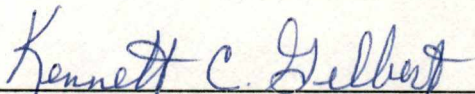
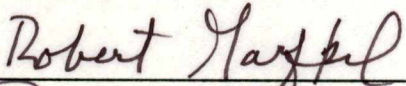
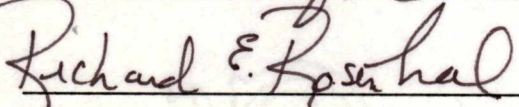


To the Graduate Council:

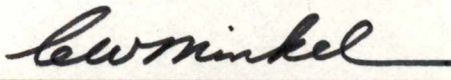
I am submitting herewith a thesis written by Richard S. Butterini entitled "An Implicit Enumeration Algorithm for the Computer System Configuration/Expansion Problem." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Management Science.


Kenneth C. Gilbert, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


The Graduate School

AN IMPLICIT ENUMERATION ALGORITHM FOR THE COMPUTER
SYSTEM CONFIGURATION/EXPANSION PROBLEM

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Richard S. Butturini

December 1983

ACKNOWLEDGMENTS

The author takes this opportunity to express his gratitude to Dr. Kenneth C. Gilbert for his unending assistance, without which this thesis would not have been possible. The author also wishes to express his appreciation to Dr. Robert S. Garfinkel and Dr. Richard E. Rosenthal for their services on the Master's degree committee.

ABSTRACT

This paper introduces the Computer System Configuration/Expansion Problem (CSCEP), a combinatorial problem of selecting the hardware and software components of a computer system to provide the required capacity at a minimum cost. We present a general integer programming formulation of the problem and a method of solution based on the implicit enumeration procedure of Trotter and Shetty [23].

By exploiting the special structure of the CSCEP, we have developed a very efficient algorithm. Typically a hardware or software device will provide only one or two types of capacity (although it may consume many different types of capacity). This fact is the basis for efficient procedures for augmentation, computing objective function bounds and dynamically bounding variables in the CSCEP algorithm.

The algorithm was implemented with data derived from Oak Ridge Associated University's historical capacity requirements. Problems with 33 variables and 19 constraints were solved in core using a 32 kilobyte region on a PDP 11/34 minicomputer.

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
II. LITERATURE REVIEW	3
General Capacity Expansion Models	3
Computer System Capacity Planning	4
Computer System Performance Monitoring	6
Computer System Performance Modelling	6
Optimization Models for Computer Design	7
III. THE COMPUTER SYSTEM CONFIGURATION/EXPANSION MODEL	9
Formulation	9
The Trotter-Shetty Algorithm	11
The CSCEP Algorithm Fathoming Procedures	14
The CSCEP Bound Redefinition Procedure	19
The CSCEP Augmentation Procedures	20
Flexibility of the CSCEP Model	23
IV. IMPLEMENTATION AND COMPUTATIONAL RESULTS	30
V. TOPICS FOR RESEARCH	37
LIST OF REFERENCES	38
APPENDICES	42
APPENDIX A: DESCRIPTIONS OF ORAU PROBLEMS	43
APPENDIX B: ENUMERATION PROGRAM LISTING	46
VITA	54

LIST OF TABLES

TABLE	PAGE
1. An Example of the Trotter-Shetty Algorithm	13
2. An Example of the CSCEP Algorithm	22
3. Devices Used in Hypothetical Example	24
4. Resources Represented by Constraints in CSCEP Example	24
5. Application of CSCEP Algorithm to a Hypothetical Example . .	25
6. Computational Results for ORAU Problems	31
7. Devices Available for Solutions to ORAU Problems	33
8. Requirements Used in ORAU Problems	34

CHAPTER I

INTRODUCTION

Previous research in the development of mathematical models for selecting computer hardware and software components has been very limited. This is in sharp contrast to the extensive research in computer performance modeling, computer performance measurement and computer capacity planning. These areas deal with determining the capacity required to meet the users' needs. However, they do not address the problem of configuring a system to provide this capacity. In discussing capacity planning Chandy [6] points out this problem:

The second step is to configure a system to handle the expected workload and to meet planned service level goals. Ideally, a DP planner would have a software package, the input to which would be the expected workload and service level agreements, and the output of which would be the minimum cost configuration. Such a package does not exist in a useful form (though there are research projects concerned with such tools). Configuring the system continues to be a creative art, not amenable to modification as a simple set of rules.

Here the configuration problem is that of selecting the components of a computer system from a finite set of choices. This is as opposed to configurations problems, e.g. see Trivedi [26], in which the parameters of a hypothetical computer system are treated as continuous decision variables. These models based on queueing networks use nonlinear programming techniques to determine the optimal capabilities of the central processing unit (CPU), memory, input/output (I/O) devices and to derive an optimal scheduling algorithm. The objective of these models is to maximize the throughput subject to a cost constraint. The cost is

generally assumed to be a continuous function of the system parameters. These models can, at a low cost, help a system designer design a system (in terms of the system parameters) best suited to his work load and budget. Once the system parameters are determined, the model developed in this paper can be used to configure a system from available hardware and software components.

CHAPTER II

LITERATURE REVIEW

The Computer System Configuration/Expansion Problem (CSCEP) is a logical extension of several major areas of research. Expansion models have traditionally been applied to production and distribution systems. Computer system capacity planning, which deals with determining the resources needed to meet user requirements, is a second area of research relevant to the CSCEP. Computer performance monitoring and measurement deals with methods used to determine the utilization of the resources of an existing computer system. Computer performance modelling deals with modelling (usually with queueing network or simulation models) the operation of a computer system and predicting its performance. Computer design models are nonlinear programming models used to optimally determine parameters such as CPU speed, I/O device capabilities and scheduling algorithms for a computer system.

General Capacity Expansion Models

Capacity expansion models have traditionally been applied to the expansion of production and distribution systems. These models differ from the model presented here in that they usually deal with meeting a time-phased demand for a single resource. The earliest research dealt with single-facility, finite-horizon models used to plan both the size and timing of expansions to meet growing demand for a single product. Lambrecht and Vander Eecken [18] developed an enumeration technique, and Baker et al. [3] presented a branch-and-bound algorithm for solutions to

this problem. Yaged [33] used a dynamic programming algorithm to select expansion sizes from a finite set of options. Smith [24] used a forward looking enumeration technique to solve a similar problem.

Water resource capacity expansion problems in which the expansion is obtained through the implementation of a finite set of projects have been solved using dynamic programming. Erlenkotter [7] developed a backward dynamic programming problem to find the optimal sequence of projects. Mitten and Tsou [22] and O'Laoghaire and Himmelblau [23] used branch-and-bound algorithms to sequence projects.

Two-facility and multi-facility problems allow the capacity of one facility to meet demand at another at a conversion or transportation cost. Dynamic programming problems like that of Luss [20] are used to produce results for the two-facility model. The multi-facility problem has been used in a variety of applications and solved with a variety of techniques. Friedenfelds and McLaughlin [9] developed a branch-and-bound algorithm using heuristic bounds for use in planning telephone feeder capacity expansions. Jacoby and Loucks [14] used simulation models to test water resource expansion policies.

Computer System Capacity Planning

Computer system capacity planning deals with determining the requirements of various computing resources needed to meet the users needs. The computer performance models and computer performance monitoring techniques discussed below are tools which may be used in a capacity planning study. The requirements determined from such a study would be the input to a Computer System Configuration/Expansion model such as the one developed in this paper.

Many capacity planning methodologies are applications of standard planning techniques to the computer facility environment. Typical of these approaches are those described by Chandy [6] and by Stevens [25]. The latter work lists the tasks in the process as:

1. Identify and measure the resources to be included in capacity planning;
2. Identify, characterize and forecast the workload to be processed by those resources;
3. Identify service levels to be delivered from those resources and project them during the capacity management process;
4. Compute the costs of providing the specific levels of performance desired.

He suggests breaking down the resources into different groups called work centers. Each work center is divided into activities called work units. The utilization of each work center by a work unit is analyzed and the effect of a change in the work unit can be predicted.

Vincent [32] suggests setting standards for an activity's utilization of a resource and comparing them to actual performance. The results are then used to determine the nature of user workload and the effect changes in the workload will have on system performance. Predictive models are used to examine hardware alternatives to meeting the demand generated by the changing workload.

A somewhat different approach is used by Kriebel and Raviv [17]. Economic theory is combined with queueing theory to produce a model which views the organization as a marketplace for computer services. Bottlenecks, utilization measurements, and quality of services are all examined.

Computer System Performance Monitoring

An important part of determining the computer resource requirements is the measurement of the current level of usage of these resources. Hardware monitors such as the one described by Bekkers and Decouty [4] are used to measure the activities of the computer from outside the system. This method offers the advantage of being able to analyze the functioning of the system without affecting its operations as opposed to software monitors which actually gather data on the system on which they run and therefore alter the true events.

Computer System Performance Modelling

Modelling approaches offer an inexpensive (compared to actual experiments) method of predicting the performance of a computer system. Many performance models use closed queueing networks. A closed queueing network is one in which the number of jobs in the system remains constant at n (in a computer system n is the level of multiprogramming). The assumption is that each job upon completion re-enters the system via a "new program path." The rate of flow along this path measures the throughput of the system.

Jackson [12,13] and Gordon and Newell [11] are credited with the introduction of queueing networks. They developed equations which related performance quantities, such as device utilizations and mean device queue lengths, to basic system parameters, such as mean service times and routing frequencies.

Fayolle and Robin [8] studied queueing policies in systems which utilized more than one computer. The jobs waiting to be executed are allowed to move from one queue to another, and an optimal utilization is

developed. Mitra [21] analyzed a system consisting of many terminals sharing a single processor in an interactive environment. A closed queueing network model is used to show how short jobs are favored over long jobs in this type of system. Buzen [5] applied queueing theory to several areas of performance evaluation, and developed efficient algorithms for computing the equilibrium probability distribution. Kameda [16] used a finite-source queueing model to compare scheduling decisions in both multiprogrammed and time-sharing systems.

