DISCUSSION PAPER NO. 268

OPTIMAL PRICING OF COMPUTER RESOURCES IN A COMPETITIVE ENVIRONMENT

by

V. Balachandran

and

Edward A. Stohr

Graduate School of Management
Northwestern University
Evanston, Illinois 60201

September, 1978

ABSTRACT

In this paper, we develop a model for the determination of optimal prices for computer centers which compete against each other in a market characterized by groups of users with different computing needs. Two cases are considered. The first model views all computer centers acting as a block against a set of users who wish to maximize their revenue. The second model considers one system which optionally determines its prices against its competitors where all have a common market of users. The system resources for which prices are determined are processing time and space in main memory, space on auxiliary storage devices such as drum, disc, and tape and usage of input-output devices such as printers, card readers, and tape drives. The pricing problem is formulated as a max-min program. This non-linear programming problem is shown to be equivalent to a pair of mutually dual linear programs. The optimal prices and corresponding allocation of programs and data sets to devices can therefore be obtained in a computationally efficient manner.

1. Introduction

The markets for computer services vary from completely monopolistic (corporate data centers, and government and educational computing centers which have a captive clientele) to highly competitive (service bureaus and time-sharing vendors). In the monopolistic case either imposed priority systems or pricing (chargeback) systems together with user budgets may be used to guide users towards an efficient use of the available computing resources (see Kriebel, Raviv and Zia [8], Cotton [4], Nunamaker and Whinston [10], Smidt [11], [12]). The objective may be either to maximize some measure of user utility or to further the aims of the organization of which the computer center is a part. On the other hand, in the competitive case the problem is to determine an optimal pricing scheme for a computer center which competes against other computing centers in a market characterized by groups of users with different computing needs (e.g., scientific vs. commercial). Since the capacities and characteristics of the computing machinery owned by the computing centers are different, each center has the opportunity to specialize its prices and services to satisfy the needs of particular user groups.

This paper provides an operational method for sensitivity analysis of different pricing policies for (i) all computing centers acting together against a common market of computer users ('combined case') and (ii) for a single computer center in short-run competition with other existing computer centers ('single center case'). No attempt is made to analyze a global economic equilibrium, in the long run.

The prices which are to be determined are those for processing time and space in main memory, space on auxiliary storage devices such as drum, disc, mass-storage and tape and usage of input-output devices such as printers,

card readers and tape drives. In terms of access time to storage devices the kind of service a user can expect from a computer system will depend to a large extent on the devices on which his programs and data files are stored. However, overall turnaround times for batch jobs and response times for on-line jobs depend on the degree of overcrowding of the system since long queues can develop under peak-load conditions. To allocate system resources to competing jobs over time, various 'priority-pricing' and 'flexible pricing' schemes have been proposed (Smidt [11], Kleinrock [7], Ghanem [5], Nielsen [9]). One solution is to designate different rates (e.g. dollars per second) for different priority classes and to allow users to specify a priority class for each job before it is run. Jobs in the most expensive priority class are given preferential service. Under certain assumptions, the optimal rates can be derived by queueing theory and the analysis is usually in the context of a monopolistic system (Ghanem [5], Kleinrock [7]). Our approach to the time scheduling problem is to compute separate rates for computer time for peak and off-peak hours. In addition we simultaneously determine charges for the use of other system resources such as storage space and input-output time.

The model to be presented represents an extension of the methodology of Babad, Balachandran and Stohr [2], which was concerned with the minimum cost allocation of data sets to storage devices in the face of known prices. In contrast our objective in this paper is to provide a method for computing prices for computer system resources. We assume that there are a number of competing computer centers and that each user can place his programs and data files on any device of any system. The processing and storage demands of the users are assumed to be known and fixed and the set up and conversion costs to implement the users work on different systems are assumed to be approxi-

mately equal. Since the demand for computing services is derived from the particular applications involved it might be expected to be rather inelastic for existing computer applications. However this might not be true for new or prospective applications. In this case, the model can be run in an iterative fashion to obtain a solution if the nature of the relevant demand curves can be estimated.

The competitive environment that we assume is discussed in section 2. The optimization problem that potential users face in determining an optimal allocation of programs and data-sets to the devices of the various computing systems is described in section 3. In section 4 the optimization problem faced by the computing centers in the combined case is formulated as a max-min programming problem and the method of solution indicated. It is shown that the non-linear max-min problem can be reduced to a pair of mutually dual linear programs having a special structure in the constraint matrix. For the single center case, this model reduction results in a computationally efficient solution procedure. Finally, in section 5 we describe various other interpretations and applications of the model, with illustrations.

2. The Competitive Environment

To save notation we denote the cardinality of a set by the symbol for the set when there is no possibility for confusion. Consider an index set $L = \{1,2,\ldots,L\}$ of competing computer centers ('systems'). Users of computer resources are able to store their data sets (programs and data files) and run their programs on any one (or more) of these systems. Each computer center $\ell \in L$ has a central processing unit (CPU) in which the programs are run and a set $J_\ell = \{1,2,\ldots,j_\ell,\ldots,J_\ell\}$ of types of auxiliary storage devices such as drums, discs, magnetic tapes, datacells and card files on which the users can store their programs and data files. Again J_ℓ would probably be small--

in the range from 1 to 3 for most systems. The storage units have different physical characteristics in terms of storage capacity, access times to data and data transfer rates to central memory. The CPU's of the different systems also differ with respect to capacity and processing power. In addition to the CPU, users may use a number of other resources such as telephone ports for on-line terminals and input-output peripherals such as line-printers, card-readers and card-punches.

In the short run, the computer centers compete for the users on a price basis. The users are charged for the storage of their data sets on the various devices, for the input and output of this data to the CPU and for the various system resources used when jobs are processed. For illustrative processes we present a specific example of a possible set of pricing parameters. We will provide a more general notation later. Let the prices charged by computer center ℓ be as follows:

$a_{j\ell}$ = average dollar cost to access the first unit (e.g. byte, word or page) of data on device j.

$b_{j\ell}$ = average dollar cost per data unit to transfer data from device j to the CPU.

