Technical Report CS75019-R

AN EVALUATION OF CPU EFFICIENCY UNDER DYNAMIC QUANTUM ALLOCATION[†][*]

U. Narayan Bhat[**]
Department of Industrial Engineering
and Operations Research
Southern Methodist University

and

Richard E. Nance
Department of Computer Science
Virginia Polytechnic Institute and State University

Revised
September, 1977

[†]This report is a major revision containing corrections to and extensions of results given in "Dynamic Quantum Allocation and Swap-Time Variability in Time-Sharing Operating Systems" (VPI & SU Technical Report CS73001-R and SMU Technical Report CP-73009). Early stages of this research were support by NSF Grant GK-19537.

[*]We wish to acknowledge the helpful suggestions and comments of Professor James E. Kalan during the early stages of this research.

[**]Revision supported by the DCEC/ONR Contract No. N0014-75-C-0517, NR 042-324. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

ABSTRACT

A model for a time-sharing operating system is developed in order to assess the effects of dynamic quantum allocation and overhead variability on central processing unit (CPU) efficiency. CPU efficiency is determined by the proportion of time devoted to user-oriented (problem state) tasks within a busy period. Computational results indicate that a dynamic quantum allocation strategy produces significant differences in CPU efficiency compared to a constant quantum. The differences are affected significantly by the variability among allocated quantum values and the demand on the system. Overhead variability also has a pronounced effect. A function that depicts overhead as decreasing with demand produces more stable values of CPU efficiency. The interaction between demand and the amount of overhead is observed to be significant.

Keywords: dynamic quantum allocation, overhead variability, semi-Markov model, time-sharing system, CPU efficiency.

<u>Computing Reviews</u> category: 4.35

INTRODUCTION

Design Considerations in Time-Sharing Operating Systems

With a time-sharing operating system, a single program is given control of the central processing unit (CPU) until execution is completed or a maximum time limit is reached. This time slice, called a quantum, is followed by a period during which the CPU is controlled by the operating system while control of the CPU is removed from one program and assigned to another. This duration is called the overhead.

A quite natural view is that the quantum represents a period of effective processing while the overhead is a necessary but nonproductive time requirement (see Mullery and Driscoll [18]). As Wichman in [13, p. 194] states, "What one wants is some cheap method, in terms of processor time, of controlling what's going on." However, in a relatively heterogeneous environment, characterized by several program priority levels, the scheduling algorithms invoked to assign control of the CPU can become complex and time-consuming. A thorough explanation and summary of scheduling algorithms is presented in the text by Coffman and Denning [9].

The trade-off between increased overhead to make the best possible CPU allocation decisions versus increased processing time for user programs characterizes what we have termed the user and operator perspectives [1]. This paper utilizes a finite-source single-server queueing model to investigate CPU efficiency treating dynamic quantum allocation and overhead variability independently. Future research is intended to consider the interrelationships between the two.

Dynamic Monitoring and Self-Regulation

Most multiprogramming and time-sharing operating systems utilize some form of dynamic adjustment. Adjustment might be accomplished simply by operator intervention, by a priority recalculation such as in the SCOPE operating system on the CDC 6000 series or by the extension of quantum based on program mix such as done by the GEORGE 3 operating system on the ICL 1900 series (see [13, p. 384]). The system monitoring necessary for dynamic adjustment can be based on the analysis of:

(1) individual program characteristics,

(2) characteristics of the program mix, or

(3) measures of the system status in terms of particular summary variables.

Obviously, the level of detail decreases as the monitoring moves from individual programs to system status measures. The quantum allocation decisions modeled herein base the dynamic adjustment on a system status measure -- the number of active user programs.

QUANTUM ALLOCATION AND OVERHEAD VARIABILITY

Quantum allocation

A benefit of allocating quanta of different lengths is that higher priority programs given longer quanta are processed quicker than otherwise, if all other conditions remain identical. This of course benefits only a certain class of users. Adopting the operator's perspective rather than the user's, what benefits are realized from the allocation of variable quantum lengths? Aside from the corequisite to the user benefit, i.e. matching service to need more appropriately, the answer is hopefully an increase in the CPU efficiency. Time-sharing system designers long ago recognized the wasteful overhead incurred when a few (say only two) CPU-dependent programs are seeking service. The accumulated overhead required for allocating the CPU between only two programs represents an inefficiency that, if removed by running each program to completion without interference, can be avoided generally without a noticeable decrease in service. Another argument favoring variable quantum allocation is that a large number of programs in queue can result from certain ones requiring excessive CPU time, and a means for correcting this situation is to provide longer processing periods for each program. This strategy can inhibit the "single user illusion" during a short period, but avoids a gradual deterioration in response time over a longer interval.