Simulation models have also been used to predict and analyze the performance of various computer environments. Artis [1] used a trace-driven model for characterizing a system's workload and for determining what benefits would have resulted from a different storage configuration. A simulation model was also used by Liu [19] to determine optimal memory partitioning in multiprogrammed systems. Badel and Shum [2] used simulation to evaluate the theoretical design of a computer system. Gomaa [10] applied both regression and simulation models to evaluate the performance of a computer.

Optimization Models for Computer Design

Most optimization models for computer design represent a computer system as a closed queueing network. The decision variables are generally the speeds of the CPU and each of the I/O devices. Other decision variables may include file assignments and the size and speed of each level of memory. The decision variables are generally considered to be continuous and the cost of a system is estimated to be a nonlinear function of these variables. The objective is to maximize the throughput of the system subject to a cost constraint.

Examples of such models are Kachhal and Arrora [15], which uses modified Hook and Jeeves search procedure, and the following models which use Lagrange Multipliers: Trivedi and Wagner [29], Trivedi and Sigmon [28], and Trivedi, Wagner and Sigmon [30]. Trivedi [26] is a good survey article in this area.

CHAPTER III

THE COMPUTER SYSTEM CONFIGURATION/EXPANSION MODEL

The Computer System Configuration/Expansion Problem (CSCEP) is one of optimally selecting the hardware and software components of a system to meet the users' requirements. The users' requirements are expressed in units such as bytes (e.g., for internal memory and disk space), processing capability per unit time (central processing, input, output and disk drive capabilities) or in number of devices (terminals and printers). To meet these exogenous requirements, one must choose components such as CPUs, internal memory, tape drives, disk drives, terminal ports and system software. These components in turn generate endogenous requirements. For example, the addition of a disk drive requires a controller which creates a load on the CPU. The drive itself consumes floor space, air conditioning and power.

The interactions between components on the system make it impossible to consider each requirement of the system independently of the others. Furthermore, there may be so many feasible combinations of hardware and software components which would meet the users' requirements that it is impractical to enumerate them all in order to find the least costly.

Formulation

We define the Computer System Configuration/Expansion Problem as one of choosing a combination of devices to meet the projected requirements as inexpensively as possible. The term "device" here generally refers to a piece of hardware but could also include an addition or modification

to the system software or any other means of meeting the projected requirements. The model contains a decision variable for each different type of device described by the user. For example, corresponding to three types of available disk drives, there would be three decision variables representing the number of each to be used.

The model contains one constraint for each resource. The net requirement for a resource is its projected exogeneous requirements minus the amount of the resource available in the present system.

While a device may consume several resources, often it supplies only one. The algorithm presented here exploits this fact.

The model is formulated as:

$$\begin{aligned} \min \vec{c}\vec{x} \\ \text{subject to } A\vec{x} &\geq \vec{b} \\ \vec{0} \leq \vec{x} \leq \vec{d}, \vec{x} &\text{ integer} \end{aligned} \quad [3.1]$$

where $\vec{c}$, $\vec{x}$ and $\vec{d}$ are n -vectors, $\vec{b}$ is an m -vector and A is an $m \times n$ matrix. The components of $\vec{c}$, $\vec{x}$, $\vec{d}$ and A are defined as follows:

- c_j is the cost per unit (assumed to be non-negative) of device j ;
- x_j is a decision variable representing the number of units of device j to be used;
- d_j is the maximum number of units of device j which can be used;
- b_i is the net demand for resource i ;
- a_{ij} is the amount of resource i supplied (or negative of amount consumed) by one unit of device j .

The Trotter-Shetty Algorithm

The basic enumerative procedure of our algorithm is that of Trotter and Shetty [31]. We follow their notation and for completeness we summarize their description of the algorithm. At any point in an enumeration of solutions to a problem of the form (3.1), denote the best known solution, the incumbent, as x^* and its objective value as $\bar{z}^*$. At any stage each variable x_j is either fixed at one of its admissible values: $0, 1, \dots, d_j$, or else is free to assume any of these values. Let the ordered set $S = \{j_1, j_2, \dots\}$ contain the indices of those variables which are currently fixed, in the order in which they become fixed. Denote by $X_S = \{x_{j_1}, x_{j_2}, \dots\}$ the partial solution vector which contains the values assumed by the fixed variables. For example, $S = \{3, 1, 5\}$ and $X_S = (2, 4, 6)$ denote that $x_3 = 2$, $x_1 = 4$ and $x_5 = 6$ have been fixed in that order. Let $F = \{k_1, k_2, \dots\}$ denote the set of free variables.

A partial solution X_S is said to be fathomed in either of the following two cases:

1. It can be demonstrated that no feasible completion (assignment of values to the free variables) exists with objective value less than $\bar{z}^*$.
2. The best feasible completion of X_S has been found.

The skeletal steps of the algorithm are as follows:

1. Initialization. Set $\bar{z} = \infty$ and choose an initial partial solution. Go to step 2.
2. Fathoming. If the current partial solution can be fathomed, go to step 3, otherwise go to 6.

3. Updating. Feasible completion with objective value less than $\bar{z}$ can be found for the current partial solution, replace the incumbent with this improved solution. Go to step 4.
4. Termination. If all of the elements of S are not underlined, go to step 5. Otherwise the incumbent solution is optimal or, if $\bar{z} = \infty$, no feasible solution exists.
5. Backtracking. Alter the value of the rightmost nonunderlined element of S to a new admissible value (not considered since this index last entered S) and drop all elements of S to its right. Underline the rightmost element of S if its corresponding variable has now assumed each of its admissible values since its index last entered S . Update X_S and go to step 2.
6. Augmentation. Fix a free variable at one of its admissible values, update S and X_S and go to step 2.

The algorithm is both exhaustive and nonredundant; hence, it converges to an optimal solution in a finite number of steps. The example in Table 1, formulated below, demonstrates the algorithm.

$$\min z = 2x_1 + 2x_2 + 3x_3 \quad [3.2]$$

$$2x_1 - x_2 - 2x_3 \geq 0 \quad [3.3]$$

$$x_2 + 2x_3 \geq 3 \quad [3.4]$$

$$0 \leq x_1, x_2, x_3 \leq 2$$

In the implementation of the algorithm, Trotter and Shetty tested several fathoming methods, augmentation rules and bound redefinition procedures. Our algorithm incorporates some of these procedures. In addition by exploiting the special properties of the CSCEP we have designed a very efficient algorithm.

Table 1
An Example of the Trotter-Shetty Algorithm

Stage	S	X_S	z	Comments
i	(1)	(2)	4	Enumerate further.
ii	(1,2)	(2,2)	8	Enumerate further.
iii	(1,2,3)	(2,2,2)	14	Violates [3.3]. Backtrack.
iv	(1,2,3)	(2,2,1)	11	Incumbent. Backtrack.
v	(1,2,3)	(2,2,0)	8	Violates [3.4]. Backtrack.
vi	(1,2)	(2,1)	6	Enumerate further.
vii	(1,2,3)	(2,1,2)	12	Inferior to current incumbent. Backtrack.
viii	(1,2,3)	(2,1,1)	9	New incumbent. Backtrack.
ix	(1,2,3)	(2,1,0)	6	Violates [3.4]. Backtrack.
x	(1,2)	(2,0)	4	Enumerate further.
xi	(1,2,3)	(2,0,2)	10	Inferior to current incumbent. Backtrack.
xii	(1,2,3)	(2,0,1)	7	Violates [3.4]. Backtrack.
xiii	(1,2,3)	(2,0,0)		Violates [3.4]. Backtrack.
xiv	(1)	(1)	2	Enumerate further.
xv	(1,2)	(1,2)	6	Does not admit to a feasible solution. Backtrack.
xvi	(1,2)	(1,1)	4	Does not admit to a feasible solution. Backtrack.
xvii	(1,2)	(1,0)	2	Enumerate further.
xviii	(1,2,3)	(1,0,2)	8	Violates [3.3]. Backtrack.
xix	(1,2,3)	(1,0,1)	5	Violates [3.4]. Backtrack.
xx	(1,2,3)	(1,0,0)	2	Violates [3.4]. Backtrack.
xxi	(1)	(0)	0	Does not admit to a feasible solution. End of enumeration. Result is $X_1 = 2$, $X_2 = 1$, $X_3 = 1$.

The CSCEP Algorithm Fathoming Procedures

At any point in the enumeration let $b_i^S = b_i - \sum_{j \in S} a_{ij}x_j$ and $z^S = \sum_{j \in S} c_j x_j$. The problem of interest is:

$$\min z^S + \sum_{j \in F} c_j x_j \quad [3.5]$$

subject to:

$$\sum_{j \in F} a_{ij} x_j \geq b_i^S \quad i = 1, 2, \dots, m \quad [3.6]$$

$$0 \leq x_j \leq d_j \text{ and } x_j \text{ integer, } j \in F.$$

Fathoming procedure 1 (best feasible completion found). If $b_i^S \leq 0$ for all i , then the best feasible completion of X_S can be obtained by assigning a value of zero to all free variables.

Fathoming procedure 2 (no feasible completion exists). Suppose for some i , $1 \leq i \leq m$, β_i^S is a lower bound on the additional amount of resource i required in any feasible completion of X_S , i.e. β_i^S is a number satisfying the following condition:

For any feasible completion of X_S it must be true that

$$\sum_{j \in F} \max(0, a_{ij}d_j) \geq \beta_i^S. \quad [3.7]$$

If $\sum_{j \in F} \max(0, a_{ij}d_j) < \beta_i^S$ for some i then no feasible completion of X_S exists and X_S can be fathomed. In the CSCEP these bounds are computed iteratively. Theorem 1 is the basis for this procedure.

Consider any set $I \subseteq \{1, 2, \dots, m\}$ having the following property:

$$\begin{aligned} &\text{For any } i \text{ and } j, 1 \leq i \leq m, j \in F, \\ &a_{ij} \geq 0 \text{ implies } \hat{a}_{ij} \leq 0 \text{ for } \hat{i} \in I - \{i\}. \end{aligned}$$

Each device j , $j \in F$, supplies only one of the resources indexed by I .

