SU-SEL-74-037 STAN-CS-74-435

BALANCED COMPUTER SYSTEMS

by

Thomas G. Price

April 1974

Technical Report No. 88

DIGITAL SYSTEMS LABORATORY

Dept. of Electrical Engineering Dept. of Computer Science

Stanford University

Stanford, California

This work was supported by the Joint Services Electronics Program: U.S. Army, U.S. Navy, and U.S. Air Force under contract N-00014-67-A-0112-0044.

ABSTRACT

We use the central server model to extend Buzen's results on balance and bottlenecks. We develop two measures which appear to be useful for evaluating and improving computer system performance. The first measure, called the balance index, is useful for balancing requests to the peripheral processors. The second quantity, called the sensitivity index, indicates which processing rates have the most effect on overall system performance.

We define the capacity of a central server model as the maximum throughput as we vary the peripheral processor probabilities. We show that the reciprocal of the CPU utilization is a convex function of the peripheral processor probabilities and that a necessary and sufficient condition for the peripheral processor probabilities to achieve capacity is that the balance indexes are equal for all peripheral processors. We give a method to calculate capacity using classical optimization techniques.

Finally, we consider the problem of balancing the processing rates of the processors. Two conditions for "balance" are derived. The first condition maximizes our uncertainty about the next state of the system. This condition has several desirable properties concerning throughput, utilizations, overlap, and resistance to changes in job mix. The second condition is based on obtaining the most throughput for a given cost.

TABLE OF CONTENTS

	page
1. Introduction	1
2. Balancing Requests to Peripheral Processors	4
3. Sensitivity	11
4. Designing a Balanced System	13
5. Computation of Balance, Sensitivity, and Capacity	22
6. Applications	27
7. Conclusions	34
Appendix	35
References	50

LIST OF TABLES

		page
1.	An example where the processor with the highest utilization does not have the largest b_i	28
2.	An example with probabilities chosen to achieve capacity.	28
3.	An example where the processor with the highest utilization has the largest b_i	29
4.	Another example where the processor with the highest utilization has the largest b_i	29
5.	An example with probabilities chosen to give equal utilizations	31
6.	Processor utilizations at capacity	33

LIST OF FIGURES

Fig.		page
1.	The central server model	3
2.	An example showing u_0 is neither convex nor concave	8
3.	Maximum throughput holding the geometric mean of the processing rates constant	17
4.	Equivalent central server model	19
5.	Capacity versus the degree of multiprogramming	32

1. Introduction

Measurement is an important tool for finding "bottlenecks" in computer systems. Frequently measurement of a computer system begins by measuring the utilizations of the central processor and I/O channels. If these measurements show that the utilization of one of the processors is substantially larger than the utilizations of the other processors this suggests that the processor with the higher utilization may be a bottleneck and may be degrading the performance of the entire system. The intuitive idea of a bottleneck needs to be made more precise. In order to do this, it is helpful to consider how we can improve the performance of the system once the bottleneck(s) have been identified. In general there may be several ways to improve the performance of the system.

If the processor with the higher utilization is an I/O channel one way to improve performance may be to balance the load on the channels e.g. by shifting some of the load from the busy channel to the other channels. This may be accomplished by moving system data sets to different devices, changing the allocation algorithms for auxiliary storage, moving one or more of the devices on the busy channel to other channels, etc.

Another approach is to try to improve the scheduling of requests at the busy processor in such a way as to reduce the average waiting time at that processor. A third way is to replace some of the processors or devices in the system with faster ones. The first method is basically different from the other two. The first method involves

changing the relative load on the various channels. The other methods are concerned with trying to increase the effective processing rate at a given server.

In this paper we use the central server model [Buzen, 1971A; Buzen, 1971B; Buzen, 1973] to study ways to evaluate and improve overall system performance. In section 2 we consider the problem of balancing requests to the peripheral processors. Sections 3 and 4 are concerned with changing the processing rates to optimize performance. In section 5 we give formulas for computing the measures developed earlier. Finally, in section 6 we give several examples of how the results might be used.

Fig. 1 is a diagram of the central server model. The reader is referred to the papers by Buzen [1971A, 1971B, 1973] for a detailed discussion of the model. The following notation is used in this paper.

μ_i = service rate at the i^{th} server

p_i = probability a job chooses the i^{th} peripheral server

n = degree of multiprogramming

u_i = utilization of the i^{th} server

$L_i(n)$ = average queue length at the i^{th} server

$W_i(n)$ = average waiting time at the i^{th} server (flow time)

$Q_i(n)$ = average queuing time at the i^{th} server (waiting and not being serviced)

T = throughput (jobs/second)

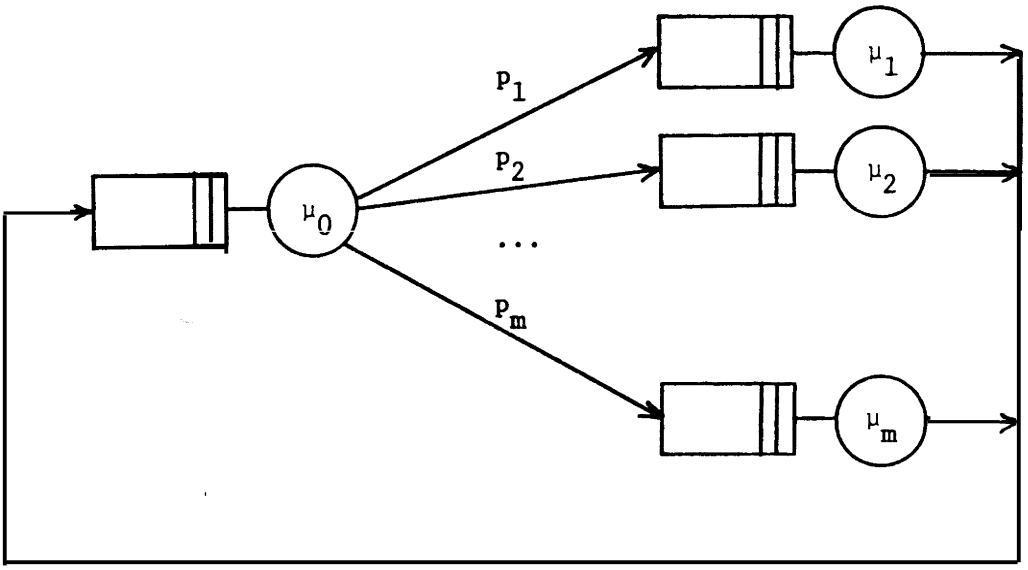

Figure 1. The central server model

2. Balancing Requests to Peripheral Processors

For a given central server model, holding the service rates constant, there is a set of peripheral processor probabilities which maximizes CPU utilization and throughput (jobs complated/unit time). We will call the maximum throughput as the peripheral processor probabilities are varied the capacity of the system. We say that the requests to the peripheral processors are balanced if the peripheral processor probabilities achieve capacity. In this section we develop a quantity which tells us if the load on the peripheral processors is balanced and if not gives us some information about how to change the *probabilities to improve performance.

We can get an intuitive idea of the problem of balancing the requests to the peripheral processors by considering the differential of the reciprocal of CPU utilization with respect to the peripheral processor probabilities. .

$$d(1/u_o) = \frac{\partial(1/u_o)}{\partial p_1} dp_1 + \ldots + \frac{\partial(1/u_o)}{\partial p_m} dp_m \qquad (1)$$

We found it more convenient to work with $1/u_o$ instead of u_o becuase the partials of $1/u_o$ with respect to p_i are positive whereas they are negative for u_o. Suppose $\frac{\partial(1/u_o)}{\partial p_i} > \frac{\partial(1/u_o)}{\partial p_j}$ and suppose we change Pi and p_j slightly so that the new values are $p_i' = pi + Ap$ and $p_j' = p_j - Ap$, while holding the other probabilities constant. Then for small Ap

$$\Delta(1/u_o) \approx \left(\frac{\partial (1/u_o)}{\partial p_i} - \frac{\partial (1/u_o)}{\partial p_j} \right) \Delta p \qquad (2)$$

If Δp is positive the CPU utilization decreases and if Δp is negative the CPU utilization increases. It appears that the quantities $\frac{\partial (1/u_o)}{\partial p_i}$ give an indication of the relative saturation of the peripheral processors. The larger $\frac{\partial (1/u_o)}{\partial p_i}$ the greater the relative saturation of the i^{th} device. We define

$$b_i = \frac{\frac{\partial (1/u_o)}{\partial p_i}}{1/u_o} = - \frac{\frac{\partial u_o}{\partial p_i}}{u_o} = - \frac{\frac{\partial T}{\partial p_i}}{T} \qquad (3)$$

as the balance index for the i^{th} processor. The normalized quantity b_i is the relative change in system performance due to an incremental change in p_i. As (2) shows, transferring a small amount of load from a device with a larger balance index to a device with a smaller balance index improves performance whereas transferring a small amount of load in the reverse direction degrades performance.