Little is known about time-sharing system behavior under dynamic quantum allocation. Coffman [6] proposes two discrete time models that classify jobs as requiring a single quantum or multiple quanta. In the summary of his paper, Coffman [6, p.352] notes the need for generalizing the models "to include an arbitrary 'quantum-function' of the number in the system." Heacox and Purdom [12] extend Coffman's model to allow adjustment of the number of quanta given to a program based on the arrival rates of programs. Also, they assume a constant non-zero swap time; whereas Coffman considers swap time to be negligible.

Chang [4] considers a case where different distributions of quantum length can be adopted under the assumption of a "look-ahead" capability, i.e. if the program is to complete its processing in the next quantum, its service time follows a distribution different from the usual. The assignment of different quanta accomodates short debugging runs as the author mentions [4, p. 122].

Clark and Rourke [5] develop a simulation model to test four quantum allocation algorithms: first-come-first-served, equal-time-slice, proportional to lack of attention (an external priority basis). All four algorithms are based on characteristics of the program mix. The model presented in subsequent paragraphs differs from that of Clark and Rourke [5] in the following ways:

(1) it is an analytical rather than a simulation model,

(2) a system status measure (the number of active user programs) is used to determine the length of quantum,

(3) CPU efficiency rather than CPU utility is the behavioral variable of interest, and

(4) overhead is considered as a controllable variable rather than a constant value.

During the period of review and revision of this manuscript, two related papers have appeared. A model of dynamic quantum allocation by Potier, Gelenbe and Lenfant [21] considers the assignment of additional quanta of fixed length based on the density of arriving jobs. Average response time serves as the behavioral measure that is analyzed as a function of total CPU time, constant overhead times, and various quantum lengths. The second paper, by Blevins and Ramamoorthy [2], outlines a perspective of real-time control, noting the effect of several statistical models on the estimation of CPU and I/O service times. The methodology of a dynamically adaptive operating system is then applied to investigating CPU scheduling functions considering overhead to be negligible relative to the quantum lengths.

Predictive scheduling disciplines, particularly those based on estimation of the second moment of the service time, prove superior.

Overhead variability

Overhead is considered negligible in some models [7,8,15] and a constant value in others [12,17,21]. The amount of time required for switching CPU control from one program to another is quite dependent on the complexity of the algorithms used for scheduling and resource allocation, i.e. assigning disk space, tape drives, peripheral devices, etc. to a particular program. As the operating system attempts allocation of its resources to reduce the average processing time per program, more time is required to determine the proper assignment. But even with less complex resource allocation methods, the overhead incurred for a single transfer of CPU control is partially dependent on the number of programs in the system, i.e. with more programs seeking service, more computations are necessary to determine the program assigned CPU control.

A significant determinant of system overhead is the scheduling algorithm, which includes the quantum allocation algorithm. Bunt and Hume [3] provide a survey description of scheduling algorithms in the context of self-regulating operating systems. They note that parametric scheduling algorithms offer the capability to alternate among several "switching algorithms" with less cost. However, the more detailed the analysis of program mix characteristics, the greater the cost in terms of required CPU time.

A MODEL OF CPU EFFICIENCY WITH VARYING QUANTUM AND OVERHEAD

We begin with the assumption of a single CPU that performs all tasks related to the allocation of resources (in particular, the CPU and main memory). Our objective is to determine the efficiency of the CPU under different conditions for quantum allocation and overhead. We define efficiency as the fraction of time the CPU is actually involved in processing within the total amount of time during which it is involved with any of the job-related activities (i.e. excluding the period when no job is present in the system). Using queue-theoretic terminology, we identify the period in which there is at least one job in the system as a busy period. Such a period is followed by an idle period during which no job demands service from the system. During a busy period if the CPU remains idle while other job-related activities are being performed, this idle time is considered as part of the overhead, but is not part of the idle period defined above.