Since in the CSCEP a device typically supplies only one resource, a set I can be defined which indexes most, if not all, of the resources.

Theorem 1: For $i = 1, 2, \dots, m$, let β_i^S be a set of bounds satisfying condition [3.7] and let I be a set indexing resources described above.

Then for $i = 1, 2, \dots, m$, $\hat{\beta}_i^S$ is also a bound satisfying [3.7] where

$$\hat{\beta}_i^S = b_i^S + \sum_{k \in I - \{i\}} r_{ik}^S \beta_k^S \quad [3.8]$$

and

$$r_{ik}^S = \begin{cases} \min_{j \in F} (-a_{ij}/a_{kj}) & \text{if } \beta_k^S > 0 \\ 0 & \text{otherwise} \end{cases} \quad [3.9]$$

for $i = 1, 2, \dots, m$ and $k \in I - \{i\}$.

Proof: For each $\beta_k^S > 0$ assume there exists an $a_{kj} > 0$, $j \in F$ (otherwise X_S can be fathomed). Thus r_{ik}^S is properly defined. Each positive value of r_{ik}^S represents a lower bound on the consumption of resource i per unit of resource k provided, using free variables. Therefore, $r_{ik}^S \beta_k^S$ is a lower bound on the amount of resource i which will be consumed by devices supplying resource k . I is defined such that the sets of devices supplying each of the resources indexed by I are mutually exclusive and hence their consumption of any resource is additive. Thus the summation is a lower bound on the consumption of resource i by devices corresponding to free variables. This summation is added to b_i^S , which is the net requirement of resource i from devices corresponding to fixed variables. QED.

Theorem 1 suggests the following iterative method for computing a nondecreasing sequence (in ℓ) of bounds vectors $\vec{\beta}^S(\ell) = (\beta_1^S(\ell), \beta_2^S(\ell), \dots, \beta_m^S(\ell))^t$ where each $\beta_i^S(\ell)$ satisfies [3.7]. Let

$$\beta_i^S(0) = \max(0, b_i^S) \quad [3.10]$$

for $i = 1, 2, \dots, m$.

For $\ell \geq 0$ let

$$\beta_i^S(\ell+1) = \max \left(b_i + \sum_{k \in I - \{i\}} r_{ik}^S(\ell) \cdot \beta_k^S(\ell), 0 \right) \quad [3.11]$$

for $i = 1, 2, \dots, m$

where

$$r_{ik}(\ell) = \begin{cases} \min_{\substack{j \in F \\ a_{kj} > 0}} (-a_{ij}/a_{kj}) & \text{if } \beta_k^S(\ell) > 0 \\ 0 & \text{otherwise} \end{cases} \quad [3.12]$$

for $i = 1, 2, \dots, m$ and $k \in I - \{i\}$.

In [3.11] one must save all the components of $\vec{\beta}^S(\ell)$ while computing $\vec{\beta}^S(\ell+1)$. It is intuitively appealing to replace components of $\vec{\beta}^S(\ell)$ with those corresponding components of $\vec{\beta}^S(\ell+1)$ which have already been computed. Assuming the bounds are computed in order $1, 2, \dots, m$ we obtain:

$$\beta_i^S(\ell+1) = \max \begin{cases} b_i + \sum_{\substack{k < i \\ k \in I}} r_{ik}^S(\ell+1) \cdot \beta_k^S(\ell+1) + \sum_{\substack{k > i \\ k \in I}} r_{ik}^S(\ell) \cdot \beta_k^S(\ell) \\ 0 \end{cases} \quad [3.13]$$

for $i = 1, 2, \dots, m$.

Theorem 2: With $\vec{\beta}^S(0)$ defined by [3.10], the iterative method [3.11] gives a nondecreasing vector sequence $\beta^S(\ell)$. If X_S has a feasible completion the sequence is convergent.

Proof: The proof is by induction. Note that for any (i, k, ℓ) $r_{ik}^S(\ell)$ is, by definition, non-negative. From expression [3.11] it can be seen that $\beta_i^S(1) \geq \max(0, b_i^S) = \beta_i^S(0)$ for $i = 1, 2, \dots, m$. Suppose for some $\ell \geq 1$, $\beta_i^S(\ell) \geq \beta_i^S(\ell - 1)$ for $i = 1, 2, \dots, m$. Together with [3.12] this implies that $r_{ik}^S(\ell) \geq r_{ik}^S(\ell - 1)$ for $i = 1, 2, \dots, m$ and $k \in I - \{i\}$. Hence

$$b_i + \sum_{k \in I - \{i\}} r_{ik}(\ell) \cdot \beta_k^S(\ell) \geq b_i + \sum_{k \in I - \{i\}} r_{ik}(\ell - 1) \cdot \beta_k^S(\ell - 1). \quad [3.14]$$

Thus $\beta_i^S(\ell + 1) \geq \beta_i^S(\ell)$ for $i = 1, 2, \dots, m$ and by induction $\vec{\beta}^S(\ell)$ is nondecreasing. If X_S has a feasible completion then $\vec{\beta}^S(\ell)$ must also be bounded and therefore convergent.

The proof of Theorem 2 for iterative method [3.13] is, with the obvious changes, the same. By a similar argument it can also be shown that at any iteration ℓ , $\ell \geq 0$, the bound vector obtained using [3.13] is greater than or equal to that obtained using [3.11]. The CSCEP algorithm uses [3.13]. The process stops (1) when

$$\frac{\|\beta_i^S(\ell) - \beta_i^S(\ell - 1)\|}{\|\beta_i^S(\ell - 1)\|} \leq \epsilon \quad [3.15]$$

where ϵ is a specified percentage change and $\|\cdot\|$ denotes the vector norm, (2) when

$$\sum_{j \in F} \max(0, a_{ij}d_j) < \beta_i^S(\ell)$$

and X_S is therefore fathomed or (3) after a specified number of iterations have been performed. In the problems we solved the best CPU times were obtained by simply stopping after the first iteration.

It should also be observed that if $\vec{\beta}^S(\lambda)$ is convergent and if for some value of λ , $\vec{\beta}^S(\lambda)$ becomes strictly positive the limit of $\vec{\beta}^S(\lambda)$ is a solution to the system of equations

$$P \vec{x} = \vec{b} \quad [3.16]$$

where $P = [p_{ik}]$ is an $m \times m$ matrix with

$$p_{ik} = \begin{cases} 1 & \text{if } i = k \\ -r_{ik}(\lambda) & \text{if } i \neq k, k \in I, \\ 0 & \text{otherwise.} \end{cases} \quad [3.17]$$

In this case iterative method [3.11] becomes the Point Jacobi method for solving system [3.16] and iterative method [3.13] becomes the Point Gauss-Seidel method.

Fathoming procedure 3 (no feasible completion exists with objective value less than $\bar{z}^*$). If $\underline{z}^S$ is a lower bound on the objective value of any feasible completion of X_S , then X_S can be fathomed if $\underline{z}^S \geq \bar{z}^*$.

The iteratively computed bounds β_i^S , $i = 1, 2, \dots, m$, satisfying condition [3.7], can be used to compute a lower bound as follows:

$$\underline{z}^S = z^S + \sum_{k \in I} (r_k \beta_k^S) \quad [3.18]$$

where

$$r_k = \begin{cases} \min_{j \in F} a_{kj} > 0, & c_j/a_{kj} \text{ if } \beta_k^S > 0 \\ 0 & \text{otherwise} \end{cases} \quad [3.19]$$

Theorem 3: $\underline{z}^S$ is a lower bound on the objective value of any feasible completion of X_S .

Proof: For each $\beta_k^S > 0$, there exists an $a_{kj} > 0$, $j \in F$ (otherwise X_S can be fathomed. Observe each term of the summation in [3.16] is a lower bound on the cost of satisfying some constraint $k \in I$ using free variables, and I is defined so that the terms are additive. QED.

The CSCEP Bound Redefinition Procedure

It is possible to accelerate the implicit enumeration by computing a temporary upper bound d_j^i on x_j , $j \in F$. For example, a free variable should never exceed a value which implies nonoptimality or infeasibility. Due to the special structure of the CSCEP, it is possible to develop very good upper bounds on variables. Let $R_j = \{i \mid a_{ij} \geq 0\}$, i.e., R_j indexes the resources supplied by devices j (typically R_j will contain one element). Then in an optimal completion of X_S any free variable x_j must satisfy:

$$x_j \leq \max_{i \in R_j} \langle [b_i^S + \sum_{k \in F} \max(-a_{ik}, 0) \cdot d_k, 0] / a_{ij} \rangle \quad [3.20]$$

where $\langle \rangle$ denotes the smallest integer which is greater than or equal to x_j .

The numerators inside the brackets of the terms above represent a maximum on the amount of resource i that could be required in a feasible completion of X_S . Therefore, by dividing each of these terms by a_{ij} and rounding up we can determine a maximum on the number of units of device j which would be needed to meet this requirement for resource i . The maximum over all resources supplied by device j is an upper bound on x_j . We define d_j^i as the minimum of d_j and the upper bound in [3.20]. In the case of device j supplying a single resource i for which $a_{ik} \geq 0$ for all

$k \in F$, i.e. no free variable consumes this resource, the bound in [3.20] is simply the number of units of the device required to satisfy demand $b_i S$:

$$x_j \leq \langle b_i S / a_{ij} \rangle \quad [3.21]$$

In the implementation of the algorithm the upper bounds for those variables having this property are computed first using [3.21] so that these improved bounds can be used in [3.20] for computing upper bounds on the other variables. Note that it is possible to develop an iterative procedure based on [3.20] in which the set of upper bounds computed at one iteration are used to compute possibly better bounds at the next iteration. However, the CSCEP augmentation procedure always attempts to do explicit enumeration on a variable to which [3.21] applies. Such an iterative procedure does not improve the bounds on these variables and hence does not significantly reduce the enumeration.