$c_{j\ell}$ = dollar cost per time period (week or month) to store one unit of data on device j.

d_{ℓ} = cost parameter related to program compute time (fixed cost per second; see (1)).

e_{ℓ} = cost parameter related to program compute time and size of program (variable cost per second; see (1)).

The processing charge on system ℓ to run a program which takes t_{ℓ} seconds of computer time and is m data units in length is assumed to be given by the following formula:[1]

[1] This formula is used by Vogelback Computing Center, Northwestern University (see Appendix C).

(1) $$d_{\ell} t_{\ell} + e_{\ell} t_{\ell} \cdot m$$

By 'compute time' in a multiprogramming or virtual storage environment we mean the time during which the program is actually executing rather than the (longer) time period for which the program (or parts of it) are resident in central memory. This measure has the advantage that it relates directly to the user's processing demands. The parameter e_{ℓ} allows for the differences in resources used by programs of different lengths. Other formulae for CPU utilization can be used in our model providing the price parameters appear linearly in the data. In addition, the model allows different computing centers to have different pricing formulae.

The demands on the systems are considered to be partitioned into K classes, forming an index set $K = \{1,2,\ldots,k,\ldots,K\}$. Typical usage classifications might be time-sharing versus batch or data processing versus scientific. Alternatively, each usage class can be identified with a particular group of users classified according to industry or on the basis of the type of application programs they run. Usage class k has $I_k = \{1,2,\ldots,i_k,\ldots,I_k\}$ data sets; a subset of these data sets $I_k^p = \{1,2,\ldots,i_k,\ldots,I_k^p\}$ are programs. Note that $I_k^p \subseteq I_k$. Programs have to be stored on the auxiliary storage devices and the costs to access and store programs are the same as those for data files. For $i_k \in I_k^p$ we let $U_k(i_k)$ denote the set of data-sets required to run program i_k. This set includes program i_k itself, together with the data files it references. To reduce data requirements more detailed information can be replaced by averages. To accomplish this we associate a multiplier, N_k, with each usage class. The data sets and programs used in the analysis can then be viewed as 'benchmarks'.

We suppose that data set $i_k \in I_k$, is accessed f_{ik} times per time period and that the length of data set i_k is m_{ik} data units. Depending on the application all or part of a data set may be accessed (transferred from the peripheral device to central memory). We let $m^a_{ik} \leq m_{ik}$ denote the average number of data units of data-set i of class k which are transferred to the CPU.[1] Continuing with the notation of the hypothetical example, if this data set is stored on device j of system ℓ the total access and storage costs for the time period are given by:

$$(2a) \qquad (a_{j\ell} + b_{j\ell} m^a_{ik}) f_{ik} + m_{ik} c_{j\ell} = a_{j\ell} f_{ik} + b_{j\ell} m^a_{ik} f_{ik} + c_{j\ell} m_{ik}$$

For a program data-set, $i_k \in I^p_k$, we let $t_{ik\ell}$ denote the average CPU time to run the program on system ℓ. From (1), since the program is run f_{ik} times, the total processing cost on system ℓ for the time period will be:

$$(2b) \qquad d_\ell t_{ik\ell} f_{ik} + e_\ell t_{ik\ell} m_{ik} f_{ik}$$

Thus if system ℓ chooses the prices $a_{j\ell}, b_{j\ell}, c_{j\ell}, d_\ell, e_\ell$ data set and program statistics indicated by (2a) and (2b) are required for costing purposes.

We now state the more general form of the pricing formulae which will be used in the remainder of this paper. Note that the price parameters in (2a) relate to the use of storage devices and data sets while those in (2b) relate to the resources used when programs are run. We will maintain this distinction but allow for more general pricing formulae and also for the fact that the different computing centers may use different sets of pricing parameters. Let $R_{j\ell} = \{1,2,\ldots,r_{j\ell},\ldots,R_{j\ell}\}$ be the index set of pricing parameters relating to data sets on device $j \in J_\ell$ which is used by computing center ℓ and let $p_{rj\ell}$ be the value of the $r^{th}_{j\ell}$ such parameter. Similarly, let $S_\ell = \{1,2,\ldots,s_\ell,\ldots,S_\ell\}$ be

---

[1] This average is across all applications. If the requisite information is available, the model can easily be altered to include data concerning the average requirements of each program for data from each file it accesses.

the index set of pricing parameters relating to the running of programs which is used by data center ℓ and let $q_{s\ell}$ be the value of the s_ℓ^{th} such parameter. For each data set $i \in I_k$ a total of $R_{j\ell}$ statistics must be provided for pricing purposes on system ℓ. Let $h_{rj\ell ik}$ denote the value of the statistic to be multiplied by $p_{rj\ell}$. Similarly, for each program $i \in I_k^p$ a total of S_ℓ statistics must be provided for pricing on system ℓ. Let $g_{s\ell ik}$ denote the value of the statistic to be multiplied by $q_{s\ell}$. For example, if computing center ℓ prices according to (2a) and (2b) we have: $P_{1j\ell} = a_{j\ell}$, $P_{2j\ell} = b_{j\ell}$, $P_{3j\ell} = c_{j\ell}$, $q_{1\ell} = d_\ell$, $q_{2\ell} = e_\ell$ and $h_{1rj1ik} = f_{ik}$, $h_{2rj\ell ik} = m_{ik}^a f_{ik}$, $h_{3rj\ell ik} = c_{j\ell}$, $g_{1\ell ik} = t_{ik\ell} f_{ik}$, $g_{2\ell ik} = t_{ik\ell} m_{ik} f_{ik}$. Values of the $P_{rj\ell}$ and $q_{s\ell}$ parameters for three different computing centers in the Chicago area are shown in Appendix A.

The model developed in this paper for the determination of the optimal prices $P_{rj\ell}, q_{s\ell}$ has a number of different interpretations and applications which will be discussed later. As described below the overall optimization problem is very complex and we make no claim to solve it completely. We offer only a partial analysis which, provides much useful information concerning optimal user behavior against any set of prices chosen by the computing center. This information includes the optimal disposition of user data sets and programs on the various storage devices available in the short-run on all devices of all systems.