In what follows we derive two results which support and strengthen our interpretation of b_i as a balance index. The results are based on the fact that $1/u_o$ is a convex function of the peripheral processor probabilities which we will now prove. $1/u_o$ can be expressed conveniently in terms of the complete symmetric functions $C_n(x)$ (see Appendix).

Let $f_n(x) = \frac{C_n(x)}{C_{n-1}(x)}$ and $g_n(p) = f_n\left(\frac{1}{\mu_0}, \frac{p_1}{\mu_1}, \ldots \frac{p_m}{\mu_m}\right)$. Then $1/u_o = \mu_o g_n(p)$ where n is the degree of multiprogramming. For a derivation, see Buzen [1973] or Muntz and Baskett [1972]. In the following for convenience we drop the subscript n on $f_n(x)$ and $g_n(p)$. The equations are true for any fixed n.

Theorem 1: $1/u_o$ is a convex function of the peripheral processor probabilities.

Proof: From the definition of f(x), $1/u_o = \mu_o g(p)$. Therefore it is sufficient to show that g(p) is convex. Let $p = (p_1, \ldots, p_m)$ and $p' = (p_1', \ldots p_m')$ be two probability vectors and let $\lambda' = 1-h$, $0 < \lambda < 1$. In the Appendix we show that f(x) is convex. Therefore

$$g(\lambda p + \lambda' p') = f\left(\frac{1}{\mu_0}, \frac{\lambda p_1 + \lambda' p_1'}{\mu_1}, \ldots, \frac{\lambda p_m + \lambda' p_m'}{\mu_m}\right)$$

$$= f\left(\lambda\frac{1}{\mu_0} + \lambda'\frac{1}{\mu_0}, \lambda\frac{p_1}{\mu_1} + \lambda'\frac{p_1'}{\mu_1}, \ldots \lambda\frac{p_m}{\mu_m} + \lambda'\frac{p_m'}{\mu_m}\right)$$

$$\leq \lambda f\left(\frac{1}{\mu_o}, \frac{p_1}{\mu_1}, \ldots, \frac{p_m}{\mu_m}\right) + \lambda' f\left(\frac{1}{\mu_o}, \frac{p_1'}{\mu_1}, \ldots \frac{p_m'}{\mu_m}\right)$$

$$= \lambda g(p) + \lambda' g(p')$$

Hence g(p) and $1/u_o$ are convex. It is interesting to note that u_o is neither convex nor concave as can be seen in Fig. 2.

Convex functions have the useful property that an unconstrained local minimum is also a global minimum. A similar result holds when the variables are constrained to be probability vectors.

Theorem 2: The relations

$$b_i = \frac{\frac{\partial g}{\partial p_i}}{g} = \lambda \qquad \text{all } i \text{ such that } p_i > 0 \tag{4}$$

$$b_i = \frac{\frac{\partial g}{\partial p_i}}{g} \geq \lambda \qquad \text{all } i \text{ such that } p_i = 0 \tag{5}$$

are necessary and sufficient conditions on the peripheral processor probabilities to maximize CPU utilization and throughput.

Proof: From Theorem 4.4.1 in **Gallager [1968]** and the fact that g(p) is convex it follows that (4) and (5) are necessary and sufficient to maximize -g(p). Therefore they must be necessary and sufficient to minimize g and to maximize u_o and **T.**

Theorem 2 states that a set of peripheral processor probabilities achieve capacity if and only if the balance indexes for all channels with non-zero probabilities are equal. This result agrees with our intuitive interpretation of the balance index. Furthermore, it is useful if we want to calculate the capacity for we know that we can use any method, however sloppy, and if the method finds a set of probabilities which **satisfy conditions** (4) and (5) then they must achieve capacity.

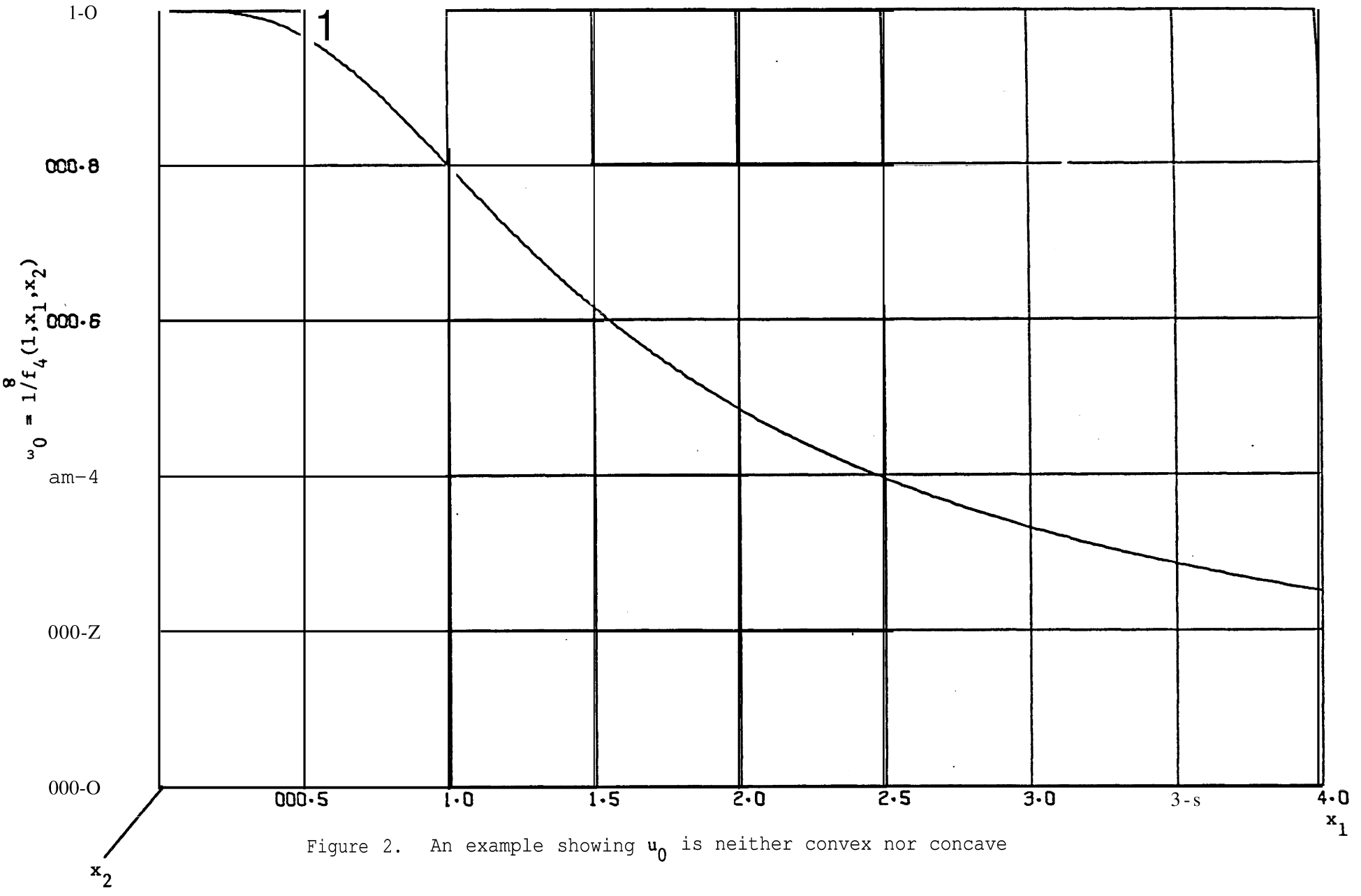

Figure 2. An example showing u_0 is neither convex nor concave

If a set of probabilities does not achieve capacity, the next result gives some information about how to change the probabilities to improve performance. Let $p = (p_1, \cdots p_m)$ and $p' = (p'_1, \cdots p'_m)$ be two probability vectors.

Theorem 3: $g(p') - g(p) > \sum_{j=1}^{m} \frac{\partial g((p)}{\partial p_j} (p'_j - pj)$ and in particular if $\sum_{j=1}^{m} bj\ (P)\ (p'_j - pj) > 0$ then $u_0(p') < u_0(p)$

Proof: From the definition of convexity

$$\lambda g(p') + (1-\lambda)g(p) \geq g[\lambda p' + (1-\lambda)p] \qquad 0 < \lambda < 1$$

Rearranging terms

$$g(p') - g(p) \geq \frac{g[\lambda p' + (1-\lambda)p] - g(p)}{a} \qquad (6)$$

Since (6) is valid for all λ, $0 < \lambda < 1$ we can pass to the limit, obtaining

$$g(p') - g(p) \geq -\frac{dg[\lambda p' + (1-\lambda)p]}{d\lambda}\Bigg|_{\lambda = 0}$$

$$= \sum_{j=1}^{m} \frac{\partial g(p)}{\partial p_j} (P'_j - Pj)$$

Theorem 3 strengthens our interpretation of the balance index. It says that if we transfer load (not necessarily an incremental amount) from a device which is less saturated to a device which is more saturated

(i.e. from one with a smaller balance index to one with a larger index) it will decrease the performance of the system. On the other hand, from (2) we know that if we transfer an incremental amount in the direction larger balance index to smaller balance index it will improve the performance. If we transfer more than an incremental amount in the correct direction we may "overshoot" and the change may either improve or degrade the performance.