The CSCEP Augmentation Procedures

Augmentation is the step of the algorithm in which a free variable is fixed. Each augmentation step in the CSCEP algorithm attempts to ensure that one additional constraint will be satisfied as cheaply as possible using a single device. The first feasible solution obtained using this augmentation procedure is essentially a "greedy" solution and usually provides a very tight upper bound on z^* , the optimal objective value.

The augmentation rules are as follows:

1. Choose a row i for which $b_i S > 0$ and $a_{ij} \geq 0$ for all $j \in F$ (none of the free variables consume resource i). If no such row

exists, choose a row i which has $b_i^S > 0$ and has the minimum number of negative a_{ij} . Go to step 2.

2. Let $J = \{j \mid a_{ij} > 0, j \in F, d_j \cdot a_{ij} \geq b_i^S\}$. (J is a set of devices which can each be used alone to completely satisfy the remaining demand for resource i). Choose j^* such that

$$c_j^* d_j'^* = \begin{cases} \min_{j \in J} c_j d_j' & \text{if } J \neq \phi \\ \min_{j \in F} c_j / a_{ij} & \text{otherwise.} \\ a_{ij} > 0 \end{cases} \quad [3.22]$$

If $J \neq \phi$ then j^* is the index of the variable which minimizes the cost of satisfying constraint i , assuming only a single variable is to be used and this variable will be fixed at its current upper bound. If $J = \phi$, j' is the index of the variable which has the minimum cost per unit of resource i supplied.

3. Fix x_{j^*} equal to d_{j^*}' .

When the value of a fixed variable is to be changed in the back-tracking step, it is decreased by one. Thus, a variable j assumes its admissible values in the order d_j' , $d_j' - 1$, ..., 0.

In order to demonstrate the CSCEP algorithm, it was applied without the iteratively computed bounds to problem [3.2, 3.3, 3.4]. The steps are displayed in Table 2.

The example below illustrates the application of the algorithm to a hypothetical problem. Assume a computer center currently has the following resources:

1 CPU

1 megabyte of memory

2 memory expansion slots

Table 2
An Example of the CSCEP Algorithm

Stage	S	X_S	$z[S]$	d_1'	d_2'	d_3'	z_S	Comments
i	(3)	(2)	6	2	0	2	10	Set $X_3 = d_3'$ to satisfy [3.4].
ii	(3,1)	(2,2)	10	2	0	2	10	Set $X_1 = d_1'$ to satisfy [3.3]. Incumbent solution. Backtrack.
iii	(3,1)	(2,1)	8	1	0	2	—	Violates [3.3]. Backtrack.
iv	(3, <u>1</u>)	(2,0)	6	0	0	2		Violates [3.3]. Backtrack.
v	(3)	(1)	3	2	1	1	9	Enumerate further.
vi	(3,2)	(1,1)	5	2	1	1	9	Set $X_2 = d_2'$ to satisfy [3.4]. Enumerate further.
vii	(3,2,1)	(1,1,2)	9	2	1	1	9	Set $X_1 = d_1'$ to satisfy [3.3]. New incumbent. Backtrack.
viii	(3,2,1)	(1,1,1)	7	1	1	1	—	Violates [3.3]. Backtrack.
ix	(3,2, <u>1</u>)	(1,1,0)	5	0	1	1	—	Violates [3.3]. Backtrack.
x	(3, <u>2</u>)	(1,0)	3	2	0	1	—	Violates [3.3]. Backtrack.
xi	(<u>3</u>)	(0)	0	2	2	0	—	No completion exists which satisfies [3.4]. End of enumeration. Result is $X_3 = 1$, $X_2 = 1$, $X_1 = 2$.

- 1 controller for type A disk drive with a capacity of four drives
- 2 type A disk drives with a capacity of sixty megabytes each
- air conditioning to support 10,000 additional BTUs
- 3 slots for additional peripheral expansion.

Future demand is determined to require 140 megabytes of disk space, two megabytes of memory, and a line printer. The formulation of the problem is:

$$\min z = \sum c_i x_i \quad [3.23]$$

$$x_1 + 0.5x_2 \geq 1 \quad [3.24]$$

$$60x_3 + 30x_4 \geq 20 \quad [3.25]$$

$$x_6 + x_7 \geq 1 \quad [3.26]$$

$$x_3 \leq 2 \quad [3.27]$$

$$-x_4 + 2x_5 \geq 0 \quad [3.28]$$

$$x_1 + x_2 \leq 2 \quad [3.29]$$

$$8000x_3 + 3000x_4 + 2000x_6 + 8000x_7 \leq 10000 \quad [3.30]$$

$$2x_5 + 2x_6 + x_7 \leq 3 \quad [3.31]$$

$$0 \leq x_1, x_2, x_3, x_4, x_5, x_6 \text{ and integer}$$

Each variable (x_i) and its cost (c_i) is defined in Table 3. The resources associated with each constraint are shown in Table 4. The enumeration steps are shown in Table 5.

Flexibility of the CSCEP Model

Since the model deals in net requirements, it applies to either the problem of designing an entirely new system or the expansion of an existing one. If, in the expansion of an existing system, the user wishes to consider the possibility of removing devices, this can also

Table 3
Devices Used in Hypothetical Example

	Description	Capacity	Cost (c_i)	Requirements
x_1	memory board A	1 megabyte	5,000	1 memory expansion slot
x_2	memory board B	0.5 megabytes	3,000	1 memory expansion slot
x_3	disk drive A	60 megabytes	25,000	controller, 8000 BTUs
x_4	disk drive B	30 megabytes	15,000	type B controller, 3000 BTUs
x_5	controller B	2 drives	10,000	2 peripheral slots
x_6	line printer A	1 printer	10,000	2 peripheral slots
x_7	line printer B	1 printer	10,000	1 peripheral slot

Table 4
Resources Represented by Constraints in CSCEP Example

Constraint	Resource
[3.24]	Memory
[3.25]	Disk space
[3.26]	Line printer
[3.27]	Type A controller capacity
[3.28]	Type B controller capacity
[3.29]	Memory expansion capacity
[3.30]	Air conditioning capacity
[3.31]	Peripheral expansion capacity

Table 5
Application of CSCEP Algorithm to a Hypothetical Example

Stage	S	X_S	$z[S]$	d_1'	d_2'	d_3'	d_4'	d_5'	d_6'	d_7'	z_S	Comments
i	(1)	(1)	5,000	1	0	1	1	0	1	1	23,333	Satisfy memory requirement. Set $X_1 = d_1'$.
ii	(1,4)	(1,1)	20,000	1	0	0	1	1	1	1	40,000	Satisfy disk requirements.
iii	(1,4,6)	(1,1,1)	30,000	1	0	0	1	1	1	0	40,000	Satisfy printer requirements. All external demands have now been satisfied.
iv	(1,4,6,5)	(1,1,1,1)	40,000	1	0	0	1	1	1	0	—	Satisfy controller requirement. Has no completion which satisfies [3.31]. Backtrack.
v	(1,4,6,5)	(1,1,1,0)	30,000	1	0	0	1	0	1	0	—	Has no completion which satisfies [3.28]. Backtrack.
vi	(1,4,6)	(1,1,0)	20,000	1	0	0	1	0	0	1	—	Enumerate further.
vii	(1,4,6,7)	(1,1,0,1)	30,000	1	0	0	1	0	0	1	—	Has no completion which satisfies [3.30]. Backtrack.
viii	(1,4,6,7)	(1,1,0,0)	20,000	1	0	0	1	0	0	0	—	Has no completion which satisfies [3.26]. Backtrack.

Table 5 (Continued)

Stage	S	X_S	$z[S]$	d_1'	d_2'	d_3'	d_4'	d_5'	d_6'	d_7'	zS	Comments
ix	(1, <u>4</u>)	(1, 0)	5,000	1	0	1	0	0	1	1	23,333	Only memory demand is satisfied.
x	(1, <u>4</u> , 3)	(1, 0, 1)	30,000	1	0	1	0	0	1	1	40,000	Satisfy disk requirement. Set $X_3 = d_3'$. Satisfy printer requirement.
xi	(1, <u>4</u> , 3, 6)	(1, 0, 1, 1)	40,000	1	0	1	0	0	1	0	40,000	Feasible solution. Save and backtrack.
xii	(1, <u>4</u> , 3, <u>6</u>)	(1, 0, 1, 0)	30,000	1	0	1	0	0	0	1	40,000	Lower bound on z equal to best known solution. Backtrack.
xiii	(1, <u>4</u> , <u>3</u>)	(1, 0, 0)	5,000	1	0	0	0	0	1	1	—	Has no completion which satisfies [3.25]. Backtrack.
xiv	(<u>1</u>)	(0)	0	0	2	1	1	0	1	1	24,333	No demands are satisfied.
xv	(<u>1</u> , 2)	(0, 2)	6,000	0	2	1	1	0	1	1	24,333	Satisfy memory demand.
xvi	(<u>1</u> , 2, 4)	(0, 2, 1)	31,000	0	2	0	1	1	1	1	46,000	Satisfy disk requirement. Set $X_4 = d_4'$. Lower bound on z exceeds best known solution. Backtrack.

Table 5 (Continued)

Stage	S	x_S	$z[S]$	d_1'	d_2'	d_3'	d_4'	d_5'	d_6'	d_7'	z_S	Comments
xvii	(<u>1</u> , 2, 4)	(0, 2, 0)	6,000	0	2	1	0	0	1	1	41,000	Lower bound on z exceeds best known solution. Backtrack.
xviii	(<u>1</u> , 2)	(0, 1)	3,000	0	1	1	1	1	1	1	—	Has no completion which satisfies [3.24]. Backtrack.
xix	(<u>1</u> , 2)	(0, 0)	0	0	0	1	1	1	1	1	—	Has no completion which satisfies [3.24]. No more feasible solutions. End of enumeration. Result is $x_1 = 1, x_3 = 1, x_6 = 1$.

be handled by the formulation. The existing system (in the model) would include only devices which will not be removed under any circumstances. The devices already in the system which can be removed will be treated as if they were not in the system but could be added at a cost equal to their salvage value.