3. The User Problem

In this section we assume that the usage classes, $k \in K$, identify different groups of users who 'own' the corresponding data sets and programs. Further, we assume that a user in class k may be able to derive revenue from data-set $i_k \in I_k$, and that this revenue may depend on the particular device and system on which it is stored (in other words on the time to retrieve the

data-set and the processing power of the associated CPU). This may be the case for example if user k sells the use of his programs or data files to a third party. We let $v_{ijk\ell}$ denote the revenue per time period obtained by a user in class k if data set $i_k \in I_k$ is stored on device $j_\ell \in J_\ell$. If only programs are considered revenue-producing we set $v_{ijk\ell} = 0$ for $i_k \notin I_k^p$. An important special case occurs when all the $v_{ijk\ell}$'s are zero. The user's objective will then be to cost minimize rather than profit maximize. In this case the informational requirements of the model are greatly reduced.

It is assumed that the users have perfect knowledge of the device characteristics and prices offered by the computing centers. They attempt to allocate their data sets to devices in an optimal way. The decision variables for user k are:

$$x_{ijk\ell} = \begin{cases} 1 & \text{if data set } i_k \in I_k \text{ is stored on device } j_\ell \text{ of system } \ell \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik\ell} = \begin{cases} 1 & \text{if program } i_k \in I_k^p \text{ is stored on device } j_\ell \text{ of system } \ell \\ 0 & \text{otherwise} \end{cases}$$

Define the vectors:[1]

$$\xi_k = ((x_{ijk\ell},\ i_k \in I_k,\ j_\ell \in J_\ell),\ (y_{ik\ell},\ i_k \in I_k^p),\ \ell \in L) \tag{3}$$

and

$$\alpha = ((p_{rj\ell},\ r \in R_{j\ell},\ j \in J_\ell),\ (q_{s\ell},\ s \in S_\ell),\ \ell \in L) \tag{4}$$

---

[1] Vectors are defined as row vectors for convenience but are column vectors in mathematical expressions. The order in which the elements appear is the same as in FORTRAN 'Implied DO' statements with the indexes on the left of the list varying most rapidly thus if $J_1 = \{1,2\}$, $J_2 = \{1\}$, $R_{11} = R_{21} = \{1,2\}$, $R_{12} = \{1\}$, $L = \{1,2\}$, $S_1 = \{1,2\}$, $S_2 = \{1,2,3\}$, then $\alpha = (p_{111}, p_{211}, p_{121}, p_{221}, q_{11}, q_{21}, p_{112}, p_{212}, q_{12}, q_{22}, q_{32})'$, where x' denotes the transpose of x.

Given the vector α, of pricing decisions by the L computing centers, user k's decision problem (parametrically) is to choose the allocation, ξ_k, which maximizes profits:

$$(5) \qquad W_k(\alpha) = \max_{\xi_k} \Bigg[\sum_{\ell \in L} \sum_{j \in J_\ell} \sum_{i \in I_k} v_{ijk\ell} x_{ijk\ell} - \sum_{\ell \in L} \sum_{j \in J_L} \sum_{i \in I_k} \left(\sum_{r \in R_\ell} p_{rj\ell} h_{rj\ell ik} \right) x_{ijk\ell} - \sum_{\ell \in L} \sum_{i \in I_k^p} \left(\sum_{s \in S_\ell} q_{s\ell} g_{r\ell ik} \right) y_{ik\ell} \Bigg] .$$

where the first term represents the gross profits resulting from the allocation of data sets, the second the costs associated with data sets, and the third the costs of running the programs.

The constraints are as follows. A requirement that each data set be assigned to exactly one device:

$$(6a) \qquad \sum_{\ell \in L} \sum_{j_\ell \in J_\ell} x_{ijk\ell} = 1 \quad ; \quad \forall \ i_k \in I_k \quad \text{and}$$

$$(6b) \qquad x_{ijk\ell} = 0 \text{ or } 1$$

If a program, $i_k \in I_k^p$ is run on system ℓ, then all the data sets associated with that program must be placed on system ℓ:

$$(7) \qquad \sum_{u \in U_k(i)} \sum_{j_\ell \in J_\ell} x_{vjk\ell} = |U_k(i)| \cdot y_{ik\ell} \quad ; \quad \forall \ i_k \in I_k^p, \ \ell \in L$$

where $U_k(i)$ is the set of data sets used by program i of usage class k and $|U_k(i)|$ denotes the number of data sets in $U_k(i)$.

A requirement that each program be assigned to exactly one system:

$$(8a) \qquad \sum_{\ell \in L} y_{ik\ell} = 1 \quad ; \quad \forall \ i_k \in I_k^p \quad \text{and}$$

$$(8b) \qquad y_{ik\ell} = 0 \text{ or } 1$$

For practical reasons there may be a limit, $M_{jk\ell} \leq M_{j\ell}$, on the space available to a class k user on device j of system ℓ. A value, $M_{jk\ell} \leq M_{j\ell}$, might be imposed by the management of system ℓ or it might represent the maximum amount of space on the device which is desired by the user. Alternatively $M_{jk\ell}$ might be determined by existing contracts for space on device j made between other users and system ℓ. To accomodate this we can add the constraint:

$$(9) \qquad \sum_{i_k \in I_k} m_{ik} x_{ijk\ell} \leq M_{jk\ell} \; ; \; \forall \; j_\ell \in J_\ell, \; \ell \in L.$$

Similarly there may be limits, $T_{k\ell} \leq T_\ell$ on the time available for processing user k's programs on the ℓ^{th} system:

$$(10) \qquad \sum_{i_k \in I_k^p} t_{ik\ell} y_{ik\ell} \leq T_{k\ell} \; ; \quad \ell \in L .$$

In many cases the constraints (9) and (10) will never be binding and may be omitted. This occurs for example with magnetic tape storage which has essentially unlimited capacity and to a certain extent on other devices if they are not operating near full capacity.