A busy period followed by an idle period is called a busy cycle. The state of the CPU can be considered as going through a sequence of busy cycles with identical statistical properties. Based on this concept a CPU productivity measure is defined by Gaver [10, p. 424] as

$$\text{CPU Productivity} = \frac{E\ [\text{busy period}]}{E\ [\text{busy cycle}]} \qquad (1)$$

This measure is the same as the utilization measure defined for single-server queueing systems (also see Shedler [23] and Gaver and Shedler [11]) and is also referred to as CPU utilization (see Nance and Bhat [20]).

Since the above productivity measure provides no information on the useful processing time spent by the CPU during a busy period (processing time and overhead are lumped together to form the service time for a task), we define CPU efficiency CPUF as the ratio of the expected amount of useful processing time during a busy period to the expected length of the busy period; i.e.,

$$\begin{aligned} \text{CPUF} &= \frac{\text{E [total useful processing time in a busy period]}}{\text{E [busy period]}} \\ &= 1 - \frac{\text{E [total overhead in a busy period]}}{\text{E [busy period]}} \, . \end{aligned} \tag{2}$$

"Useful processing time" is also referred to as "problem state time" [22, p. 324].

CPU productivity reflects the proportion of a specified time period that the CPU is engaged in some form of processing related to user demand. The CPU efficiency is a measure of the proportion of the processing period devoted to the execution of user programs in contrast with the overhead necessary to maintain the time-sharing environment. While both CPU productivity and CPU efficiency are determined by the match of user and operator (time-sharing system) characteristics, operator decisions with regard to scheduling and resource allocation exert the stronger short-term influence on CPU efficiency.

By defining CPU efficiency to reflect the loss of time due to overhead, we are dividing processing time into two categories: (1) useful CPU time and (2) CPU overhead. In effect we are counting only a portion of the time the system is in supervisor mode [22, p. 313] as contributing to the overhead. This is justified since we wish to investigate the effect of quantum allocation and time-sharing overhead assumptions on the CPU behavior. Time required for tasks unrelated to time-sharing might remain in an entirely different environment. Moreover, this overhead unrelated to the time-sharing environment is quite dependent on the individual program and the mix of programs rather than on decisions related to CPU allocation.

We assume the following conditions:

(1) a time-sharing system using some feedback discipline and to which programs are submitted via an input device,

(2) N input devices potentially can access a single CPU, *i.e.* submit a single program at a time to the CPU,

(3) an input device after being "free" initiates a demand for the CPU after a length of time that has a negative exponential distribution with mean $1/\lambda$,

(4) the processing times of submitted programs are independent, identically distributed negative exponential random variables with mean $1/\mu$,

(5) an overhead computation (D_i) is incurred at the initiation of each period, in which i jobs are in the system,

(6) CPU allocation is not affected by memory availability, and

(7) no explicit considerations are made for the interruption of task execution to accomplish input/output functions.

Now in the most general case the programs are processed in some order for a period of time that does not exceed the quantum, a positive value assumed by the random variable Q_i (where the subscript indicates a possible dependency on the number of programs currently seeking control of the CPU). If the processing of the program is completed within the quantum, it exits from the system; otherwise, the program surrenders control of the CPU to another program and remains to seek a future assignment of the CPU. In either case an overhead period D_i is incurred, with D_i a random variable (again, possibly dependent on the number of programs seeking service). Thus each program requires one or more processing periods $(D_i + Q_i)$, which are called tasks. For the purpose of simplifying the analysis we set $Q_i = q_i$ and $D_i = \delta_i$ with probability 1; i.e., q_i and δ_i are variables, possibly dependent on the number of programs seeking control of CPU. Extension of results for the general case is direct.

Within a busy period, let the completion of a task at t_0 define the origin of observations of the process of task completions. Observations of the task completion process are marked by an ordered pair (t_0,J_0), (t_1,J_1), (t_2,J_2) ... where t_0, t_1, t_2, ... indicate the time epochs, and J_0, J_1, J_2, ... record the number of programs in the system at the respective time points. Let $Z_n(i)$ be a random variable representing the task completion time for the nth task, conditioned on i programs seeking the CPU on completion of the (n-1)st task, i.e. for $J_{n-1} = i$

$$Z_n(i) = t_n - t_{n-1} \qquad (n=1,2,\ldots).$$

Then the distribution of the time devoted by the CPU to the nth task $(b_i(t))$ is given by

$$B_i(t) = \Pr\{Z_n(i) \le t\} = \begin{cases} 0 & t \le \delta_i \\ 1-e^{-\mu(t-\delta_i)} & \delta_i < t < \delta_i + q_i, \\ 1 & \delta_i + q_i \le t. \end{cases} \qquad (3)$$