Other special considerations can be included without any major changes in the model. In some cases only one device from a certain category, for example a CPU, may be included. This restriction can be modeled by a constraint of the form:

$$\sum_{j \in [J]} x_j \leq 1 \quad [3.31]$$

This constraint is of the same form as those given in formulation [3.1] and can be thought of as defining a fictitious resource.

In other cases more than one unit of a device may be included, but the devices may not be mixed with other devices of the same type. This restriction requires constraints of the form:

$$x_j \cdot x_k = 0 \quad [3.32]$$

Constraints of this type can be implemented via the bound redefinition procedure. When x_j becomes fixed at a non-zero value, d'_k is set to zero; and when x_k is fixed at a non-zero value, d'_j is set to zero.

Another type of consideration is that in which certain devices can be included in the system only if a particular device is also included. For example, certain hardware or software options may be compatible with one specific operating system. This restriction is modelled by the constraints of the form:

$$x_k - \sum_{\ell \in [L]} \alpha x_\ell \geq 0 \quad [3.33]$$

where $[L]$ is a set indexing these devices which require device k and

$$\alpha = \frac{1}{\sum_{\ell \in [L]} d_\ell} \quad [3.34]$$

Again the constraint can be thought of as defining a fictitious resource. Device k supplies one unit of this resource; and each device which requires device k consumes a small amount, α , of this resource. The value of α is chosen to be small enough that at most one unit of the resource will be required.

CHAPTER IV

IMPLEMENTATION AND COMPUTATIONAL RESULTS

The algorithm was implemented on a Digital Equipment Corporation PDP 11/34 minicomputer. The interactive menu-driven program is written in the Basic-Plus language. The problems on which the algorithm was tested were all solved in core using 32 kilobytes of memory or less.

The problems solved are derived from the historical data of Oak Ridge Associated Universities (ORAU). In 1979 ORAU purchased a PDP 11/34 computer with 256 kilobytes of core memory, two RL01 disk drives providing 10.4 megabytes of disk storage and interfaces for sixteen user terminals. In 1980 an RM02 disk drive providing 67 megabytes of additional disk storage and interfaces for four additional terminals were added to the system. In 1981 a proposal was written to justify the replacement of the PDP 11/34. The proposal defined the new system as having 768 kilobytes of memory, 125 megabytes of disk space, and servicing twenty-five user terminals. The proposal also called for the system to be expandable to 250 megabytes of disk space and to provide access for thirty user terminals. The equipment which was purchased included a PDP 11/44 minicomputer with 768 kilobytes of memory (expandable to one megabyte), and another RM02 disk drive which was used in conjunction with the previously purchased RM02 to provide a total of 134 megabytes of disk space. The expansion to the second capacity level described in the proposal is planned for the near future.

The model was tested on ten problems as shown in Table 6. The exogenous demands for resources in these problems were chosen from one of

Table 6. Computational Results for ORAU Problems

Problem	Cost	No. of a Devices (Variables)	No. of b Admissible Solutions	No. of Augmentation	No. of Complete Solutions Enumerated	CPU Seconds Required
1	\$ 49,550	25	8.64×10^{15}	21	6	21.4
2	82,940	25	8.64×10^{15}	123	28	117.9
3	110,290	25	8.64×10^{15}	223	29	206.2
4	145,570	25	8.64×10^{15}	898	7	1194.8
5	33,930	29	1.38×10^{17}	238	27	345.7
6	97,190	29	1.38×10^{17}	3567	53	4696.4
7	131,090	29	1.38×10^{17}	3006	11	3671.7
8	72,940	31	5.53×10^{17}	5581	42	6682.7
9	101,790	31	5.53×10^{17}	7111	27	8277.2
10	51,860	33	2.21×10^{18}	22618	33	30330.3

^aIncrease due to additional devices which are considered for removal after purchase.

^bComputed from the maximum number of each device available.

four levels: the levels provided by the systems installed in 1979, 1980 or 1981 or the projected level for a future systems as specified in the proposal for the 1981 system. The existing components around which the expansion would take place consisted of no components or those of the systems installed in 1979, 1980 or 1981. The ten problems shown in Table 6 are derived from the different combinations of existing systems and demand levels.

The problems solved had from twenty-five to thirty-three devices available. These included twenty-five devices chosen from a single vendor's catalogue plus those devices in the existing system. These devices are shown in Table 7. The nineteen different resources are shown in Table 8. Hence, the problems had at least twenty-six variables and nineteen constraints. The salvage value of devices in the existing system was assumed to be zero.

The optimal costs of each of the problems is also given in Table 6. The components of each solution are given in Appendix I. The components used in the solutions to problems, 1, 5 and 8 are very similar to the systems installed in 1979, 1980 and 1981, respectively. This is not surprising since the exogenous requirements for these problems are respectively exactly equal to the resources provided by these systems.

Table 6 shows that as the exogenous demands increase the number of augmentation steps also tends to increase. This is simply because as the demands increase a larger number of devices are required in a feasible solution and the dynamic upper bounds cannot be as tight.

The data also shows that expansion problems are more difficult to solve than those problems in which an entirely new system is being

Table 7

Devices Available for Solutions to ORAU Problems

CPUs

PDP 11/34 computer
PDP 11/44 computer

Memory

MS11-MB 256 Kb memory board
MS11-MC 512 Kb memory board
MS11-MD 768 Kb memory board

Communications Options

DL11 single line interface
DZ11-A 8-line asynchronous multiplexers
DZ11-E 16-line asynchronous multiplexers

Disk Drives

RK711-PA 28 mb disk drive with controller
RK07 28 mb disk drive
RL11-AK 5.2 mb drive with controller
RL01 5.2 mb drive
RL211-AK 10.4 mb drive with controller
RL02 10.4 mb drive
RJM02-AA 67 mb drive with controller
RM02 67 mb drive
RWP06 176 mb drive with controller
RP06 176 mb drive

Expansion Mounting Hardware

DD11-CK expansion backplane
DD11-DK expansion backplane
BA11-K UNIBUS expansion box
BA11-L UNIBUS expansion box
DB11-A UNIBUS repeater

Cabinets

H960 cabinet
H9642 cabinet

Other Peripherals

LP11-A line printer

Table 8
Requirements Used in ORAU Problems

Exogenous Requirements	Endogenous Requirements
CPU	<u>Environmental</u>
Disk storage	Heat dissipation
Core memory	Floor space
80 column printers	<u>System</u>
132 column printers	Quad slots
Terminal interfaces	Hex slots
	Bus loads
	System units
	5.25 inch distribution panel
	Memory expansion locations
	<u>Peripherals</u>
	RL02 controller capacity
	RK07 controller capacity
	RM02 controller capacity
	RL01 controller capacity
	RP06 controller capacity

configured. This is due to the fact that the augmentation procedure always first chooses the existing devices (which have a cost of zero). However, these devices may use resources inefficiently. Thus, even at a cost of zero their use is suboptimal, or in some cases infeasible. This characteristic of the algorithm can be ameliorated by including (for augmentation purposes only) an estimate of the resources it consumes in the cost of a device. For example, the augmentation cost could be defined as

$$c'_j = - \sum_{a_{ij} < 0} e_i a_{ij} + c_j \quad [5.1]$$

for $j = 1, 2, \dots, n$

where e_i is the minimum per unit cost of providing resource i .

$$e_i = \min_{j = 1, 2, \dots, m} \frac{c_j}{a_{ij}} \quad [5.2]$$

for $i = 1, 2, \dots, n$.

This modification does not, however, prevent the augmentation procedure from choosing devices which inefficiently use a resource which has zero cost but is in limited supply. This was the case in problems 7, 8 and 10 in which limited floor space precluded the inclusion of some of the devices of the old system in a feasible solution. Obviously if the user can anticipate such problems, he could manipulate the augmentation costs to speed up the algorithm.

Problem 4 shows that the total cost of building the final system from scratch is \$145,570. On the other hand, the minimum cost of the

stepwise expansion of the system without the benefit of forecasts is the sum of the solutions of problems 1, 5, 8 and 10: \$219,420. The difference of \$75,850 represents the value of an accurate 1979 forecast of future requirements, ignoring the value of money and equipment price changes.

CHAPTER V

TOPICS FOR RESEARCH

The configuration of a computer system involves two optimization steps, (1) determining the optimal specification of device capabilities, and (2) selecting the hardware which provides these capabilities at a minimum cost. Thus the ultimate objective in configuring a system may be initially expressed in terms of performance, but this objective is translated into a rigid set of system parameters which are used as constraints in the CSCEP model. Ideally these two optimization steps should be combined with the performance measures used as constraints or objectives in the CSCEP model. Since most performance measures are nonlinear functions of the device capabilities this approach would be more difficult computationally.

The CSCEP model is a static model in that it determines how to optimally meet a fixed set of user requirements. A model which develops an optimal sequence of expansions for meeting time-phased requirements would be a useful extension.

Finally, the CSCEP model could be used for expansion problems other than those of computer systems. It is likely that many expansion problems exist which have a structure similar to the CSCEP.