The assignment of data sets to different types of device (e.g. disc versus tape) can be controlled either through the $v_{ijk\ell}$ parameters or through explicit constraints. The latter procedure will be necessary if the user problem is viewed as a cost minimization problem (all $v_{ijk\ell}$'s are zero).

For later reference we note that the class k user's problem can be stated in matrix form:

$$(11) \qquad W_k(\eta) = \min_{\xi_k} (\eta' A_k) \xi_k$$

subject to

$$(12) \qquad D_k \xi_k \geq \epsilon_k \; ; \; x_{ijk\ell} = 0 \text{ or } 1, \; y_{ik\ell} = 0 \text{ or } 1 \; ;$$

where $x_{ijk\ell}$, $y_{ik\ell}$ are elements of ξ_k as defined by (3), $\eta = (1 \mid \alpha)$, the form of the matrix A_k is given in Appendix B, and the matrix D_k and vector ϵ_k embody the constants of constraints (6) through (10).

The user problem (5) through (10) has $I_k \cdot J + I_k^p \cdot L$ variables and $I_k + I_k^p \cdot L + I_k^p + J + L$ constraints, where $J = \sum_{\ell \in L} |J_\ell|$. Thus if there are (say) 2 systems with $J_1 = 2$, $J_2 = 3$; user group k has 10 'typical' data sets ($I_k = 10$), 5 of which are programs ($I_k^p = 5$) then the user problem has 60 variables and 32 constraints. Since this is a large integer program it will often be computationally convenient in practice to drop the requirement that the $x_{ijk\ell}$ and $y_{ik\ell}$ are zero-one variables by dropping (6b) and (8b). The constraints (6a) and (8a) automatically ensure the upper bounds:

$$0 \leq x_{ijk\ell} \leq 1 \; ; \; \forall \; i_k \in I_k, \; j_\ell \in J_\ell, \; k \in K, \; \ell \in L$$

$$0 \leq y_{ik\ell} \leq 1 \; ; \; \forall \; i_k \in I_k^p, \; k \in K, \; \ell \in L$$

Note, that from the systems point of view the decision variables now have a valid interpretation in terms of probabilities. Thus $x_{ijk\ell}$ can be interpreted as the probability that one of the N_k users in class k will place a j_k^{th} data-set on device j of system ℓ.

4. The System Problem - Different Cases

In this section, we first formulate the problem faced by a single computing center in attempting to maximize its revenues in the face of competition from other competing computer centers. Much insight for this computationally difficult problem can be obtained by solving two related problems. Thus, we discuss two polar cases which can easily be solved and which will provide information concerning the optimal pricing policy.

For a given allocation $\xi = (\xi_1, \xi_2, \ldots, \xi_K)$ of user data sets to systems and devices the revenue obtained by system $\ell = 1$ (considered to be 'our' system) in a competitive market is:

$$(13)\qquad V^1(\xi) = \sum_{k\in K} N_k \left[\sum_{j_1\in J_1} \sum_{i_k\in I_k} \left(\sum_{r\in R_{j1}} p_{rj1} h_{rj1ik} \right) x_{ijk1} + \sum_{i_k\in I_k^p} \left(\sum_{s\in S_1} q_{r1} g_{r1ik} \right) y_{ik1} \right]$$

$$= \sum_{k\in K} N_k \alpha'_{(1)} A_k^{(1)}\} \text{ where } \alpha_{(1)} = ((P_{rji}, j\in R_{j1}),\ q_{s1},\ S\in S_1) \text{ and}$$

$A_k^{(1)}$ consists of the rows of A_k associated with the price parameters $\alpha_{(1)}$.

We assume that the management of system 1 can estimate the processing and storage requirements of the users and knows the technologies and pricing policies of the competing computer centers. This information is publicly available. Both the technologies and the prices of the competitors are assumed to be fixed in the short run.

The system 1 problem in the competitive case is to choose the non-negative price parameters (p_{rj1}, q_{s1}) to maximize (13) subject to the requirement that the users will profit maximize (or cost minimize if the $v_{ijk\ell}$'s are zero); i.e., that the allocations, ξ_k, are optimal for the respective user problems.

Various other policy constraints might also be imposed. For example, it might be desirable to place lower and upper bounds on the allowable prices:

$$(14)\quad \underline{P}_{rj1} \le p_{rj1} \le \bar{p}_{rj1},\ r \in R_{j1},\ j \in J_1, \text{ and } \underline{q}_{s1} \le q_{s1} \le \bar{q}_{s1},\ s \in S_1$$

where $\underline{P}rj1$, $\underline{q}_{s1}$, $\bar{p}_{rj1}$, $\bar{q}_{s1}$ represent the appropriate bounds. As another example, the management of computer center 1 may make their system attractive to certain users by adding constraints for relevant values of i and k to ensure that the profit associated with the i^{th} program of user k is greatest when that program is run at computer center 1; this is illustrated in the example in Appendix C.

Let the pricing decisions of the competing computer centers be represented by: $\hat{p}_{rj\ell}$, $r \in R_{j\ell}$, $j \in J_\ell$, $\ell \neq 1$ and $\hat{q}_{s\ell}$, $s \in S_\ell$, $\ell \neq 1$. These are assumed to be known constants. However, for convenience in proving certain results we will treat the pricing parameters of the competing computer centers as decision variables and include the constraints:

$$(15)\qquad P_{rj\ell} = \hat{p}_{rj\ell},\ r \in R_{j\ell},\ j \in J_\ell,\ \ell \neq 1 \text{ and } q_{s\ell} = \hat{q}_{s\ell},\ s \in S_\ell,\ \ell \neq 1.$$