To summarize the results of this section, we have developed a balance index which indicates the relative saturation of the peripheral processors; the larger the balance index the greater the relative saturation of the processor. System capacity is achieved when and only when all of the balance indexes are equal. If the probabilities are changed so that the "average" balance index increases then the performance will decrease. If the probabilities are changed an incremental amount so that the average balance decreases then the performance will increase. If the probabilities are changed more than an incremental amount so that the average balance decreases the performance may either improve or decrease.

3. Sensitivity

In the previous section we considered the problem of how to improve system performance by changing the frequency of useage of the different peripheral processors. As mentioned before, other approaches to improving system performance are to buy faster devices or to try to improve the scheduling of one or more servers to reduce average waiting time.[1] The effect of these changes is to increase the processing rate of the server.

If we are considering increasing the processing rate of a server it would be useful to know which processor should receive attention in order to give the most improvement in system performance. Buzen [1971A, 1971B] has proposed $\frac{\partial T}{\partial \mu_i}$ as a measure of the extent to which the i^{th} server is creating a bottleneck. The quantity $\frac{\partial T}{\partial \mu_i}$ is basically a sensitivity measure showing how sensitive system performance is to an incremental change in the processing rate of the i^{th} server. If we intend to improve performance by increasing the processing rate of one of the servers then $\frac{\partial T}{\partial \mu_i}$ should give us an indication of which server will have

[1]Since the central server model assumes exponential service times any scheduling algorithm which does not use information about processing times will have the same average waiting time. Therefore the model is not directly applicable to questions involving scheduling. However, in actual systems the service times will not be exactly exponential and information about latency on rotating storage devices and processing times may be available. In this case a scheduling algorithm such as shortest-latency-time-first may decrease the average waiting time. When considering system performance measures such as utilization or throughput this decrease in average waiting time can be modeled crudely as an increase in processing rate. Thus the idea of sensitivity being developed in this section may suggest where improved scheduling will make the most improvement in system performance.

the most effect. We suggest however that a normalized measure may be more useful.

A normalized measure seems more appropriate because any change we make in a system is most likely going to be relative e.g. we may consider increasing the speed of a server by 20% or 50% etc. An increase in processing rate of 10 operations/second for a processor with $\mu = 10$ is not comparable to an increase of 10 operations/second for a processor with $\mu = 100$. We define

$$s_i = \frac{\frac{\partial T}{\partial \mu_i}}{\frac{T}{\mu_i}} \qquad 0 \leq i \leq m$$

as the sensitivity index for the i^{th} server. si gives the relative change in system performance for a small relative change in μ_i. The sensitivity index is useful if we are considering changing processing rates because we know that system performance is most sensitive to the processing rates of the servers with larger values of s_i.

4. Designing a Balanced System

The word "balance" has another meaning, different from that of section 2, which is often used in connection with computer systems. If a component is very fast relative to the other components of the system increasing its speed further has little effect on system performance. In fact it might be possible to replace it with a component which is considerably slower, and possibly cheaper, without degrading system performance significantly. If possible, a designer would like to choose the speeds of the components so that none are faster than necessary and none are so slow that they significantly degrade system performance. This idea is often expressed by saying that the system should be "balanced." In this section we attempt to make this concept of balance more precise.

A similar problem faces the manager of an installation. He may be able to change the job mix and thereby the processing rates through pricing, by varying the job scheduling algorithm, etc. As much as possible, a manager would like to "match" the job mix to the hardware to maximize throughput. The conditions for balance derived below show how to change the job mix to improve performance.

From the above discussion we see that balancing a system involves varying the processing rates in order to optimize performance. Thus we have an optimization problem similar to the one discussed in section 2. In section 2 we considered the problem of maximizing throughput by varying the peripheral processor probabilities. Similarly one way to approach the problem of balancing a system is to maximize

throughput by changing the μ's. However, there is one fundamental difference in the two problems. As we varied the probabilities they had to satisfy the constraint that $\Sigma\ p_i = 1$. In the case of the μ's there is no obvious constraint. Maximizing the throughput over the μ's without any constraint doesn't make sense because we could always increase throughput by increasing the μ's. Two types of constraints on the μ's which are interesting are the following:

1. Keep the geometric mean of the μ's constant.
2. Keep the total cost of the system constant.

We will consider the problems of maximizing the throughput subject to each of the above constraints.

The geometric mean of the processing rates is

$$\left(\mu_o \mu_1^{p_1} \ \ldots\ \mu_m^{p_m}\right)^{\frac{1}{2}} \tag{7}$$

The set of configurations with the same geometric mean seems to correspond closely to the pertubations we might consider intuitively to determine if a system is balanced. For example if we are trying to decide if a system is balanced we might see if we can increase the throughput of the system by making the CPU say 10% faster and the I/O devices 10% slower. Of course both the original and the modified systems have approximately the same geometric means. This intuitive approach to balance amounts to holding the geometric mean approximately constant and varying the μ's to maximize throughput.

The constraint that we keep the geometric mean constant can be written as

$$\ln \mu_o + p_1 \ln \mu_1 + \ldots + p_m \ln \mu_m = \text{constant} \qquad (8)$$

Lagrange's conditions for a maximum of T subject to (8) are

$$\frac{\partial T}{\partial \mu_i} - \lambda \frac{p_i}{\mu_i} = 0$$

or

$$p_i = \frac{s_i}{\text{constant}} \qquad (9)$$

where p_o is taken to be 1. When (9) holds $s_o = \sum_{i=1}^{m} si$. In section 3 we show that $\sum_{i=o}^{m} s_i = 1$. Therefore the conditions given by (9) yield $s_0 = .5$ and $s_i = .5p_i$, $i \neq 0$. If an actual system has $s_o > .5$ then according to these conditions we will say it is CPU bound; if $s_o < .5$ it is I/O bound. It is interesting to note that the conditions given by (9) are the same conditions derived for balancing the load to the peripherals in section 2.

The conditions given by (9) can also be justified on the basis of sensitivity. The conditions define a "center" point where the system is not particulary sensitive to the processing rate of any

single component. This allows for the most variation in processing rates before some processor becomes saturated. When we design a system we usually have only estimates of the μ's and it seems reasonable to design the system so that a small percentage error in the CPU rate or the I/O rates would have about the same effect on throughput. Similarly, the job mix will probably change over the life of the machine and again it makes sense for the system to have approximately equal sensitivity to changes in either CPU or I/O rates. Fig. 3 is a plot of the maximum throughput as a function of ρ = geometric mean CPU time/geometric mean I/O time, keeping the geometric mean of the processing rates constant. The data displayed in Fig. 3 is for a system with four jobs and three peripheral processors with $p_1 = p_2 = p_3 = .333$. The value of ρ which gives the most throughput is given indirectly by (9). The graph shows that the system has the best resistance to changes in job mix when the conditions given by (9) are satisfied.

Managers are often concerned about making good utilization of the hardware. This suggests maximizing some average of the utilizations. For a given set of p's the conditions above maximize the geometric mean of the utilizations

$$\left(u_0 u_1^{p_1} u_2^{p_2} \cdots u_m^{p_m}\right)^{\frac{1}{2}} \qquad (10)$$

The weighting in (10) may be reasonable in many cases. Suppose that server i has a relatively large p_i and relatively small utilization. This means that the server must be relatively fast. Therefore a