LIST OF REFERENCES

LIST OF REFERENCES

1. Artis, H. P., "Predicting the Behavior of Secondary Storage Management Systems for IBM Computers," in F. J. Kylstra (ed.), Performance '81: Proceedings of the Eighth International Symposium on Computer Performance Modelling, Measurement and Evaluation, North-Holland Publishing Co., Amsterdam, 1981, pp. 435-443.
2. Badel, M. and Shum, A. V. Y., "Accuracy of an Approximate Computer System Model," in H. Beilner and E. Gelenbe (eds.), Modelling and Performance Evaluation of Computer Systems, North-Holland Publishing Co., Amsterdam, 1978, pp. 11-33.
3. Baker, K. R., Dixon, P., Magazine, M. J. and Silver, E. A., "An Algorithm for the Dynamic Lot-Size Problem with Time Varying Production Capacity Constraints," Management Science, Vol. 24 (1978), pp. 1710-1720.
4. Bekkers, Y. and Decouty, B., "A Versatile Programmable Hardware Monitor," in H. Beilner and E. Gelenbe (eds.), Modelling and Performance Evaluation of Computer Systems, North-Holland Publishing Co., Amsterdam, 1977, pp. 423-425.
5. Buzen, J. P., "Operational Analysis: An Alternative to Stochastic Modelling," in D. Ferrari (ed.), Performance of Computer Installations, North-Holland Publishing Co., Amsterdam, 1978, pp. 175-194.
6. Chandy, K. M., "Capacity Planning: The Art of Reasonable Decision Making," Infosystems, Vol. 29, No. 4 (April 1982), pp. 88-90.
7. Erlenkotter, D., "Sequencing Expansion Problems," Operations Research, Vol. 21 (1973), pp. 542-553.
8. Fayolle, G. and Robin, M., "Optimal Queueing Policies in Multiple Processor Computers," in H. Beilner and E. Gelenbe (eds.), Modelling and Performance Evaluation of Computer Systems, North-Holland Publishing Co., Amsterdam, 1977, pp. 103-117.
9. Friedenfelds, J. and McLaughlin, C. D., "A Heuristic Branch-and-Bound Algorithm for Telephone Feeder Capacity Expansion," Operations Research, Vol. 27 (1979), pp. 567-582.
10. Gomaa, H., "A Modelling Approach to the Evaluation of Computer System Performance," in H. Beilner and E. Gelenbe (eds.), Modelling and Performance Evaluation of Computer Systems, North-Holland Publishing Co., Amsterdam, 1978, pp. 171-199.
11. Gordon, W. J. and Newell, G. F., "Closed Queueing Systems with Exponential Servers," Operations Research, Vol. 15 (1967), pp. 254-265.

12. Jackson, J. R., "Networks of Waiting Lines," *Operations Research*, Vol. 5 (1957), pp. 518-521.
13. Jackson, J. R., "Jobshop Like Queueing Systems," *Management Science*, Vol. 10 (1963), pp. 131-142.
14. Jacoby, H. D. and Loucks, D. P., "Combined Use of Optimization and Simulation Models in River Basin Planning," *Water Resources Research*, Vol. 8 (1972), pp. 1401-1414.
15. Kachhal, S. K. and Arrora, S. R., "Seeking Configurational Optimization in Computer Systems," *Proceedings of the 1975 Annual Conference of the Association for Computing Machinery*, pp. 96-101.
16. Kameda, H., "Appraisals of Scheduling Decisions in Computing Systems Using a Finite-Source Queueing Model," in F. J. Kylstra (ed.), *Performance 81: Proceedings of the Eighth International Symposium on Computer Performance Modelling, Measurement and Evaluation*, North-Holland Publishing Co., Amsterdam, 1981, pp. 435-443.
17. Kriebel, C. H. and Raviv, A., "An Economics Approach to Modelling the Productivity of Computer Systems," *Management Science*, Vol. 26 (1980), pp. 297-311.
18. Lambrecht, M. and Vander Eecken, J., "A Capacity Constrained Single-Facility Dynamic Lot-Size Model," *European Journal of Operational Research*, Vol. 2 (1978), pp. 132-136.
19. Liu, M., "A Simulation Study of Memory Partitioning Algorithms," in D. Ferrari (ed.), *Performance of Computer Installations*, North-Holland Publishing Co., Amsterdam, 1978, pp. 158-173.
20. Luss, H., "A Capacity Expansion Model for Two Facility Types," *Naval Research and Logistics Quarterly*, Vol. 27 (1979), pp. 291-303.
21. Mitra, D., "Waiting Time Distributions from Closed Queueing Network Models of Shared-Processor Systems," in F. J. Kylstra (ed.), *Performance '81: Proceedings of the Eighth International Symposium on Computer Performance Modelling, Measurement and Evaluation*, North-Holland Publishing Co., Amsterdam, 1981, pp. 113-131.
22. Mitten, L. G. and Tsou, C. A., "Efficient Solution Procedure for Certain Scheduling and Sequencing Problems," in S. E. Elmagharby, (ed.), *Symposium on the Theory of Scheduling and Its Applications*, Springer-Verlag, New York, 1973, pp. 127-142.
23. O'Laoghaire, D. T. and Himmelblau, D. M., "Optimal Capital Investment in the Expansion of an Existing Water Resources System," *Water Resources Bulletin*, Vol. 8 (1971), pp. 1194-1209.
24. Smith, R. L., "Turnpike Results for Single Location Capacity Expansion," *Management Science*, Vol. 25 (1979), pp. 474-484.

25. Stevens, B., "Capacity Management and DP Planning: A Basic Approach," Computerworld, Vol. 15, No. 36 (September 7, 1981), pp. In Depth/1-In Depth/16.
26. Trivedi, K. S., "Analytic Modeling of Computer Systems," Computer, Vol. 11, No. 10 (October 1978).
27. Trivedi, K. S. and Kinicki, R. E., "A Mathematical Model for Computer System Configuration Planning," in D. Ferrari (ed.), Performance of Computer Installations, North-Holland Publishing Co., Amsterdam, 1978, pp. 323-332.
28. Trivedi, K. S. and Sigmon, T. M., "A Performance Comparison of Optimally Designed Computer Systems with and without Virtual Memory," SIGARCH Newsletter, Vol. 7, No. 6 (April 1979), pp. 117-121.
29. Trivedi, K. S. and Wagner, R. A., "A Decision Model for Closed Queueing Networks," IEEE Transactions on Software Engineering, Vol. SE-5, No. 3 (July 1979), pp. 328-332.
30. Trivedi, K. S., Wagner, R. A. and Sigmon, T. M., "Optimal Selection of CPU Speed Device Capacities and File Assignments," Journal of the Association for Computing Machinery, Vol. 28 (1981), pp. 270-288.
31. Trotter, L. E. and Shetty, C. M., "An Algorithm for the Bounded Variable Integer Programming Problem," Journal of the Association for Computing Machinery, Vol. 21 (1974), pp. 505-513.
32. Vincent, D. R., "Performance Standards for Capacity Planning," Computerworld Extra! (1981), pp. 39-48.
33. Yaged, B., Jr., "Minimum Cost Routing for Dynamic Network Models," Networks, Vol. 10 (1973), pp. 193-224.

APPENDICES

APPENDIX A

DESCRIPTIONS OF ORAU PROBLEMS

Problem 1: Initial Configuration of First System

Demand: 10 megabytes of disk storage space
16 terminal lines
1 line printer
1 CPU

Configuration: 1 PDP 11/34 computer
1 RL211-AK disk system
1 DZ11-E terminal interface
1 LP11-A line printer

Problem 2: Initial Configuration of Second System

Demand: 77 megabytes of disk storage space
20 terminal lines
1 line printer
1 CPU

Configuration: 1 PDP 11/34 computer
1 RL211-AK disk system
1 DZ11-A terminal interface
1 DZ11-E terminal interface
1 H9642 cabinet
1 RJM02-AA disk system

Problem 3: Initial Configuration of Third System

Demand: 125 megabytes of disk storage space
25 terminal lines
750 kilobytes of internal memory
1 line printer
1 CPU

Solution: 1 PDP 11/44 computer
1 DD11-CK expansion backplane
2 DL11 terminal interface
1 RWP06 disk system
1 MS11-MC memory board
1 DZ11-A terminal interface
1 DZ11-E terminal interface
1 LP11-A line printer
1 H9642 cabinet

Problem 4: Initial Configuration of Fourth System

Demand: 250 megabytes of disk storage space
30 terminal lines
750 kilobytes of internal memory
1 line printer

Solution: 1 PDP 11/44 computer
1 RWP06 disk system
1 RP06 disk drive
1 MS11-MC memory board
2 DZ11-E terminal interface
1 LP11-A line printer
1 DD11-DK expansion backplanes
1 H9642 cabinet

Problem 5: Expansion from First System to Second

Demand: See Problem 2

Solution: 1 RJM02-AA disk system
1 DZ11-A terminal interface
1 H9642 cabinet

Problem 6: Expansion from First System to Third

Demand: See Problem 3

Solution: 1 PDP 11/44 computer
1 MS11-MC memory board
1 DL11 terminal interface
1 RWP06 disk system
1 DZ11-A terminal interface
1 H9642 cabinet

Problem 7: Expansion from First System to Fourth

Demand: See Problem 4

Solution: 1 PDP 11/44 computer
1 RJM02-AA disk system
1 RWP06 disk system
1 MS11-MC memory board
1 DZ11-E terminal interface
1 DD11-DK expansion backplane
1 BA11-L UNIBUS expansion box
1 H960 cabinet
1 existing LP11-A line printer
1 existing DZ11-E terminal interface

Problem 8: Expansion from Second System to Third

Demand: See Problem 3

Solution: 1 PDP 11/44 computer
1 DD11-CK expansion backplane
1 DL11 terminal interface
1 MS11-MC memory board
1 DZ11-A terminal interface
1 H9642 cabinet
1 RM02 disk drive
1 Existing RJM02-AA disk system
1 Existing DZ11-E terminal interface
1 Existing LP11-A line printer

Problem 9: Expansion from Second System to Fourth

Demand: See Problem 4

Solution: 1 PDP 11/44 computer
2 RWP06 disk system
1 MS11-MC memory board
1 DZ11-E terminal interface
1 DD11-DK expansion backplane
1 BA11-L UNIBUS expansion box
2 H960 cabinet
1 Existing RJM02-AA disk system
1 Existing RL211-AK disk system
1 Existing DZ11-E terminal interface
1 Existing LP11-A line printer

Problem 10: Expansion from Third System to Fourth

Demand: See Problem 4

Solution: 1 RWP06 disk system
1 DZ11-A terminal interface
1 DD11-DK expansion backplane
1 BA11-L UNIBUS expansion box
1 H960 cabinet
1 Existing RJM02-AA disk system
1 Existing RL211-AK disk system
1 Existing DZ11-E terminal interface
1 Existing LP11-A line printer
1 Existing DZ11-A terminal interface
1 H9642 cabinet

APPENDIX B

ENUMERATION PROGRAM LISTING

The following is a listing of the enumeration program used to solve the ORAU problems. Additional programs are necessary to store and retrieve data used in the process. System logicals may be used to specify different devices. Although the program is implemented on a PDP 11/34, it can be transported to any PDP 11 series machine. In addition, it should be possible to convert the program for use on most microcomputers with limited modifications. There are no special functions or file structures used which would make the conversion of the algorithm to most other programming languages difficult.