Let the 'decision' variables for the system be $\eta = (1,\alpha)$ where α is given by (4). Thus from (11), (12) and (13), we see that computer center 1's problem can be written in the form:

$$(16) \qquad V^1 = \max_{\alpha_{(1)}} \left[\sum_{k \in K} N_k \alpha'_{(1)} A_k^{(1)} \xi_k \right]$$

subject to

$$(17) \qquad \eta' B \leq \gamma$$

and

$$(18) \qquad W_k(\eta) = \min_{\xi_k} [\eta' A_k \xi_k]$$

subject to:

$$\left.\begin{aligned} & D_k \xi_k \geq \epsilon_k \\ & x_{ijk\ell} = 0 \text{ or } 1, \ \forall \ i_k \in I_k, \ j_\ell \in J_\ell, \ k \in K, \ \ell \in L; \\ & y_{ik\ell} = 0 \text{ or } 1, \ \forall \ i_k \in I_k^P, \ k \in K, \ \ell \in L \end{aligned}\right\} \quad \text{for } k \in K$$

where the matrix B and vector γ incorporate the constraints (14) through (15) where appropriate together with the definitional requirement:

$$(19) \qquad \eta_1 = 1$$

Let $R = \sum_{\ell \in L} \sum_{j \in J_\ell} |R_{j\ell}|$ and $S = \sum_{\ell \in L} |S_\ell|$. If only the constraints (14), (15) and (19) are present, B is a $(1+R+S) \times (2+R+S)$ matrix consisting of the identity matrix and one additional column which converts the equality constraints to inequalities.

In addition to the computational feasibility and cost of the problem (16) through (19) which may (as mentioned earlier) necessitate our switching from an integer to a linear programming formulation, there may be difficulties caused by the competition for space and time on the various auxiliary storage devices and CPU's. To solve (16) through (19) exactly we would need to precisely specify

what happens when the demand for a given resource exceeds its capacity; i.e., which data-sets of which users are denied access to the overcrowded device. This seems to be an impossible task with the available integer programming codes, and we therefore change the problem statement slightly in order to obtain a practical and useful model. Note however that the user problem ((11)) and (12)) is valid for properly estimated values of $M_{jk\ell}$ and $T_{k\ell}$ and can be used as it stands.

To avoid the difficult modeling problem associated with the competition among different users for space and time on storage devices and cpu's, we create an artificial 'combined user', by combining all users of all classes together. This is equivalent to considering only a set of K benchmark programs owned by a single user.

We now consider two cases:

A. The 'Combined Case': all systems are viewed to act jointly to maximize their combined revenues by choosing an optimal price vector, α.

B. The 'Single System Case': system $\ell = 1$ chooses an optimal price vector $\alpha_{(1)}$ to maximize its revenues while the prices of the competing computing centers remain fixed.

A. <u>The Combined Case</u>

For given prices, α, the 'combined user' problem is:

$$W(\alpha) = \max_{\xi_1,\xi_2,\ldots,\xi_K} \sum_{k\in K} N_k \Bigg[\sum_{\ell\in L} \sum_{j\in J_\ell} \sum_{i\in I_k} v_{ijk\ell} x_{ijk\ell} - \sum_{\ell\in L} \sum_{j\in J_\ell} \sum_{i\in I_k} \Big(\sum_{r\in R_{j\ell}} p_{rj\ell} h_{rj\ell ik} \Big) x_{ijk\ell} - \sum_{\ell\in L} \sum_{i\in I_k^p} \Big(\sum_{s\in S_\ell} q_{s\ell} g_{s\ell ik} \Big) y_{ik\ell} \Bigg] \tag{20a}$$

subject to:

$$(20b) \qquad \sum_{k\in K} N_k \sum_{i\in I_k} m_{ik} x_{ijk\ell} \leq M_{j\ell} \quad , \quad \forall \ j \in J_\ell, \ \ell \in L$$

$$(20c) \qquad \sum_{k\in K} N_k \sum_{i\in I_k^P} t_{ik\ell} y_{ik\ell} \leq T_\ell \quad , \quad \ell \in L$$

$$(20d) \qquad \sum_{u\in U_k(i)} \sum_{j\in J_\ell} x_{ujk\ell} - |U_k(i)| y_{ik\ell} = 0 \quad , \quad \forall \ i \in I_k^P, \ \ell \in L, \ k \in K$$

$$(20e) \qquad \sum_{\ell\in L} \sum_{j\in J_\ell} x_{ijk\ell} \leq 1 \quad , \quad \forall \ i \in I_k, \ k \in K$$

$$(20f) \qquad \sum_{\ell\in L} y_{ik\ell} \leq 1 \quad , \quad \forall \ i \in I_k^P, \ k \in K$$

Equality holds in (20 e) and (20 f) if the prices offered by the computing centers are attractive enough to justify the storage of datasets and running of programs in one of the centers, $\ell \in L$. Strict inequality implies that the user considers other alternatives outside of the L systems.

Let $A = (A_1 \ A_2 \ldots A_K)$, where the A_k are defined as in (11). Then in matrix notation the 'combined user' problem is:

$$(21) \qquad W(\eta) = \min_{\xi} \eta' A \xi$$

subject to

$$(22) \qquad D\xi \geq \epsilon \ , \ \xi \geq 0$$

where, defining $I = \sum_{k\in K} |I_k|$ and $I^P = \sum_{k\in K} |I_k^P|$, D is $a(J + L + I_L^P + I + I^P + 1) \times (IJ + I_L^P))$ matrix representing the constraints in (20) and including an additional row which allows the equality constraints to be converted to inequalities. Similarly, the vector ϵ represents the right-hand sides of these constraints. Note that the elements of ϵ which correspond to the capacity constraints (20b) and (20c) should contain the values, $M_{j\ell}$ and T_ℓ, of the actual space and time capacities.

We see from (21) and (22) that the combined user attempts to allocate optimally based on the prices announced by the computing centers. The combined systems will attempt to maximize their total revenues by choosing a price vector

$\eta = (1 \mid \alpha)$ (where α is given by (4)) to minimize total user profits. Since policy or other considerations may dictate bounds on the prices for all systems we extend constraint (14) as follows:

(14') $\underline{P}_{rj\ell} \leq P_{rj\ell} \leq \overline{P}_{rj\ell}$, $r \in R_{j\ell}$, $j \in J_\ell$, $\ell \in L$

$\underline{q}_{s\ell} \leq q_{s\ell} \leq \overline{q}_{s\ell}$, $s \in S_\ell$, $\ell \in L$

where $\underline{P}_{rj\ell}$, $\underline{q}_{s\ell}$, $\overline{P}_{rj\ell}$, $\overline{q}_{s\ell}$ are the appropriate lower and upper bounds.