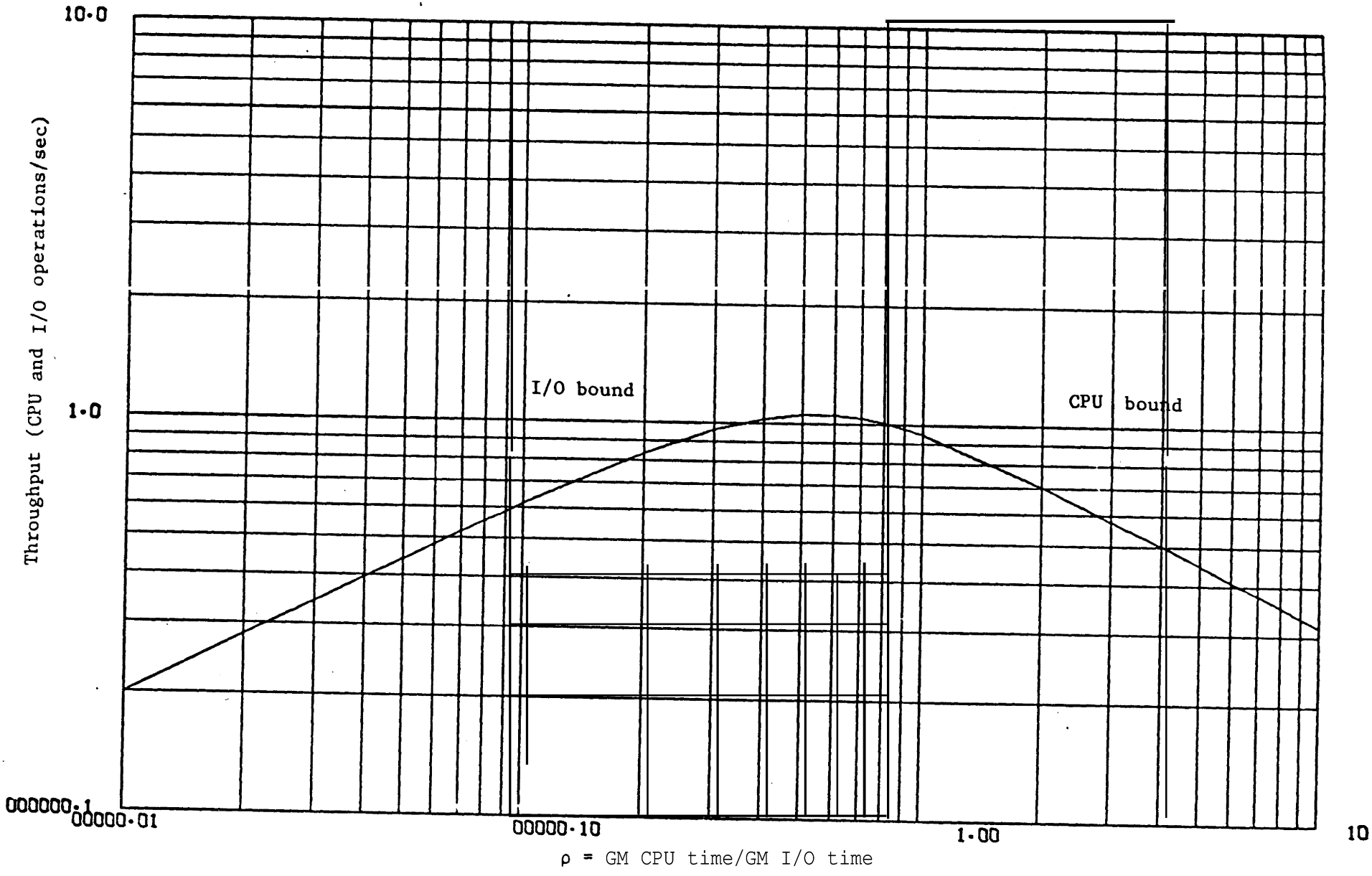

Figure 3, Maximum throughput holding the geometric mean of the processing rates constant (=1)

server with large frequency of useage and low utilization must be a relatively fast server with low utilization, which is probably undesirable. Thus it seems more important to make good utilization of the servers with larger p's.

Finally, we note that the conditions given by (9) maximize our uncertainty about the next state of the system. The average entropy of the service times is given by

$$U = 1 - \frac{1}{2}\left(\ln \mu_o + \sum_{i=1}^{m} p_i \ln \mu_i\right) \qquad (11)$$

The meaning of U requires some explanation. The entropy of a continuous system is really infinite; it takes an infinite amount of information to specify a real number with zero tolerance. However, since any physical measurement is limited in precision to some smallest discernible interval At we can talk about the entropy per discernible interval of the corresponding discrete system. If we measure the values in multiples of the smallest discernible interval (i.e. let At = 1) then the average entropy per operation of the central server model is given by (11) with units of nats per discernible interval [Beckman, 1967; Gallager, 1968]. The average entropy has an interesting interpretation in terms of an equivalent form of the central server model.

It is not difficult to show that the model in Fig. 1 is equivalent to the model in Fig. 4 [Feller, 1966, pp. 54-54], where $r_i > \mu_i$ $0 \leq i \leq m$. In this equivalent model a job is served by an exponential

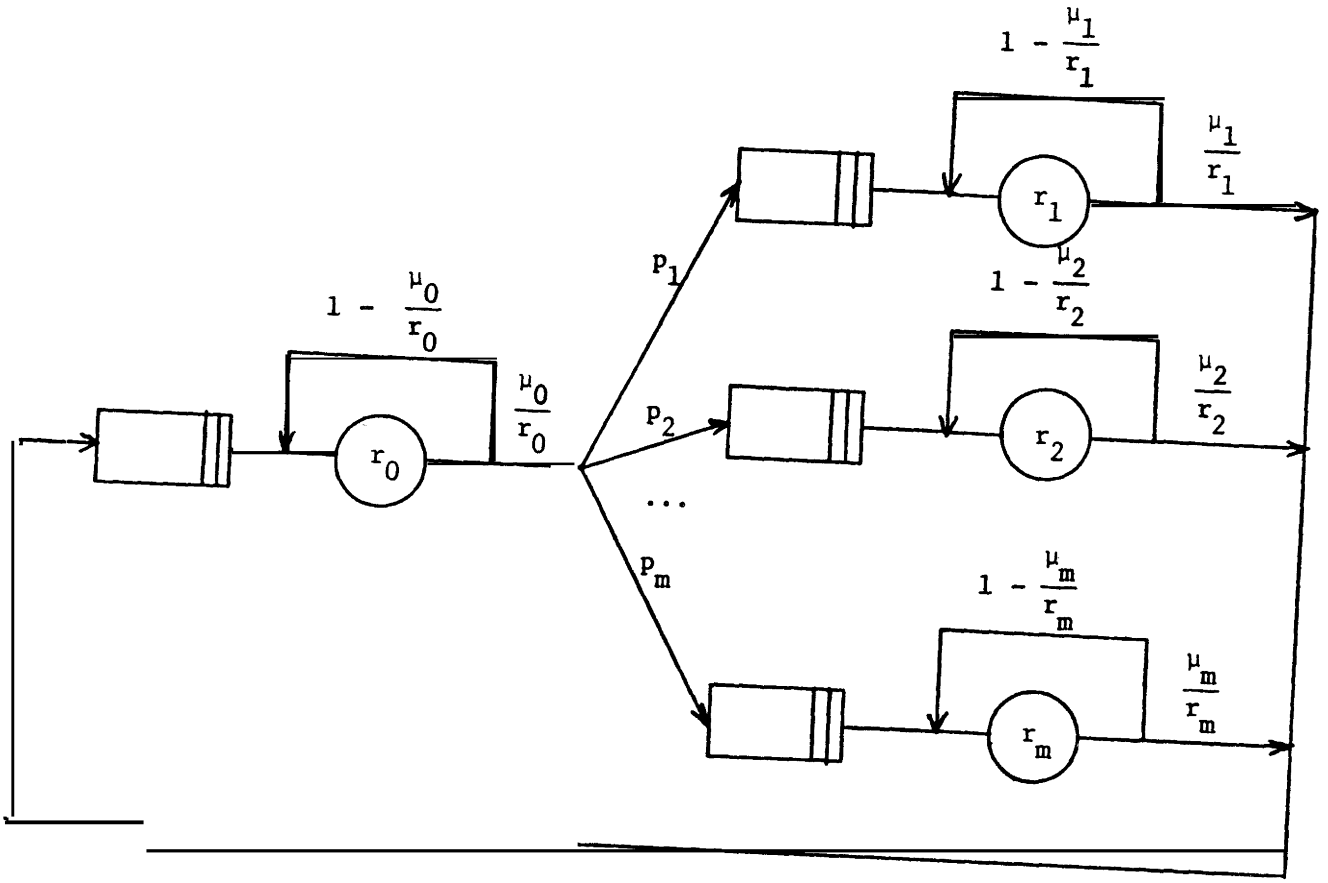

Figure 4. Equivalent central server model

server at a rate r_i. It then chooses with probability $\frac{\mu_i}{r_i}$ to leave that server and with probability $1 - \frac{\mu_i}{r_i}$ to return for more service. The total service time at the i^{th} server is still exponentially distributed and the mean service rate is still μ_i. If we let $r_i = \frac{1}{\Delta t}$ where At is the smallest discernible time and measure all times in units of At then the average amount of information per operation to determine the next state of the system is given by (11). Thus if U is large we are relatively uncertain about what the next state of the system will be whereas if U is small we are relatively certain about what the next state of the system will be. Therefore if the system is balanced we would expect U to be relatively large; if the system is either CPU bound or I/O bound U should be relatively small since there is a high probability that the next state of a job will be at the limiting server. The conditions given by (9) maximize the product of the rate of completion of CPU and I/O operations and U or in other words they maximize our uncertainty about the next state of the system.