1

```

!
! PROGRAM NAME: TREE4.BAS
! AUTHOR: RICHARD S. BUTTURINI
! OAK RIDGE ASSOCIATED UNIVERSITIES, INC.
! P. O. BOX 117
! OAK RIDGE, TENNESSEE 37831-0117
!

```

```

! DESCRIPTION: TREE4 is a BASIC-PLUS program written
! on a PDP 11/34 computer to apply an implicit enumera-
! tion technique to a problem of configuring a computer
! system to meet pre-defined demand. The components
! included in the configuration are stored in a file
! named OPTION.DAT on disk drive DR1:. The demand
! to be met is stored in a file called DR1:DEMAND.DAT.
! The resources existing in the current system are in
! file DR1:CURRENT.SYS.
!

```

```

! FILE OPTION.DAT
! BYTES 1-32 NAME OF OPTION
! 33-40 COST OF OPTION (REAL VARIABLE WITH IMPLIED
! DECIMAL)
! 41-42 MAXIMUM NUMBER OF DEVICES USABLE (INTEGER)
! 43-50 EFFECT ON FIRST RESOURCE (REAL VARIABLE
! WITH IMPLIED DECIMAL)
! 51-58 EFFECT ON SECOND RESOURCE
! .
! .
! .
! 491-498 EFFECT ON FIFTY-EIGHTH RESOURCE
!

```

```

! FILE CURRENT.SYS
! BYTES 1-8 CURRENT AMOUNT OF FIRST RESOURCE
! 9-16 CURRENT AMOUNT OF SECOND RESOURCE
! .
! .
! .
! 457-464 CURRENT AMOUNT OF FIFTY-EIGHTH RESOURCE
!

```

```

! FILE DEMAND.DAT
! RECORD n IS DEMAND IN PERIOD n
! BYTES 1-8 DEMAND FOR FIRST RESOURCE
! 9-16 DEMAND FOR SECOND RESOURCE
! .
! .
! .
! 457-464 DEMAND FOR FIFTY-EIGHTH RESOURCE
!

```

```

! KEY VARIABLES -
!

```

```

! PATH IS THE CURRENT PATH THROUGH THE TREE
! OPTION IS AN ARRAY OF THE DEVICES' EFFECT ON EACH
! RESOURCE
! CUR.STATE IS AN ARRAY CONTAINING THE CURRENT AMOUNT
!

```

```

!           OF EACH RESOURCE AT THIS POINT IN THE ENUMERATION      &
!           FIXED IS AN ARRAY CONTAINING THE LEVEL AT WHICH EACH &
!           DEVICE IS FIXED AT THIS POINT IN THE ENUMERATION, OR &
!           -1 IF THE DEVICE IS FREE                                &
!           SOLUTION IS AN ARRAY CONTAINING THE LEVEL AT WHICH    &
!           EACH DEVICE IS FIXED IN THE BEST SOLUTION YET FOUND  &
!
1000  EXTEND
1100  ON ERROR GOTO 19000
1200  OPEN 'DR1:OPTION.DAT' AS FILE 2% !File of options           &
\      DIM EFFECT$(58)                                           &
\      FIELD #2%, 32% AS DESCRIPT$,8% AS COST$,2% AS MAX$      &
\      FIELD #2%,34%+A%*8% AS DUM$,8% AS EFFECT$(A%)          &
\      FOR A%=1% TO 58%                                          &
\      GET #2%,RECORD 1%           !Get number of options      &
\      OPTION.CNT%,FEAS.SOL%=0%                                  &
\      DIM MAXIMUM$(40),SOLUTION$(40%),COST(40)                &
\      DIM OPTION(40,50),BETA(50)                               &
\      PATH$=''           !Initialize path                      &
\      MAT SOLUTION%=ZER                                         &
\      FOR A%=1% UNTIL DESCRIPT$=STRING$(32%,0%)              &
\      OPTION.CNT%=1%+OPTION.CNT%                               &
\      GET #2%,RECORD A%           !Read data                  &
\      OPTION(A%,I%)=CVT$(EFFECT$(I%)) FOR I%=1% TO 50%        &
\      COST(A%)=CVT$(COST$)      !Save cost of option          &
\      MAXIMUM$(A%)=CVT$(MAX$)    !Save maximums               &
\      NEXT A%                                                   &
\      MASK$='#,###,###.##'    !Mask for output                &
\      DIM CUR.STATE(58),CUR$(58)    !File of current status &
\      OPEN 'DR1:CURRNT.SYS' AS FILE 2%                          &
\      FIELD #2%,8%*(A%-1%) AS DUM$,8% AS CUR$(A%) FOR        &
\      A%=1% TO 58%                                             &
\      GET #2%,RECORD 1%                                         &
\      CUR.STATE(A%)=CVT$(CUR$(A%)) FOR A%=1% TO 58%          &
1225  FOR I%=50% TO 1% STEP -1%           !Determine number of options &
\      FOR A%=1% TO OPTION.CNT%-1%          &
\      IF OPTION(A%,I%)<>0 THEN RES%=I%      &
\      GOTO 1250                                              &
1230  NEXT A%                                               &
\      NEXT I%                                               &
1250  INPUT'Enter number of periods';TOTAL.TIME%           &
\      INPUT'Enter maximum cost, <RET> for none';BEST.YET.COST &
\      !Enter maximum cost to be considered                  &
\      TOTAL.COST,AUGMENT=0           !Initialize             &
\      OPEN 'DR1:DEMAND.DAT' AS FILE 2%           !File of demands &
\      DIM DEMAND$(58),DEMAND(58)                  &
\      FIELD #2%,8%*(A%-1%) AS DUM$,8% AS DEMAND$(A%)        &
\      FOR A%=1% TO RES%                                  &
\      GET #2%,RECORD TOTAL.TIME%           !Get final demand &
\      DEMAND(A%)=CVT$(DEMAND$(A%)) FOR A%=1% TO RES%        &

```

```

\      DIM FIXED%(40)                                &
\      FIXED%(A%)=-1% FOR A%=1% TO 40% !Free all X's    &

1300  GOSUB 12000          !Partition solutions          &
\      AUGMENT=AUGMENT+1      !Count augmentations      &

1350  GOSUB 12500          !Check bounds on cost and constraints &
\      IF BEST.YET.COST>0 AND    &
\      LOW.COST>=BEST.YET.COST OR LOW.COST<0 THEN      &
\      GOSUB 11000          !If this solution can be fathomed, do so&
\      GOTO 1350              &

1400  FOR A%=1% TO RES%      !Compare with demand        &
\      IF DEMAND(A%)>CUR.STATE(A%) OR    &
\      CUR.STATE(A%)<0 THEN    &
\      GOTO 1300              !Need more, augment again    &

1500  NEXT A%              &

1600  MAT SOLUTION%=FIXED%    !!SOLUTION!!              &
\      BEST.YET.COST=TOTAL.COST      !Save cost          &
\      GOSUB 11000          !Fathom                    &
\      FEAS.SOL%=FEAS.SOL%+1% !Keep track of how many    &
\      GOTO 1400              !Look for better solution    &

3100  !      Print results of best solution              &
\      PRINT TAB(10%);'Quantity';TAB(20%);'Item';TAB(70%); &
\      'Cost'                                          &
\      OPEN 'DR1:OPTION.DAT' AS FILE 2%              &
\      FOR A%=1% TO OPTION.CNT%      !Pick out those included&
\      IF SOLUTION%(A%)>0% THEN GET #2%,RECORD A%      &
\      PRINT TAB(12%);SOLUTION%(A%);TAB(20%);DESCRIPT$; &
\      TAB(62%);                                          &
\      PRINT USING MASK$,SOLUTION%(A%)*CVT$(F(COST$))/100 &

3200  NEXT A%              &
\      PRINT                                          &
\      PRINT TAB(62%);                                &
\      PRINT USING MASK$,BEST.YET.COST/100      !Print total &
\      CHAIN 'DR1:CPMENU'          !Go back to menu      &

11000 !      Subroutine to fathom node                  &

11100 LAST.NODE$=RIGHT(PATH$,LEN(PATH$)-1%)          &
\      OPT%=CVT$(LAST.NODE$)!Get last option increased &
\      TOTAL.COST=TOTAL.COST-COST(OPT%) IF FIXED%(OPT%)>0% &
\      CUR.STATE(J%)=CUR.STATE(J%)-OPTION(OPT%,J%)    &
\      FOR J%=1% TO RES% IF FIXED%(OPT%)>0%      !Modify system &
\      FIXED%(OPT%)=FIXED%(OPT%)-1%      !Remove cost & decrement&
\      IF FIXED%(OPT%)<0% THEN                      &
\      PATH$=LEFT(PATH$,LEN(PATH$)-2%) !Free when zero    &
\      IF LEN(PATH$)=0% THEN 3100 ELSE GOTO 11100      &