The problem for the combined case can now be written in the form:

(23) $V = \max_{\eta} \min \eta' A\xi$

subject to:

$D\xi \geq \epsilon \quad ; \quad \xi \geq 0$

$\eta' B \leq \alpha \quad ; \quad \eta \geq 0$

where the matrix B and vector α contain the constants of constraints (14') (and any other constraints on the prices which may be appropriate).

One can see that (23) has a feasible solution (viz: $x_{ijk\ell}$, $y_{ik\ell}$ can be zero) and that by virtue of (14') the optimal solution is bounded.

Theorem. The properties just cited guarantee a solution to the following mutually dual linear programs:

(24) $\max_{\nu,\eta} \nu'\epsilon$

subject to:

$-\eta' A + \nu D \leq 0$

$\eta' B \leq \gamma$

$\eta, \nu \geq 0$

(25) $\min_{\mu,\xi} \gamma'\mu$

subject to:

$-A\xi + B\mu \geq 0$

$D\xi \geq \epsilon$

$\xi, \mu \geq 0$

Furthermore let (η^*,ν^*) and (ξ^*,μ^*) be the optimal solutions for (24) and (25) respectively. Then (η^*,ξ^*) is a saddle-point of $\eta' A\xi$ subject to $\eta' B \leq \gamma, \eta \geq 0$ and $D\xi \geq \epsilon$, $\xi \geq 0$. Also: $\nu^{*\prime}\epsilon = \gamma'\mu^* = \eta^{*\prime} A\xi^*$.

Proof: See Kawaguchi and Maruyama, [6], theorems 1 and 2.

In order to solve the combined problem for optimal $p_{rj\ell}$ and $q_{s\ell}$, we can either solve (24) or find the optimal dual solution (η^*,ν^*) to (25). The latter

procedure will be better computationally since (25) has fewer constraints than (24). Note that ν^* contains the shadow prices on the capacity constraints for all the CPU's and auxiliary storage devices. This information can be very useful in that it indicates the most desirable directions for future capacity expansion. Furthermore, if the rates of increase for the N_k (sizes of usage classes) are known, the optimal time sequence of changes in capacity can be estimated.

We note that the constraints in (24) and (25) are predominantly unimodular in character so that most of the $x_{ijk\ell}$'s and $y_{ik\ell}$'s will be 0 or 1 in the optimal solution to the linear program. An efficient heuristic similar to the one in [2] could be developed to obtain, an all (0,1) solution from the linear relaxation solution, if necessary.

The problem (25) has $1 + R + S + J + L + I^P L + I + I^P$ constraints and $IJ + I^P L + 2 + R + S$ variables. Suppose as earlier that $L = 2$, $J_1 = 2$, $J_2 = 3$, and that $K = 5$, with average $I_k = 10$ and $I_k^P = 5$. If the systems have pricing parameter set sizes $R_1 = 2$, $S_1 = 5$, $R_2 = 3$, $S_2 = 5$, then (25) has 149 constraints and 317 variables which is computationally feasible. We can make further computational savings by using the specialized algorithm of Agbadudu [1], or other commercial codes to handle the generalized upper bound constraints (20e) and (20f). Finally, the constraints in (25) when grouped according to the value of k have a block-diagonal structure which lends itself to solution by decomposition.

An illustrative problem is solved in Appendix C for the case where the users cost minimize ($v_{ijk\ell} = 0$).

B. Single System Case

We now return to the problem (16), (17), (18) posed at the beginning of this section with the modification that the user problems (18) are replaced by the combined user problem (20). In addition, the constraints (20 e) and (20 f) are taken to be strict equalities rather than inequalities. The system 1 problem in the 'single center' case can then be written as:

$$(16') \quad V_1 = \max_{\alpha_{(1)}} \left\{ f(\eta) = \sum_{k \in K} N_k \alpha^1_{(1)} A^{(1)}_k \xi_k \right\}$$

subject to:

$$(26) \quad \eta \in \Gamma$$

where Γ is the set of η satisfying (14) and (15)

$$(27) \quad W(\xi) = \min_{\xi \in Z} \{ W(\eta, \xi) = \eta' A \xi \} \geq 0$$

where Z is the set of ξ satisfying (22). Note that $\alpha_{(1)}$ is a sub vector of η.

Theorem 2: If $f(\eta)$ is a concave function of η on Γ, Γ is a convex set and $W(\eta, \xi)$ is concave in η on Γ for every $\xi \in Z$ where Z is a convex set, then the mathematical program given by (16'), (26) and (27) is a convex program.

Proof: Note that $f(\eta)$ being linear in η is a concave function in η; also $W(\eta, \xi)$ due to linearity in η to concave in η or Γ for every $\xi \in Z$. The proof now follows from Bracken and McGill [3].

Since the above program is convex, there exists a set of optimal prices $\alpha_{(1)}$ satisfying (16'), (26) and (27). Bracken and McGill, [3] provide a computational procedure for the general convex programming case using the 'sequential unconstrained minimization technique'. However, unlike their problem, our objective function is bi-linear in η and ξ which allows us to solve for optimal prices using the following algorithm which is similar to the convex simplex method of Zangwill, [13].

An initial starting point is chosen such that the system 1 price parameters are low enough compared to the parameters in (15) to make it optimal for each user program and dataset to be placed on system 1.

Step 1: Problem (24) is solved. As a result, the solution will have the following properties:

1. Assignment of programs and data sets:
   a. If capacity constraints for system 1 are present and binding, some data sets and programs will be assigned by the LP to the other systems in such a way that the combined revenues for all systems are maximized and at the same time the total profitability for the combined users is maximized.
   b. If capacity constraints for system 1 are not present or if they are present and not binding, then all user programs and data sets will be assigned by the LP to system 1.
2. Dual variables for system 1 price constraints.
   Some or all of these dual variables may be positive indicating that the combined revenues for the systems can be increased by relaxing the constraints corresponding to the positive dual variables.