A system which satisfies the conditions given by (9) is desirable because it is not sensitive to errors in estimating job characteristics or changes in the job mix. Also, since the conditions maximize the geometric mean of the utilizations they will often yield an economical system. However, if one component is much more expensive than the others then a system which is I/O bound or CPU bound may be more economical. One approach to getting the most throughput per dollar is to hold the system cost constant and vary the μ's to maximize throughput.

Suppose the cost of a processor is proportional to $\mu^{\alpha}, a>0$. Then holding the system cost constant is equivalent to the constraint.

$$c_o\mu_o^{\alpha} + c_1\mu_1^{\alpha} + \ldots + c_m\mu_m^{\alpha} = \text{constant} \qquad (12)$$

where the c's are arbitrary constants. Using Lagrange multipliers the necessary conditions for maximum throughput are

$$\frac{s_i}{c_i\mu_i^{\alpha}} = \text{constant} \qquad (13)$$

5. Computation of Balance, Sensivitity, and Capacity

Balance

In section 2 the balance index was defined as

$$b_i = \frac{\frac{\partial(1/u_o)}{\partial p_i}}{(1/u_o)} = -\frac{\frac{\partial u_o}{\partial p_i}}{u_o} = -\frac{\frac{\partial T}{\partial p_i}}{T}$$

After taking the derivative, the results can be expressed in several equivalent forms each of which may be useful in some applications. Let λ be the rate of completion of CPU and I/O operations $= \mu_0 u_o$. We have $1 \le i \le m$

$$b_i = \frac{L_i(n) - L_i(n-1)}{p_i} \qquad (14a)$$

$$= \frac{L_i(n) - \frac{L_i(n)}{u_i(n)} + 1}{p_i} \qquad (14b)$$

$$= \lambda(n)\left(W_i(n) - \frac{Q_i(n)}{u_i}\right) \qquad (14c)$$

$$= \frac{X(n)}{\mu_i}\left(L_i(n-1) + \right) - \frac{\lambda(n-1)}{\mu_i}\left(L_i(n-2) + 1\right) \qquad (14d)$$

Equation (14b) should be useful in calculating b_i for actual systems since all of the quantities involved (average queue length, utilization and relative frequency) can be measured relatively easily. One might expect that balancing requests to the peripheral processors to obtain the best performance would result in equal average waiting times at each server. Form (14c) is interesting because it shows that this is not true. A similar result has been observed for open networks with multiple servers [Schrage, 1972]. Equation (14d) shows that b_i remains finite as $p_i \to 0$ and may be useful in calculating b_i for small p_i, for example if gradients are needed in an optimizing program.

Sensitivity

Sensitivity is defined in Section 3 as

$$s_i = \frac{\frac{\partial T}{\partial \mu_i}}{\frac{T}{\mu_i}} \qquad 0 \le i \le m$$

Mfferentiation yields

$$s_i = L_i(n) - L_i(n-1) \qquad (15a)$$

Corresponding to the alternate forms for balance we have

$$s_i = L_i(n) - \frac{L_i(n)}{u_i(n)} + 1 \qquad (15b)$$

$$= p_i \lambda(n) \left(W_i(n) - \frac{Q_i(n)}{u_i} \right) \qquad (15c)$$

$$= u,(n) \left(L_i(n-1) + 1 \right) - u_i(n-1) \left(L_i(n-2) + 1 \right) \qquad (15d)$$

where p_o is defined to be 1. Thus $s_i = p_i b_i$ for $1 \leq i \leq m$. It is also interesting to note that

$$\sum_{i=0}^{m} s_i = \frac{1}{T} \sum_{i=0}^{m} \mu_i \frac{\partial T}{\partial \mu_i} = 1$$

This follows from Euler's theorem on homogeneous functions.

Capacity

It is unlikely that simple expressions for the probabilities which achieve capacity exist. However a simple way to compute these probabilities numerically has been found. Finding the capacity is basically the problem of minimizing a convex function of m variables. Many numerical methods for finding the unconstrained local minima of functions of several variables have been studied [Brent, 1973]. However, the algorithms for unconstrained minima cannot be applied directly to find capacity because of the constraints $p_i \geq 0$ and $\Sigma p_i = 1$.

The constraint that $p_i \geq 0$ can be handled by assigning a large positive value to the function whenever the minimization routine asks to evaluate the function at negative arguments. The constraint that $\Sigma p_i = 1$ can be satisfied by minimizing the composite function

$$g(p) + (M(1 - \Sigma p_i))^2 \qquad (16)$$

where M is some large number. The term $(M(1-\Sigma p_i))^2$ is sometimes referred to as a penalty function. If the minimization algorithm finds a minimum of (16) over the region where the components are non-negative then the conditions

$$\frac{\partial g(p)}{\partial p_i} = 2M^2(1 - \Sigma p_i) \qquad Pi > 0$$

$$\frac{\partial g(p)}{\partial p_i} \geq 2M^2(1 - \Sigma\ pi) \qquad Pi = 0$$

must be satisfied. Also, the Σ pi must be approximately 1 because

$$p_i u_o \mu_o 2M^2(1 - \Sigma\ p,) = p_i u_o \mu_o \frac{\partial g(p)}{\partial p_i} = s_i < 1$$

or

$$0 \leq 1 - \Sigma p_i < \frac{1}{p_i u_o \mu_o 2M^2}$$

Therefore, by Theorem 2 the probabilities must achieve capacity. Several linear constraints on subsets of the probabilities can be handled in the same way.

6. Applications

In this section we give several examples of how the definitions and theory of the previous sections might be applied. Consider a system with the parameters shown in Table 1. The system has a CPU and three peripheral processors. The degree of multiprogramming is 3. First consider balancing the load on the peripherals. Inspecting only the utilizations we might try to balance the load on the peripherals by transferring some of the load from server 1 to server 2 or 3. However, from Theorem 3 we know that this would actually decrease performance since $b_3 > b_1$ and $b_2 > b_1$. Also, since the balance indexes are not equal we know that we can improve the performance, for example by increasing p_1 and decreasing p_3 a small amount. Using the method described in Section 5 we calculated the capacity of this system. The probabilities which achieve capacity are shown in Table 2. At capacity the CPU utilization is 57.2%. Thus the maximum increase in CPU utilization that can be obtained by balancing the requests to the peripherals is 5% (a relative improvement of 9.6%). Based on this knowledge we can decide whether or not the potential gain is worth the effort.

Table 1 is an example where the processor with the highest utilization does not have the largest value of b_i (in fact it has the smallest value of bi). In this case we saw that moving load away from the server with the highest utilization will decrease performance. Tables 3 and 4 show examples where the processor with the highest utilization does have the largest value of bi. In these cases moving

Table 1

An example where the processor with highest utilization does not have largest b_i.

i	μ_i	p_i	u_i	b_i	s_i	L_i
0	6.5	-	.522	-	.270	.800
1	4	.667	.566	.469	.313	.895
2	1.5	.2	.452	1.044	.209	.654
3	1	.133	.452	1.566	.208	.654

Table 2

An example with probabilities chosen to achieve capacity.

i	μ_i	p_i	u_i	b_i	s_i	L_i
0	6.5	-	.572	-	.300	.943
1	4	.865	.804	.704	.609	1.608
2	1.5	.109	.271	.704	.077	.344
3	1	.026	.097	.704	.018	.105

Table 3

An example where the processor with the highest utilization has the largest b_i.

i	μ_i	p_i	u_i	b_i	s_i	L_i
0	6.5	-	.358	-	.130	.485
1	4	.333	.194	.151	.050	.228
2	1.5	.333	.517	.753	.251	.801
3	1	.333	.775	1.708	.569	1.485

Table 4

Another example where the processor with highest utilization has the largest b_i.

i	μ_i	p_i	u_i	b_i	s_i	L_i
0	6.5	-	.563	-	.267	.942
1	4	.950	.868	.749	.712	1.902
2	1.5	.030	.073	.348	.010	.078
3	1	.020	.073	.522	.010	.078

load away from the server with the highest utilization can improve performance. Thus one must be careful in interpreting measurements ifonly utilizations are available.

Finally, while constructing the examples for this section we observed that although the probabilities which achieve capacity are often substantially different from the probabilities which give equal utilizations and the resulting queue lengths and utilizations are considerably different (compare Tables 2 and 5), the difference in throughput is frequently relatively small.

Next consider changing the processing rates. For the system shown in Table 2, if we want to improve the scheduling of one of the processors, processor 1 should be considered since s_1 is relatively large.