We denote the Laplace-Stieltjes (LS) transform of $dB_i(t)$ by $\beta_i(\theta)$ where

$$\beta_i(\theta) = \int_0^\infty e^{-\theta t} dB_i(t) \qquad (\mathrm{Re}(\theta)>0).$$

Then

$$\beta_i(\theta) = e^{-\theta\delta_i}[\mu + \theta e^{-(\theta+\mu)q_i}] \; (\theta+\mu)^{-1}$$

and the expected value of $Z_n(i)$ is determined by (4)

$$E\{Z_n(i)\} = -\beta_i'(0) = \eta_i$$

where

$$\eta_i = \delta_i + \mu^{-1}(1-e^{-\mu q_i}). \qquad (5)$$

Before continuing, we note that three special cases can exist for the general form given above for $\beta_i(\theta)$. These cases relate to the assumptions with respect to quantum and overhead:

(1) constant quantum and overhead,

(2) constant overhead, variable quantum, and

(3) constant quantum, variable overhead.

All three cases have physical significance in the modeling of time-sharing systems.

In the first case, the efficiency of the CPU (CPUF) is determined simply as

$$\mathrm{CPUF} = 1 - \frac{\delta}{\eta} = 1 - \frac{\mu\delta}{1+\mu\delta-e^{-\mu q}} \qquad (6)$$

(since the overhead and quantum are both constant neither δ nor η is subscripted). The result in this case is developed in a paper by Kleinrock [16], using an approach based on the expected waiting time of programs and assuming loss of the remaining portion of a quantum after a program's completion. The remaining two cases require a more extensive analysis which we develop using properties of finite Markov chains. Development of the initial relationships below is presented in more detail in previous papers [1,19].

As defined above the ordered pair (t_n,J_n) provides the information on the number of programs in the system just after the task completion epoch t_n. The sequence $\{J_n,Z_n(i)\}$ describes a semi-Markov process with the distribution function for state transitions within the interval $(0,x]$ for this process given by

$$A_{ij}(x) = \Pr\{J_n=j,\ Z_n(i)\leq x | J_{n-1}=i\}.$$

These transition probabilities are determined as

$$dA_{i,i-1}(x) = \begin{cases} \mu e^{-\mu(x-\delta_i)}\ (e^{-\lambda x})^{N-i}dx & \delta_i < x \leq \delta_i + q_i \\ 0 & \delta_i + q_i \leq x \end{cases} \tag{7}$$

$$dA_{ij}(x) = \begin{cases} \mu e^{-\mu(x-\delta_i)} \binom{N-i}{j-i+1} (1-e^{-\lambda x})^{j-i+1} \\ \quad \cdot\, (e^{-\lambda x})^{N-j-1} dx & \delta_i < x < \delta_i + q_i \\ e^{-\mu q_i} \binom{N-i}{j-i} (1-e^{-\lambda(\delta_i+q_i)})^{j-i} \left\{e^{-\lambda(\delta_i+q_i)}\right\}^{N-j} \\ \quad \cdot\, h(\delta_i+q_i-x) \end{cases}$$

$$i=1,2,\ldots, N;\ i \le j \quad (8)$$

where

$$h(z) = \begin{cases} 1 & \text{if } z=0 \\ 0 & \text{otherwise} \end{cases}$$

Let $\alpha_{ij}(\theta)$ be the LS transform of $dA_{ij}(x)$, *i.e.*

$$\alpha_{ij}(\theta) = \int_0^\infty e^{-\theta x} dA_{ij}(x), \qquad \text{Re}(\theta)>0.$$

After some simplifications equations (7) and (8) lead to

$$\alpha_{i,i-1}(\theta) = \frac{\mu e^{-[\theta+\lambda(N-i)]\delta_i}\left[1-e^{-[\theta+\mu+(N-i)\lambda]\,q_i}\right]}{\theta+\mu+(N-i)\lambda} \tag{9}$$

$$\alpha_{ij}(\theta) = \binom{N-i}{j-i+1}\sum_{k=0}^{j-i+1}(-1)^k\binom{j-i+1}{k}\mu e^{-[\theta+(N-j+k-1)\lambda]\delta_i} \cdot \left[\frac{1-e^{-[\theta+\mu+(N-j+k-1)\lambda]q_i}}{\theta+\mu+(N-j+k-1)\lambda}\right]$$

$$+\binom{N-i}{j-i}\sum_{k=0}^{j-i}(-1)^k\binom{j-i}{k} . e^{-[\theta+(N-j+k)\lambda]\delta_i-[\theta+\mu+(N-j+k)\lambda]q_i} \tag{10}$$