Step 2: Choose the system 1 price (right-hand-side value) with the highest corresponding dual variable and parametrically increase its value until a change in the basis occurs. If for infeasibility reasons, the right-hand-side value can not be increased, choose the next highest dual variable. If none exists, stop.

Step 3: Perform the corresponding pivot. If there are no positive dual variables associated with the system 1 price constraints in the current basis, stop. Otherwise repeat step 2.

This procedure will ensure that system 1's share of the total system revenues as computed by problem (24) is maximized. This is because the iterative procedure only changes system 1 prices while those for the competing systems are held constant by (15). The procedure will have a finite number of iterations since if the system 1 prices are increased indefinitely, all user data sets will be driven to the other systems.

5. Alternative Applications of the Model

In section 3 we identified the usage classes $k \in K$ with various groups of owners of data sets and programs and formulated a model which will allow the k^{th} user group to make optimal decisions concerning the allocation of their computing requirements to different systems and devices. In section 4 we showed how system 1 can determine an optimal short-run pricing policy under the assumption that other competing computer centers do not change their pricing policies. Since the programs and data of all users were combined it was no longer strictly necessary to maintain the identification between usage classes and user groups.

We now enumerate some additional applications for the model developed in Section 4:

1. It is not necessary to simultaneously specify the complete universe of processing demands. The model can be used to analyze prices which would optimize revenues generated from (say) one or two of many possible usage classes--in particular for one or more sets of benchmark programs.

2. As shown by the example in Appendix C, the mini-max model can be used to analyze changes in pricing policy for a given computing center. One common situation where such an analysis is necessary arises when a new computer system is installed or when an old configuration is upgraded. In this case two computing centers can be assumed with $\ell=1$ representing the system under the new pricing policy or hardware configuration and $\ell=2$ representing the same system with the old pricing policy or hard-

ware configuration. The necessary program and data set statistics for each usage class can be collected from past data for the system.

3. Finally, the model can be extended to include the case where computer center 1 wishes to charge CPU usage (and/or usage of other resources) at a differential rate depending on the time of day. Our approach to this peak-load problem is to 'slice' each day into different time periods and to assign a part of the CPU time constraint (10) to each time slice. This will necessitate introducing artificial CPU's associated with different time slices for each center (an additional subscript for the variable $y_{ik\ell}$).

APPENDIX A

Pricing Parameters for Three Computing Centers

(All values are in dollars)

	SYSTEM 1	SYSTEM 2	VOGELBACK[1]
Data Storage Costs, $p_{r\ell}$ (per month)			
Disc Storage	1.00 per 1000 characters	1.21 per 1000 characters	1.80 per 1000 characters
Tape Rental	12.00	6.00	1.00
Program/Job Related Costs $q_{s\ell}$			
CPU + IO Charges	.055 per CPU second	.35 per unit[2]	See footnote 3
Terminal Connect Time	13.00 per hour	9.00 per hour + .25 per 1000 characters	0
Terminal Session Charge	1.00	0	0
Line Printing	.05 per page + .08 per 1000 characters	.06 per page	.04
Card Reading	.005 per card	.005	.005
Card Punching	.02 per card	.02	.02
Magnetic Tape Mount	.25	0	.25

1. Vogelback Computing Center, Northwestern University
2. The number of resource units used by a job is based on the number of CPU seconds and IO processor seconds used.
3. The form of the formula used to compute the processing cost for a job on Vogelback's CDC 6600 computer is given in Appendix C.

APPENDIX B

Structure of the Matrix A_k in (11)

		$\ell = 1$		$\ell = 2$	
	η \ ξ_k	(x_{ijk1})	(y_{ik1})	(x_{ijk2})	(y_{ik2})
$\ell = 1$	1	$(-V_{ijk1})$	0	$(-V_{ijk2})$	0
	(P_{rj1})	(h_{rj1ik})	0	0	0
	(q_{s1})	0	(g_{s1ik})	0	0
$\ell = 2$	(P_{rj2})	0	0	(h_{rj2ik})	0
	(q_{s2})	0	0	0	(g_{s2ik})

APPENDIX C

Illustrative Example

In September, 1977 the Vogelback Computing Center at Northwestern University replaced their CDC 6400 computer by a more powerful CDC6600 model with approximately twice the memory and processing speed. At that time, the pricing formula for running programs on the CDC6400 computer had the form:

$$\text{Job Cost} = q_{1\ell} \cdot t_{ik\ell} + q_{2\ell} \cdot m_{ik}\, t_{ik\ell} + q_{3\ell} \cdot t^{p}_{ik\ell} + q_{4\ell} \cdot m_{ik}\, t^{p}_{ik\ell} + q_{5\ell} \cdot m^{2}_{ik} t^{p}_{ik\ell}$$

where $t_{ik\ell}$ = cpu seconds, $t^{p}_{ik\ell}$ = peripheral processor seconds. The actual prices for the CDC6400 were:

$$q_{12} = 7,\ q_{22} = \frac{3}{140k},\ q_{32} = 1.4,\ q_{42} = \frac{2.0}{140k},\ q_{52} = \frac{.6}{140k^{2}}$$

where the constant 140k represents 140,000 octal.

To analyze this decision in retrospect, we assume (as is actually the case) that management wished to make the cost of running jobs on the CDC6600 cheaper than on the CDC6400. Since changes in storage costs are not involved, these are excluded from the analysis. Letting $\ell = 1$ represent the CDC6600 system and $\ell = 2$ represent the CDC6400 system, the pricing problem can be stated in a form similar to (23):

$$(28) \qquad \max_{q_{11}, q_{q1}, \ldots, q_{51}} \ \min_{\xi} \ \sum_{k\in K} \sum_{\ell\in L} \sum_{i\in I^{p}_{k}} \Big(\sum_{s\in S_{\ell}} q_{s\ell} q_{s\ell ik} \Big) y_{ik\ell}$$

Subject to:

$$(29) \qquad \sum_{\ell \in L} y_{ik\ell} = 1 \quad , \ i \in I^{p}_{k},\ k \in K \ , \ y_{ik\ell} \geq 0$$

$$(30) \qquad \sum_{s=1}^{5} g_{s1ik}\, g_{s1} \leq w_{ik} \sum_{s=1}^{5} g_{s2ik}\, \hat{q}_{s2}, \quad i \in I^{p}_{k},\ k \in K$$

where the summation term on the right-hand-side of (30) is known and the w_{ik} are parameters used for sensitivity analysis. If $w_{ik} \leq 1$ usage class k programs of type i will be cheaper to run on the CDC6600; if $w_{ik} > 1$ they will be more expensive than they were on the CDC6400. These parameters can be used to control the relative share of any cost reductions across both programs and usage classes.