Looking at theutilizations in Table 1 it is not obvious whether the system is CPU bound, I/O bound, or balanced. The conditions given by (9) tell us that the system is I/O bound ($s_o < .5$). This means simply that changes in the I/O rate have more effect on overall performance than changes in the CPU rate. This suggests that the CPU may be unnecessarily fast and that a slower CPU or faster I/O devices may give more throughput per dollar. Also, the fact that the system is I/O bound shows that throughput can probably be increased by changing the job mix to decrease the I/O time per job and increase the CPU time per job. A pricing schedule which makes I/O time relatively expensive and CPU time relatively cheap might be appropriate.

Table 5

An example with probabilities chosen to give equal utilizations.

i	μ_i	p_i	u_i	b_i	s_i	L_i
0	6.5	-	.500	-	.250	.75
1	4	.615	.500	.406	.250	.75
2	1.5	.231	.500	1.083	.250	.75
3	1	.154	.500	1.625	.250	.75

An important problem which has not been addressed in this paper is the question of how much main memory we should have. To help answer this question we can compute the capacity of the system for several values of n, the degree of multiprogramming, as shown in Fig. 5. The processing rates used in this exampe are $\mu_0 = 6.5$, $\mu_1 = 4.0$, $\mu_2 = 1.5$ and $\mu_3 = 1.0$. Fig. 5 shows that as n is increased the marginal return decreases. The utilization of the processors at capacity for each value of n are shown in Table 6. As n increases the utilizations approach equality [Baskett, 1973]. However, they approach equality fairly slowly.

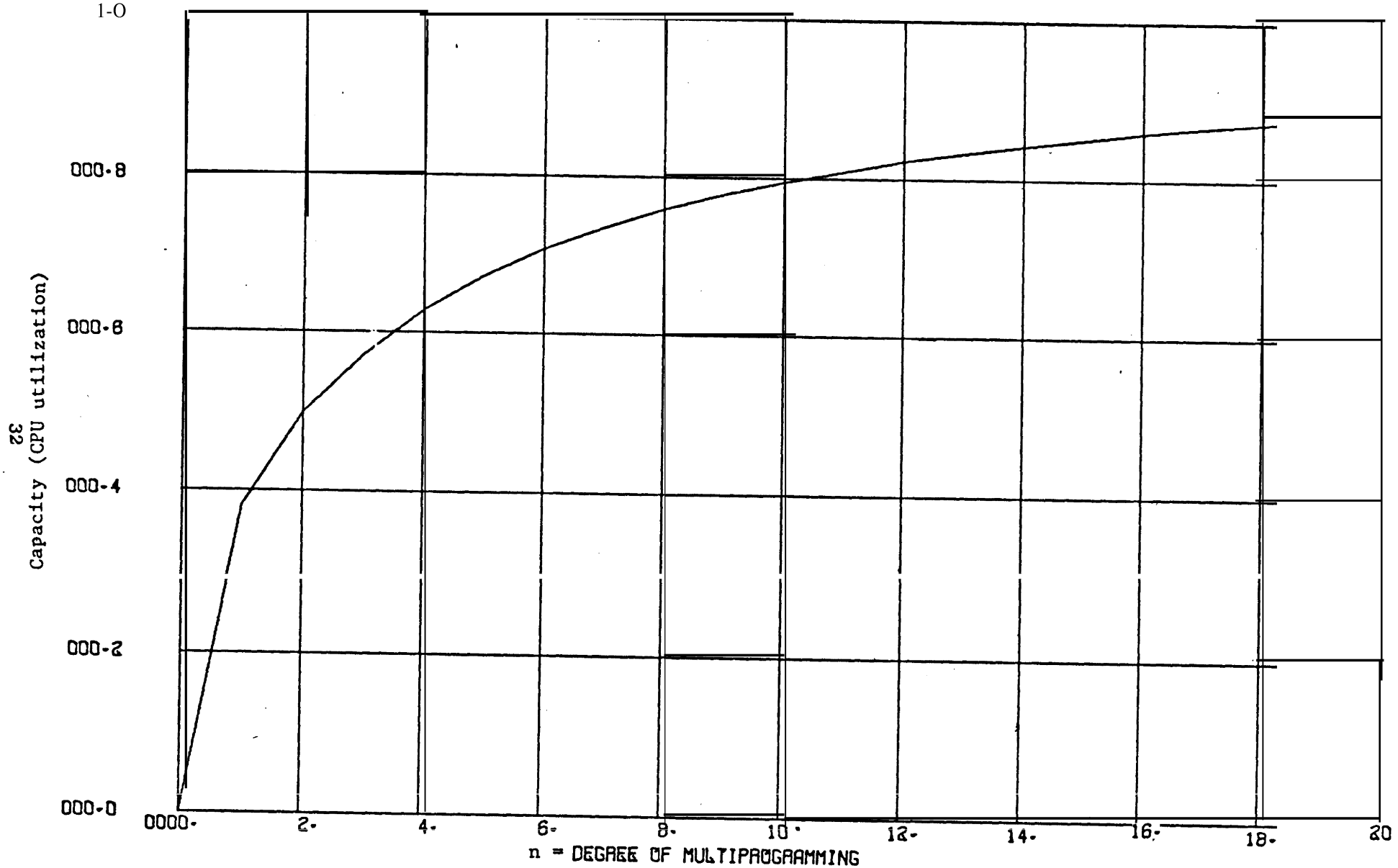

Figure 5. Capacity versus the degree of multiprogramming

Table 6

Processor utilizations at capacity.

n	u_0	u_1	u_2	u_3
1	.381	.619	0	0
2	.499	.767	.114	.000
3	.572	.804	.271	.097
4	.629	.817	.391	.231
5	.673	.832	.476	.332
6	.708	.846	.539	.410
7	.736	.858	.588	.472
8	.760	.869	.628	.522
9	.779	.878	.661	.563
10	.796	.886	.687	.598
11	.810	.894	.710	.626
12	.823	.900	.730	.653
13	.834	.906	.748	.675
14	.843	.911	.764	.693
15	.852	.915	.777	.712
16	.860	.920	.787	.727
17	.867	.923	.799	.741
18	.873	.927	.809	.754
19	.879	.933	.807	.766
20	.884	.933	.826	.775

7. Conclusions

We have used the central server model to develop two quantities, the balance index and sensitivity, which appear to be useful for evaluating and improving computer system performance. The balance index is useful for balancing requests to peripheral processors. Sensitivity indicates which processing rates have the most effect on overall system performance. Many of the results can be extended directly to general exponential networks of queues [Muntz, 1972].

We have shown that the balance index is a measure of the relative saturation of a processor. Specifically, shifting load from a processor with a smaller balance index to a processor with a larger balance index decreases performance. Throughput is maximized when all of the balance indexes are equal and we call the maximum throughput the capacity of the system. Capacity should be a useful concept because it gives a convenient upper bound on the performance of a configuration.

Two conditions for balancing processing rates were derived. The first condition maximizes our uncertainty about the next state of the system. This condition tends to give the most overlap of resource utilization and it shows us how to change the job mix to increase throughput. Also, it may suggest which processors are unnecessarily fast, if any. The second approach, used to derive (13), can be used to determine optimum processing rates based directly on actual cost functions.

APPENDIX

AN INEQUALITY CONCERNING SYMMETRIC FUNCTIONS

Let $a_1,\ldots,a_m$ and $b_1,\ldots,b_m$ be non-negative real numbers. Let $E_n(a)$ denote the n^{th} elementary symmetric function of $a_1,\ldots,a_m$ and let $C_n(a)$ denote the n^{th} complete symmetric function of $a_1,\ldots,a_m$, the formal definitions being

$$E_n(a) = \sum_{\substack{i_1+\ldots+i_m=n \\ i_j=0 \text{ or } 1}} a_1^{i_1} \cdots a_m^{i_m} \qquad (1)$$

$$C_n(a) = \sum_{\substack{i_1+\ldots+i_m=n \\ i_j\geq 0}} a_1^{i_1} \cdots a_m^{i_m} \qquad (2)$$

The following inequalities have been proved by Whitely [1958] and others:

$$E_n^{1/n}(a+b) \geq E_n^{1/n}(a) + E_n^{1/n}(b) \qquad (3)$$

$$C_n^{1/n}(a+b) \leq C_n^{1/n}(a) + C_n^{1/n}(b) \qquad (4)$$

These inequalities are equivalent to the statements that $E_n^{1/n}(a)$ is concave and $C_n^{1/n}(a)$ is convex. Marcus and Lopes [1957] show that

$$\frac{E_n(a+b)}{E_{n-1}(a+b)} \geq \frac{E_n(a)}{E_{n-1}(a)} + \frac{E_n(b)}{E_{n-1}(b)} \qquad (5)$$

which is equivalent to the statement that the ratio $E_n(a)/E_n$, (a) is concave. In this paper we prove the following related inequality.