Now, consider the process $\{J_n, n=0,1,2,\ldots\}$. It is a finite Markov chain with state space $\{0,1,2,\ldots,N\}$. Its transition probabilities are given by

$$\int_0^\infty dA_{ij}(x) = \alpha_{ij}(0)$$

which we denote as α_{ij}, $i,j=0,1,2,\ldots,N$. Denoting the matrix of transition probabilities as A, we obtain

$$A = \begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} \cdots \alpha_{0,N-1} & \alpha_{0N} \\ \alpha_{10} & \alpha_{11} & \alpha_{12} \cdots \alpha_{1,N-1} & \alpha_{1N} \\ & \alpha_{21} & \alpha_{22} \cdots \alpha_{2,N-1} & \alpha_{2N} \\ & & \ddots \quad \vdots & \vdots \\ 0 & & & \vdots \\ & & \alpha_{N,N-1} & \alpha_{NN} \end{bmatrix}$$

which we partition as below

$$A = \left[\begin{array}{c:cccc} \alpha_{00} & \alpha_{01} & \alpha_{02} & \cdots & \alpha_{0N} \\ \hdashline \alpha_{10} & & & & \\ & & H & & \\ & & & & \end{array}\right] \qquad (11)$$

From the theory of finite Markov chains [14], we know that the (i,j)th element of $(I-H)^{-1}$ represents the expected number of visits of the process $\{J_n\}$ to state j before entering state 0 (which designates the end of the busy period), having started originally from state i. Whenever the process visits state j, it spends an expected length of time η_j (δ_j in overhead and v_j in useful processing) before the subsequent task completion. Thus we have

$$(I-H)^{-1} \; (\underline{n}-\underline{\delta}) = \underline{v} \tag{12}$$

where $\underline{n}$ is the vector of expected task completion times and $\underline{v}$ is the vector af processing times. The elements of the vector $\underline{v}$ are therefore obtained from the relation

$$\underline{n}-\underline{\delta} = (I-H)\underline{v}. \tag{13}$$

Similarly the expected time devoted to overhead $\underline{b}$ is determined by

$$\underline{\delta} = (I - H)\underline{b} \tag{14}$$

Note that the elements of $\underline{v}$ and $\underline{b}$ correspond to the number of programs seeking control of the CPU at the initiation of a busy period. Since a normal busy period starts with one program in the system, the CPU efficiency defined in (2) is obtained as

$$\text{CPUF} = \frac{v_1}{v_1 + b_1} \; . \tag{15}$$

COMPUTATIONAL RESULTS

The experimental procedure

The computational procedure required for the determination of processing times ($\underline{v}$), overhead ($\underline{b}$) and CPU efficiency (CPUF) involves two stages: (1) The determination of the matrix of transition probabilities A (equation (11)) and (2) the solution of the linear system of equations (see (13) and (14)). The transition probabilities are obtained from expressions given in equations (9) and (10), where the elements $\alpha_{i,i-1}$ are shown separately from all the remaining ones. Although a general expression can be given for any element of A, we achieve computational savings by avoiding the unnecessary terms in (10) for the subdiagonal elements.

The determination of the transition probabilities also involves a combinatorial calculation routine. For small values of N, high precision might not be necessary. However, the routine utilizes double precision arithmetic so as to increase the range of values for N.

