2
```

```
      AJ=J-1
      AJJ=J-2
      IF(K.EQ.1) AJJ=J-1
      MEAN(2,1)=MEAN(2,1)+AJ*P(I,J,K)
   48 MEAN(2,2)=MEAN(2,2)+AJJ*P(I,J,K)
      GO TO 850
C     ****************************************************************************
C     CALCULATE STATE PROBABILITIES FOR AN ASSIGNMENT WITH ONLY ONE POPULATION
C     ****************************************************************************
   76 AC=N
      L1=N+1
      LL=2
      MEAN(1,1)=0.
      MEAN(1,2)=0.
      GO TO 78
   77 AC=M
      L1=M+1
      LL=1
      MEAN(2,1)=0.
      MEAN(2,2)=0.
   78 P(1,1,1)=1.
      DO 79 J=2,L1
      AD=J-1
      JJ=J-1
   79 P(1,1,1)=P(1,1,1)+EXP(ALGAMA(AC+1.)-ALGAMA(AC-AD+1.))*(ALAM(LL)/
     1AMU(LL))**JJ
      P(1,1,1)=1./P(1,1,1)
      DO 68 J=2,L1
      AD=J-1
      JJ=J-1
   68 P(J,1,1)=EXP(ALGAMA(AC+1.)-ALGAMA(AC-AD+1.))*(ALAM(LL)/AMU(LL))**
     1JJ*P(1,1,1)
```

```
C  ***********************************************************************
C  CALCULATE THE EXPECTED SYSTEM AND QUEUE LENGTHS FOR SYSTEM WITH
C  ONE POPULATION
C  ***********************************************************************
      MEAN(LL,1)=0.
      MEAN(LL,2)=0.
      DO 69 J=2,L1
      AJ=J-1
      AJJ=J-2
      MEAN(LL,1)=MEAN(LL,1)+AJ*P(J,1,1)
   69 MEAN(LL,2)=MEAN(LL,2)+AJJ*P(J,1,1)
C  ***********************************************************************
C  EVALUATE THE OBJECTIVE FUNCTION
C  ***********************************************************************
  850 YY=YY+CSERV(1)*(MEAN(1,1)-MEAN(1,2))+CQUE(1)*MEAN(1,2)
     1+CSERV(2)*(MEAN(2,1)-MEAN(2,2))+CQUE(2)*MEAN(2,2)+CMAN(LM)
   86 CONTINUE
      GO TO 85
   55 YY=10.**10
   85 RETURN
  400 FORMAT(10F5.0)
  401 FORMAT(2I5)
      END
```

The vita has been removed from the scanned document

THE ALLOCATION OF NON-IDENTICAL MACHINES AMONG NON-IDENTICAL SERVERS

by

Thomas A. Carpenito

(ABSTRACT)

The problem of allocating non-identical machines among non-identical servers is considered under steady state conditions for the case of quasirandom input and exponential service times. Machines are assigned to operators with the objective of minimizing an expected cost model of the queuing system. Different classes of machines have different service rates and, for a given class of machines, the service rates are different among servers. Both dynamic programming and pattern search are used to solve the resulting optimization problem. Computational experience is presented, along with an analysis of the sensitivity of the model to errors in estimating the values of the parameters of the model.