Theorem:'

$$\frac{C_n(a+b)}{C_{n-1}(a+b)} \leq \frac{C_n(a)}{C_{n-1}(a)} + \frac{C_n(b)}{C_{n-1}(b)} \qquad (6)$$

This inequality is equivalent to the statement that the ratio $C_n(a)/C_{n-1}$ (a) is convex. The proof given in this paper is similar to the proof of (4) by Whitely [1958]. The proof uses the classical theory of maxima and minima, together with induction on m and n.

Let the operator θ, applicable to forms in m variables $a_1,\ldots,a_m$ be defined by

$$\theta = \sum_{j=1}^{m} \frac{\partial}{\partial a_j} \qquad (7)$$

The effect of θ on C,(a) is easily evaluated, as follows.

Lemma 1: [Whitely, 1958]

$$\theta C_n(a) = (m+n-1)\, C_{n-1}(a) \qquad (8)$$

Proof:

$$\frac{\partial}{\partial a_j} C_n(a) = \sum_{i_1+\ldots+i_m=n} i_j a_1^{i_1} \ldots a_j^{i_j-1} \ldots a_m^{i_m}$$

$$= \sum_{i_1+\ldots+i_m=n-1} (i_j+1)\, a_1^{i_1} \ldots a_m^{i_m} \qquad (9)$$

Summation over j from 1 to m gives (8) since $\sum_{j=1}^{m} i_j+1 = n-1-h$

Let

$$f_n(a) = \frac{C_n(a)}{C_{n-1}(a)} \qquad (10)$$

The effect of θ on $f_n(a)$ can be determined using Lemma 1.

Lemma 2:

$$\theta f_n(a) = m+n-1 - (m+n-2)\frac{f_n(a)}{f_{n-1}(a)} \qquad (11)$$

Proof: Differentiating (10)

$$\frac{\partial}{\partial a_j} f_n(a) = \frac{C_{n-1}(a)\frac{\partial}{\partial a_j} C_n(a) - C_n(a)\frac{\partial}{\partial a_j} C_{n-1}(a)}{[C_{n-1}(a)]^2}$$

Summing over j and using Lemma 1 we get

$$\sum_{j=1}^{m} \frac{\partial}{\partial a_j} f_n(a) = m+n-1 - (m+n-2) \frac{C_n(a)}{C_{n-1}(a)} \frac{C_{n-2}(a)}{C_{n-1}(a)}$$

which is equivalent to (11).

We also require a result for the effect of an operator similar to θ when the summation over j in (7) is incomplete. In the following paragraphs we prove several lemmas which lead to the required result Lemma 6.

Lemma 3: [Whitely, 1962]

$$f_n(a) \leq f_{n-1}(a) \tag{12}$$

with equality only if m = 1.

In section 6 of the paper referenced, Whitely shows that $C_n^2/C_{n-1}C_{n+1} \geq 1$ which is equivalent to (12).

Let

$$L_j(n) = \frac{1}{C_n} \sum_{i_1+\ldots+i_m=n} i_j a_1^{i_1} \ldots a_j^{i_j} \ldots a_m^{i_m} \tag{13}$$

In applications concerning queuing networks L_j has the interpretation of average queue length.

Lemma 4:

$$L_j(n) - L_j(n-1) \geq 0, \quad n \geq 1 \tag{14}$$

Proof: The proof is by induction. For n = 1 we have

$$L_j(1) - L_j(0) = \frac{a_j}{\sum_{j=1}^{m} a_j} - 0 \geq 0$$

In general, for $n \geq 1$

$$C\frac{L_j(n)C_n}{a_j} = {}_{n-1}(L_j(n-1)+1) \tag{15}$$

Thus

$$L_j(n) = \frac{C_{n-1}}{C_n} a_j(L_j(n-1)+1) \tag{16}$$

and

$$L_j(n) - L_j(n-1) = a_j\left(\frac{L_j(n-1) + 1}{f_n} - \frac{L.(n-2) + 1}{f_{n-1}}\right) \tag{17}$$

By the inductive hypothesis and Lemma 3 the right hand side of (17) must be non-negative.

Lemma 5:

$$\frac{a}{\partial a_j} f_n(a) \geq 1 - \frac{f_n(a)}{f_{n-1}(a)} \tag{18}$$

Proof:

$$\frac{a}{\partial a_j} f_n = \frac{C_{n-1} \frac{\partial}{\partial a_j} C_n - C_n \frac{\partial}{\partial a_j} C_{n-1}}{C^2_{n-1}} \tag{19}$$

It is not difficult to show that

$$\frac{a}{\partial a_j} C_n = \frac{C_n L_j}{a_j} \tag{20}$$

Substituting (20) in (19)

$$\frac{a}{\partial a_j} f_n = \frac{C_{n-1} C_n L_j(n) - C_n C_{n-1} L_j(n-1)}{a_j C^2_{n-1}}$$

$$= \frac{1}{a_j} f_n (L_j(n) - L_j(n-1)) \tag{21}$$

or

$$\frac{\frac{\partial}{\partial a_j} f_n}{f_n} = \frac{1}{a_j} (L_j(n) - L_j(n-1)) \tag{22}$$

Using (16) in (22)

$$\frac{\frac{\partial}{\partial a_j} f_n}{f_n} = \frac{L_j(n-1) + 1}{f_n} - \frac{L_j(n-2) + 1}{f_{n-1}} \tag{23}$$

Subtracting $\frac{1}{f_n} - \frac{1}{f_{n-1}}$ from both sides of (23)

$$\frac{\frac{\partial}{\partial a_j} f_n}{f_n} - \left(\frac{1}{f_n} - \frac{1}{f_{n-1}}\right) = \frac{L_j(n-1)}{f_n} - \frac{L_j(n-2)}{f_{n-1}} \tag{24}$$

By Lemmas 3 and 4 the right hand side of 24 must be non-negative and we have

$$\frac{\frac{a}{\partial a_j} f_n}{f_n} - \left(\frac{1}{f_n} - \frac{1}{f_{n-1}}\right) \geq 0 \tag{25}$$

which is equivalent to (18)

Lemma 6:

$$\sum_{j=s+1}^{m} \frac{\partial}{\partial a_j} f_n(a) \leq m+n-s-1 - (m+n-s-2) \frac{f_n(a)}{f_{n-1}(a)}, \quad 1 \leq s \leq m-1 \qquad (26)$$

Proof: From Lemma 5

$$\frac{\partial}{\partial a_j} f_n \geq 1 - \frac{f_n}{f_{n-1}} \qquad (27)$$

Summing (27)

$$\sum_{j=1}^{s} \frac{\partial}{\partial a_j} f_n - s\left(1 - \frac{f_n}{f_{n-1}}\right) \geq 0$$

Therefore

$$\theta f_n(a) - \sum_{j=s+1}^{m} \frac{\partial}{\partial a_j} f_n - s\left(1 - \frac{f_n}{f_{n-1}}\right) \geq 0$$

or

$$\sum_{j=s+1}^{m} \frac{\partial}{\partial a_j} f_n \leq \theta f_n(a) - s\left(1 - \frac{f_n}{f_{n-1}}\right),$$

$$= m+n-1-s - (m-h-2-S) \frac{f_n}{f_{n-1}}$$

Proof of the theorem

We use a double induction on m and n. The inequality (6) holds (with equality) for all m if $n = 1$ and for all n if $m = 1$. We shall prove that it holds for a particular pair m,n ($m \geq 2$, $n \geq 2$) provided that it holds for all pairs m',n with $m' < m$ and all pairs m,n' with $n' < n$.

Let $a_1, \ldots, a_m$, $b_1, \ldots, b_m$ be variables which are subject to the conditions

$$f_n(a_1+b_1, \ldots, a_m+b_m) = 1 \tag{28}$$

$$a_1 \geq 0, \ldots, a_m \geq 0, \; b_1 \geq 0, \ldots, b_m \geq 0$$

These conditions define a closed set of points in 2m dimensional space. The set of points is also bounded for if $\frac{C_n(a+b)}{C_{n-1}(a+b)} = 1$ then $\frac{C_n(a+b)}{[C_1(a+b)]^{n-1}} < 1$ and since $[C_1(a)]^n \leq n! C_n(a)$ this implies that $\frac{\frac{1}{n!}[C_1(a+b)]^n}{[C_1(a+b)]^{n-1}} < 1$ or $C_1(a+b) < n!$ Therefore the function

$$f_n(a) + f,(b) \tag{29}$$

has an absolute minimum in the set defined by (28) [Kaplan, 1952].

Let M denote the minimum of (29) subject to the above conditions. It suffices to prove that $M \geq 1$, since this implies (6) by conditions of homogenenity.