```

```

11200 RETURN &
12000 ! Subroutine to partition solutions &
12100 THRU%=0% !Indicator of number of times search is made &
12115 FOR A%=1% TO RES% !Go through resources &
\ GOTO 12300 IF CUR.STATE(A%)>=DEMAND(A%) !Compare &
\ EXTRA=0 &
12125 GOTO 12300 IF OPTION(I%,A%)<0 AND FIXED%(I%)<0% AND THRU%=0% &
\ AND MAXIMUM%(I%)>0% FOR I%=1% TO OPTION.CNT%-1% &
\ NXT.ADD%=0% !Initialize for partitioning &
\ NXT.ADD.CNT%,CHEAPEST%,CHEAPEST.CNT%=0% &
\ NXT.ADD.COST,TOTAL.AVAIL,FEAS=0 &
\ EXTRA=0 &
\ PER.UNIT=-1 &
\ EXTRA=EXTRA-OPTION(I%,A%)*MAXIMUM%(I%) IF &
\ OPTION(I%,A%)<0 AND FIXED%(I%)<0 FOR I%=1% TO &
\ OPTION.CNT%-1% IF THRU%=1% &
\ FOR I%=1% TO OPTION.CNT%-1% &
\ GOTO 12200 IF FIXED%(I%)>=0% !Ignore if fixed &
\ OR MAXIMUM%(I%)=0% &
\ IF OPTION(I%,A%)<=0 THEN 12200 ELSE NEED=(DEMAND(A%)- &
\ CUR.STATE(A%)+EXTRA)/OPTION(I%,A%) !How much is needed? &
\ UPPER%=FIX(NEED) &
\ UPPER%=UPPER%+1% IF FIX(NEED)<NEED !Not too much &
\ FEAS=FEAS+OPTION(I%,A%)*MAXIMUM%(I%) &
\ IF NXT.ADD%=0% AND MAXIMUM%(I%)<NEED &
\ AND MAXIMUM%(I%)>0% AND (PER.UNIT> &
\ COST(I%)/OPTION(I%,A%) OR PER.UNIT<0) THEN CHEAPEST%=I% &
\ PER.UNIT=COST(I%)/OPTION(I%,A%) &
12150 NEED=NEED+1 IF NEED>FIX(NEED) &
\ NEED%=FIX(NEED) !Convert to integer solution &
\ GOTO 12200 IF MAXIMUM%(I%)<NEED%!Is it within limits &
\ IF NXT.ADD%=0% OR NXT.ADD.COST>NEED%*COST(I%) &
\ THEN NXT.ADD%=I% !This is the best so far &
\ NXT.ADD.CNT%=NEED% &
\ NXT.ADD.COST=COST(I%)*NEED% !Save values &
12200 NEXT I% &
\ IF DEMAND(A%)>CUR.STATE(A%)+FEAS THEN GOSUB 11000 &
\ GOTO 12100 !No feasible solution possible &
12225 IF NXT.ADD%=0% AND CHEAPEST%=0% &
\ THEN GOSUB 11000 !No feasible solution, fathom &
\ GOTO 12100 !Try again &
12250 IF NXT.ADD%=0% THEN NXT.ADD%=CHEAPEST% &
\ NXT.ADD.CNT%=MAXIMUM%(CHEAPEST%) &
\ GOTO 12300 !None found, take cheapest &
12270 GOTO 12350 IF A%=RES% &

```

```

\      FOR I%=A%+1% TO RES%      !How many should be added?      &
\      IF OPTION(NXT.ADD%,I%)>0% THEN      &
\      IF CUR.STATE(I%)+OPTION(NXT.ADD%,I%)*NXT.ADD.CNT%<      &
DEMAND(I%) THEN NEED=(DEMAND(I%)-CUR.STATE(I%))/      &
OPTION(NXT.ADD%,I%)      !Need to add some more      &
\      NEED=NEED+1 IF NEED>FIX(NEED)      !Round amount needed up &
\      NXT.ADD.CNT%=FIX(NEED)      !Convert to integer      &
\      IF NXT.ADD.CNT%>MAXIMUM%(NXT.ADD%) THEN NXT.ADD.CNT%=      &
MAXIMUM%(NXT.ADD%)      !Don't exceed maximum      &
\      GOTO 12350      &

12272 NEXT I%      &
\      GOTO 12350      &

12300 NEXT A%      &
\      IF THRU%=0% THEN THRU%=1%      &
\      GOTO 12115      !Try again if no option gives enough      &

12350 GOTO 1600 IF NXT.ADD.CNT%=0%      !Solution is feasible      &
\      FEAS=NXT.ADD.CNT%      !Add in data      &
\      SMALLEST.BOUND%=FIX(FNMIN(FNMAX.CNT(NXT.ADD%),FEAS))      &
\      FIXED%(NXT.ADD%)=SMALLEST.BOUND%      &
\      TOTAL.COST=TOTAL.COST+COST(NXT.ADD%)*SMALLEST.BOUND%      &
\      PATH$=PATH$+CVT$(NXT.ADD%)      !Add to current path      &
\      CUR.STATE(J%)=CUR.STATE(J%)+OPTION(NXT.ADD%,J%)*      &
SMALLEST.BOUND% FOR J%=1% TO RES%      &

12400 RETURN
12500 !      Tighten resource constraints      &
!      &
FOR R%=1% TO RES%      &
\      BETA(R%)=DEMAND(R%)      &
\      BETA(R%)=0 IF BETA(R%)<0      &
\      NEXT R%      &
\      FOR I%=1% TO 10%      &
\      OK%=0%      !Set close-enough flag      &
\      FOR R%=1% TO RES%      &
\      U=BETA(R%)      &
\      B=DEMAND(R%)      &
\      SUM1=0      &
\      FOR C%=1% TO RES%      &
\      GOSUB 16500      &
\      SUM1=SUM1+RATIO*BETA(C%)      &
\      NEXT C%      &
\      BETA(R%)=SUM1-B      !Gauss-Seidel      &
\      OK%=1% IF ABS(BETA(R%)-U)/ABS(U)>0.0001 IF U<>0      &
\      GOTO 12550 IF BETA(R%)-CUR.STATE(R%)<=0      &
\      FOR A%=1% TO OPTION.CNT%-1%      !Can demand be met?      &
\      GOTO 12550 IF FIXED%(A%)<0% AND OPTION(A%,R%)>0      &
AND MAXIMUM%(A%)>0%      &
\      NEXT A%      &
\      LOW.COST=-1      !No way      &
\      RETURN      &

```

```

12550 BETA(R%)=0 IF BETA(R%)<0                                &
      \      NEXT R%                                          &
      \      GOTO 12575 IF OK%=0%                            !Not getting better &
      \      NEXT I%                                          &

12575 IF BEST.YET.COST=0 THEN 12950                          &

12600 LOW.COST=TOTAL.COST      !Tighten lower bound on cost  &
      \      FOR A%=1% TO RES%                                &
      \      BETA(A%)=BETA(A%)-CUR.STATE(A%)                 &
      \      GOTO 12900 IF BETA(A%)<=0                       &
      \      RATIO=-1                                          &
      \      FOR I%=1% TO OPTION.CNT%-1%                     &
      \      GOTO 12650 IF(OPTION(I%,A%)<=0 AND FIXED%(I%)<0%) &
      \      OR FIXED%(I%)>=0% OR MAXIMUM%(I%)=0%            &
      \      FOR J%=1% TO A%-1%      !Might resource be elsewhere &
      \      IF OPTION(I%,J%)>0 AND BETA(J%)>0 THEN          &
      \      RATIO=0                                           &
      \      GOTO 12650                                         &

12625 NEXT J%                                                &
      \      RATIO=FNMIN(RATIO,COST(I%)/OPTION(I%,A%))        &

12650 NEXT I%                                                &
      \      LOW.COST=LOW.COST+RATIO*BETA(A%)                 !Add cost*ratio &
      \      IF RATIO<0 THEN LOW.COST=-1                      &
      \      RETURN                                             &

12900 NEXT A%                                                &

12950 RETURN                                                  &

13000 DEF FNMAX.CNT(OPT%)      !Determine how many can be supported &
      \      FEAS.SUB=-1                                        &
      \      FOR J%=1% TO RES%      !Go through resources    &
      \      IF OPTION(OPT%,J%)>=0 THEN 13100 ELSE SUPPLY=0   &
      \      FOR K%=1% TO OPTION.CNT%-1%      !Can it increase? &
      \      SUPPLY=SUPPLY+OPTION(K%,J%)*MAXIMUM%(K%) IF      &
      \      OPTION(K%,J%)>0 AND FIXED%(K%)<0% AND MAXIMUM%(A%)>0% &
      \      NEXT K%                                           &
      \      IF CUR.STATE(J%)+SUPPLY<0 THEN FEAS.SUB=0 ELSE NEED= &
      \      (CUR.STATE(J%)+SUPPLY)/ABS(OPTION(OPT%,J%))      &
      \      FEAS.SUB=FNMIN(FEAS.SUB,NEED)                    &

13100 NEXT J%                                                &
      \      FNMAX.CNT=FEAS.SUB                                &

13200 FNEND                                                  &

16500 !      Determine relationships between constraints      &
      !                                                         &
      \      RATIO=-1                                          &
      \      IF C%=R% THEN RATIO=0                            &
      \      RETURN                                             &

```

```
16550  FOR A%=1% TO OPTION.CNT%-1%                                &
      \      IF FIXED%(A%)<0% AND MAXIMUM%(A%)>0%                &
      THEN IF OPTION(A%,R%)<0 THEN IF OPTION(A%,C%)>0 THEN IF &
      RATIO>-1*OPTION(A%,R%)/OPTION(A%,C%) OR RATIO<0 THEN      &
      RATIO=-1*OPTION(A%,R%)/OPTION(A%,C%)                      &

16600  NEXT A%                                                    &
      \      RATIO=0 IF RATIO<0                                   &
      \      RETURN                                              &

17000  DEF FNMIN(X,Y)      !Minimum function                      &
      \      IF X>Y OR X<0 THEN FNMIN=Y ELSE FNMIN=X           &

17100  FNEND

19000  ON ERROR GOTO 0      !Error trap here if desired

32767  END
```

VITA

Richard Stewart Butturini was born in Oak Ridge, Tennessee, on November 7, 1953, and has remained a resident of that city. He graduated from Oak Ridge High School in June of 1971. He earned a Bachelor of Science degree with a major in Accounting from The University of Tennessee, Knoxville, in June of 1975.

After accepting a position as Programmer/Analyst in the Computer Services section of Oak Ridge Associated Universities, he began study toward a Master's degree at The University of Tennessee, Knoxville, as a part-time student. In August 1983, he earned a Master of Science degree with a major in Management Science. He and his wife, the former Janet Gentry, live in Oak Ridge and have a son, Russell.