Data was available from actual usage of the CDC6400 and CDC6600 which enabled the estimation of the data shown in Table 1.

Table 1 User Data

Average Number of Jobs per Hour and Core Requirements (1000 octal): August 1977

	University Funds (Educational Uses)		Research Grants + Outside Funds		Center Projects System programs & Development	
	k = 1		k = 2		k = 3	
i_k	f_{i1}	m_{i1}	f_{i2}	m_{i2}	f_{i3}	m_{i3}
1	14.28	20	3.87	20	2.89	20
2	15.39	40	3.10	40	2.27	40
3	9.67	60	2.65	60	1.81	60
4	2.86	100	.77	100	.66	100
5	1.32	120	.42	120	.35	120
6	.35	140	.19	140	.19	140
7	.09	160	.04	160	.08	160

ℓ = 1: CDC6400 - CPU Seconds and PP Seconds: July, 1977

	k = 1		k = 2		k = 3	
i_k	t_{i11}	t^p_{i11}	t_{i21}	t^p_{i21}	t_{i31}	t^p_{i31}
1	3.10	12.72	4.06	16.23	16.92	121.15
2	16.14	23.09	23.70	71.11	28.91	118.52
3	29.17	47.34	43.35	86.70	40.89	96.70
4	42.21	39.13	62.99	94.49	52.86	72.66
5	55.24	48.20	82.64	99.20	64.84	80.97
6	68.28	44.01	102.28	112.51	76.82	73.09
7	81.31	32.04	121.93	85.35	88.80	55.39

ℓ = 2: CDC6600 - CPU Seconds and PP Seconds: August 1977

	k = 1		k = 2		k = 3	
i_k	t_{i12}	t^p_{i12}	t_{i22}	t^p_{i22}	t_{i32}	t^p_{i32}
1	.94	8.31	1.81	18.38	7.82	88.39
2	4.89	15.09	13.25	42.22	15.74	79.56
3	8.84	30.94	24.68	88.86	23.46	89.17
4	12.79	25.58	36.12	75.84	31.28	68.60
5	16.74	31.50	47.55	94.24	39.11	77.51
6	20.69	28.76	58.98	87.89	46.93	69.93
7	24.64	20.94	70.41	68.89	54.75	52.01

A preliminary run indicated that only q_{11} and q_{21} would be positive in an optimal solution of (28), (29) and (30). The additional constraints

(31) $q_{s1} \geq \hat{q}_{s2}, \quad 1 \leq S \leq 5$

were therefore imposed. Let the notation $w._k$ denote that the index i varies over $i \in I_k$. The results for $w._1 = w._2 = w._3 = 1$ are shown in the following table together with the actual prices announced by Vogelback![1]

	Optimal Solution	Announced Prices
Optimal Prices		
q_{11}	11.38	11.8
q_{21}	4.89	4.8
q_{31}	1.65	1.4
q_{41}	2.00	2.0
q_{51}	0.60	0.6
% Reduction in Revenue[2]		
From User Class, k = 1	35.1	37.1
From User Class, k = 2	18.1	19.2
From User Class, k = 3	15.9	16.0
Overall	27.5	29.1

[1]Newsletter, Vogelback Computing Center, July 1977

[2]The percentage change is measured relative to the revenue that would have resulted from the same job mix on the CDC6400.

REFERENCES

1. Agbadudu, Amos, "Generalized Upper Bound and Extensions for Large Scale Systems," unpublished Ph.D. dissertation, Graduate School of Management, Northwestern University.

2. Babad, J. M., Balachandran, V., and Stohr, E. A., "Management of Program Storage in Computers," Management Science, January, 1977.

3. Bracken, J. and J.T. McGill, "Mathematical Programs with Optimization Problems in the Constraints", Operations Research, Vol.21, No. 1, Jan-Feb 1973, pp. 37-44.

4. Cotton, Ira W., "Microeconomics in the Market for Computer Services," ACM Computing Surveys, Vol. 14, No. 3, 1975.

5. Ghanem, S. B., "Computer Center Optimization by a Pricing-Priority Policy," IBM Systems Journal, Vol. 14, No. 3, 1975.

6. Kawaguchi, T. and Maruyama, Y., "A Note on Minimax (Maximin) Programming," Management Science, Vol. 22, No. 6, February 1976.

7. Kleinrock, L., "Optimum Bribing for Queue Position," Operations Research, Vol. 15, No. 2, March-April, 1967.

8. Kriebel, C. H., A. Raviv and H. Zia, "Air Economics Approach to Modeling the Productivity of Information Systems", Technical Report No. NSF APR 75-20546/76/TR2R, Carnegie-Mellon University, Pittsburgh, Pennsylvania

9. Nielsen, Norman R., "Flexible Pricing: An Approach to the Allocation of Computer Resources," AFIPS, Proceedings, Fall Joint Conference, 1968.

10. Nunamaker, J. F. and Whinston, A., "A Planning and Cost Allocation Procedure for Computer System Management," Proceedings of Third Annual SIGCOSIM Symposium, October, 1972, pp. 11-26.

11. Smidt, Seymour, "Flexible Pricing of Computer Services," Management Science, Vol. 14, No. 10, June 1968.

12. Smidt, Seymour, "The Use of Hard and Soft Money and Budgets and Prices to Limit Demand for Centralized Computer Facility, "AFIPS, Proceedings,

13. Zangwill, W.I., "The Convex Simplex Method", Management Science, 16, No. 1, September 1969, pp. 1-13.