Suppose first that the minimum is obtained at a point for which $a_1>0,\ldots,a_m>0$, $b_1>0,\ldots,b_m>0$. This point cannot be a singular point on the surface for by Euler's theorem on homogeneous functions we have

$$\sum_{i=1}^{m} a_i \frac{\partial}{\partial a_i} f_n(a+b) + \sum_{j=1}^{m} b_j \frac{\partial}{\partial b_j} f_n(a+b) = f_n(a+b) = 1 \qquad (30)$$

so that the first partial derivatives cannot all vanish. Hence Lagrange's relations for a local extremal of (29) subject to (28) are applicable. They are:

$$\frac{\partial}{\partial a_i} f_n(a) - \lambda \frac{\partial}{\partial a_i} f_n(a+b) = 0$$

$$\frac{\partial}{\partial b_j} f_n(b) - \lambda \frac{\partial}{\partial b_j} f_n(a+b) = 0$$

where λ is an undetermined multiplier. Hence

$$\frac{\partial}{\partial a_i} f_n(a) = \lambda \frac{\partial}{\partial a_i} f_n(a+b) \qquad (31)$$

$$\frac{\partial}{\partial b_j} f,(b) = \lambda \frac{\partial}{\partial b_j} f_n(a+b) \qquad (32)$$

We combine these relations in two different ways. First we multiply (31) by a_i and (32) by b_j and sum for i and j from 1 to m. By Euler's theorem on homogeneous functions we obtain

$$f,(a) + f_n(b) = \lambda f_n(a+b) \tag{33}$$

By (28) and the definition of M, this implies $\lambda = M$. Secondly we sum over i in (31) and use the result of Lemma 2, in the two forms

$$\sum_{i=1}^{m} \frac{\partial}{\partial a_i} f_n(a) = m+n-1 - (m+n-2) \frac{f_n(a)}{f_{n-1}(a)}$$

$$\sum_{i=1}^{m} \frac{\partial}{\partial a_i} f_n(a+b) = m+n-1 - (m-k-2) \frac{f_n(a+b)}{f_{n-1}(a+b)}$$

We obtain

$$m+n-1 - (m+n-2) \frac{f_n(a)}{f_{n-1}(a)} = \lambda \left(m+n-1 - (m+n-2) \frac{f_n(a+b)}{f_{n-1}(a+b)} \right) \tag{34}$$

Similarly from (32)

$$m+n-1 - (m+n-2) \frac{f_n(b)}{f_{n-1}(b)} = \lambda \left(m+n-1 - (m+n-2) \frac{f_n(a+b)}{f_{n-1}(a+b)} \right) \tag{35}$$

Now suppose A = M < 1. Then (34) yields

$$m+n-1 - (m+n-2)\frac{f_n(a+b)}{f_{n-1}(a+b)} > m+n-1 - (m+n-2)\frac{f_n(a)}{f_{n-1}(a)}$$

Rearranging

$$f_{n-1}(a) \leq \frac{f_n(a) f_{n-1}(a+b)}{f_n(a+b)} \tag{36}$$

Similarly from (35)

$$f_{n-1}(b) \leq \frac{f_n(b) f_{n-1}(a+b)}{f_n(a+b)} \tag{37}$$

Adding (36) and (37)

$$f_{n-1}(a) + f_{n-1}(b) < \frac{f_{n-1}(a+b)(f_n(a) + f_n(b))}{f_n(a+b)} \tag{38}$$

Using (33) in (38)

$$\frac{f_{n-1}(a) + f_{n-1}(b)}{f_{n-1}(a+b)} < \lambda < 1 \tag{39}$$

But by the inductive hypothesis with $n' = n-1$

$$f_{n-1}(a+b) \leq f_{n-1}(a) + f_{n-1}(b)$$

Hence (39) is a contradiction and we must have $\lambda = M \geq 1$.

Suppose next that the minimum of (29) subject to (28) is attained at a point at which one or more of $a_1,\ldots,a_m$, $b_1,\ldots,b_m$ is 0. We can suppose that a_i and b_i are not both 0 for any i, for in that case the result would follow from the inductive hypothesis with $m' < m$. Thus without loss of generality we can suppose the minimum point has

$$a_1 = \ldots = a_q = 0,\ b_{r+1} = \ldots = b_m = 0,$$

where $q \leq r$, and has all the other a_i and b_j positive.

The minimum M is also the minimum of the function

$$f_n(a_{q+1},\ldots,a_m) + f_n(b_1,\ldots,b_r) \tag{40}$$

of m-q+r variables, subject to

$$f_n(b_1,\ldots,b_q, a_{q+1}+b_{q+1},\ldots,a_{r+1},\ldots,a_m) = 1 \tag{41}$$

with all the variables non-negative. These conditions define a closed and bounded set of points in a space of m-q+r dimensions. The minimal point is again a non-singular point on the surface (41), for the relation (30) remains valid if i is summed from q+1 to m and j is summed from 1 to r.

Hence Lagrange's conditions for an extremal of (40) subject to (41) are applicable. They again give (31) and (32) except that i and

j are limited to the ranges $i > q$ and $j \leq r$. The deduction (33) remains valid, on the understanding that a denotes the set $a_{q+1},\ldots,a_m$ and b denotes the set $b_1,\ldots,b_r$. We again have $\lambda = M$.

Summation over i from q+1 to m in (31) gives

$$\sum_{i=q+1}^{m} \frac{\partial}{\partial a_i} f_n(a) = \lambda \sum_{i=q+1}^{m} \frac{\partial}{\partial a_i} f_n(a+b) \qquad (42)$$

where $f_n(a+b)$ now denotes the function on the left of (41). By Lemma 2 applied to $f_n(a_{q+1},\ldots,a_m)$ we have

$$\sum_{i=q+1}^{m} \frac{\partial}{\partial a_i} f_n(a) = m-q+n-1 - (m-q+n-2) \frac{f_n(a)}{f_{n-1}(a)} \qquad (43)$$

By Lemma 6 applied to the function on the left of (41) we have

$$\sum_{i=q+1}^{m} \frac{\partial}{\partial a_i} f_n(a+b) \leq m-q+n-1 - (m-q+n-2) \frac{f_n(a+b)}{f_{n-1}(a+b)} \qquad (44)$$

Combining (42), (43), and (44) we get

$$m-q+n-1 - (m-q+n-2) \frac{f_n(a)}{f_{n-1}(a)} \leq$$

$$\lambda \left(m-q+n-1 - (m-q+n-2) \frac{f_n(a+b)}{f_{n-1}(a+b)} \right) \qquad (45)$$

This corresponds to equation (34) with the constants modified and inequality in place of equality. Similarly the inequality analogous to (35) holds. Proceeding as before we again reach the contradiction (39) and therefore we must have $A = M \geq 1$.

REFERENCES

Baskett, F. and Muntz, R. [1973] Networks of queues. Proceedings of the Seventh Annual Princeton Conference on Information Science and Systems, Princeton University (March 1973).

Beckenbach, E.F. and Bellman, R. [1965] Inequalities. Springer-Verlag, New York, 1965.

Beckmann, P. [1967] Probability in Communications Engineering. Harcourt, Brace, and World, Inc., New York, 1967.

Brent, R. P. [1973] Algorithms for Minimization Without Derivatives. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

Buzen, J. P. [1971A] Queuing network models of multiprogramming. Ph.D. Thesis, Harvard University, Cambridge, Mass., 1971.

Buzen, J. P. [1971B] Analysis of system bottlenecks using a queuing network model. Proceedings of the ACM SIGOPS Workshop on System Performance Evaluation, Harvard University, 1971.

Buzen, J. P. [1973] Computational algorithms for closed queuing networks with exponential servers. C. ACM 16, 9 (September 1973), 527-531.

Courant, R. [1947] Differential and Integral Calculus. Interscience, New York, 1947,

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

Ferdinand, A. E. [1970] A statistical mechanical approach to systems analysis. IBM J. Res. Development, September 1970, 539-547.

Gallager, R. G. [1968] Information Theory and Reliable Communication. Wiley, New York, 1968.

Hardy, G. H., Littlewood, J. E., and Polya G. [1964] Inequalities. Cambridge University Press, 1964.

Kaplan, W. [1952] Advanced Calculus. Addison-Wesley, Reading Massachusetts, 1952.

Marcus M. and Lopes L. [1957] Inequalities for symmetric functions and hermitian matrices. Canadian Journal of Mathematics 9, 2 (May 1957), 305-512.

Muntz, R. and Baskett, F. [1972] Open, closed, and mixed networks of queues with different classes of customers. Technical Report No. 33, Digital Systems Lab, Stanford University, 1972.

Schrage, L. [1972] Unpublished class notes, 1972.

Whitely, J. N. [1958] Some inequalities concerning symmetric forms. Mathematika 5, 9 (June 1958), 49-57.

Whitely, J. N. [1962] A generalization of a theorem of Newton. American Mathematical Society Proceedings 13, 1 (Feb. 1962), 144-151.