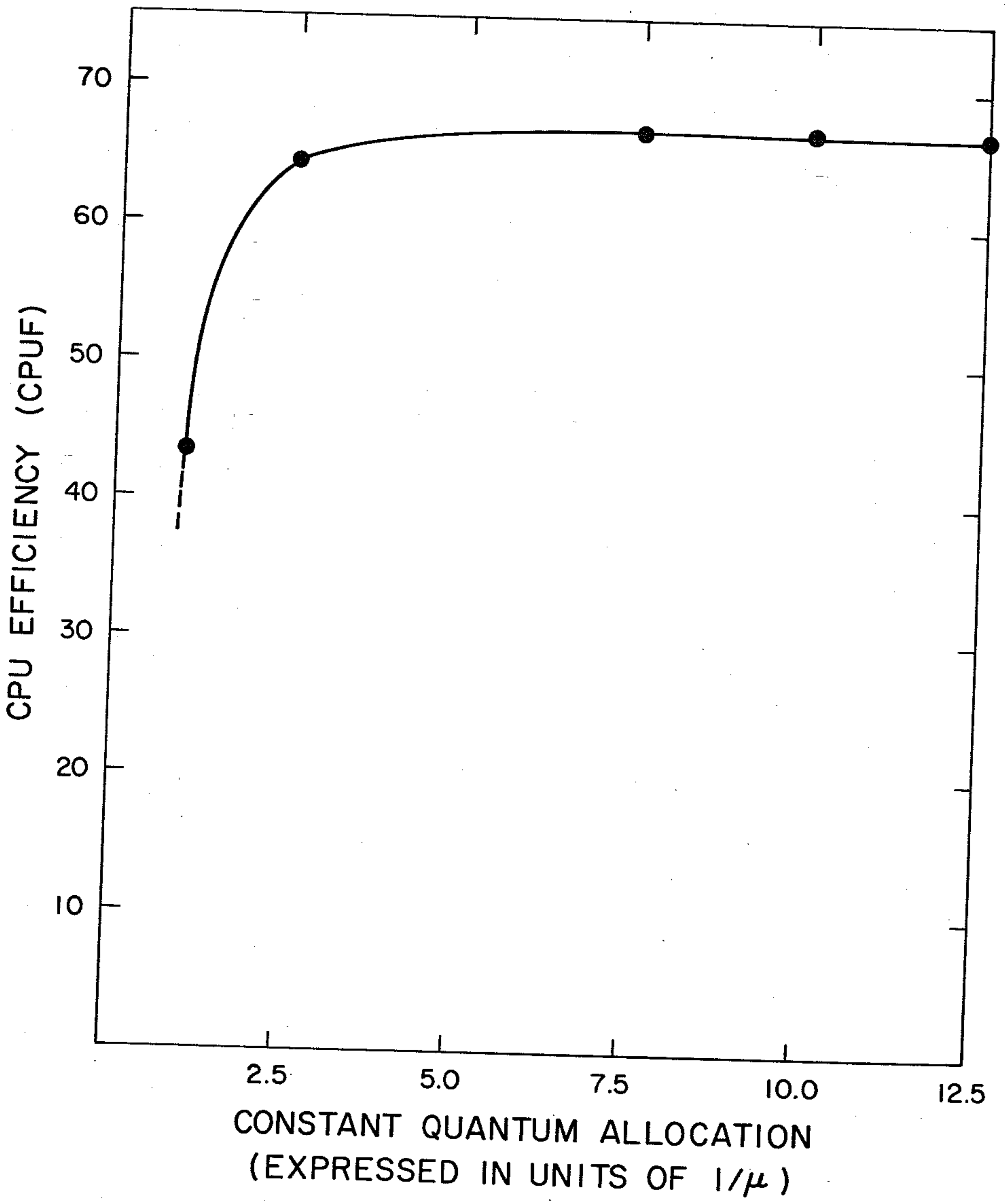
The linear system of equations is solved using a modified Gauss-Seidel routine programmed by Professor James E. Kalan. Because of the Hessenberg structure of the matrix A, this routine proves to be extremely efficient.

For computational comparisons the mean processing time $1/\mu$ is assumed to be of unit length . Consequently, all time values are expressed in terms of $1/\mu$. All programs are coded in FORTRAN. Results are based on execution of the programs on a UNIVAC 1108 and an IBM S/370 dual 158 system.

Dynamic quantum allocation

Prior to observing the effect of dynamically varying the quantum, we can note in Figure 1 the general behavior of CPUF as the constant quantum length is increased. By increasing the quantum from .5 to 2.5 (in units of the mean processing time $(1/\mu)$), an increase of approximately 20 percent efficiency is observed. However, by increasing the quantum from 2.5 to 12.5, a gain of only two percent is realized. Obviously, little advantage in CPUF is obtained by setting the quantum length beyond 2.5.

Figure 1. Effect of Constant Quantum with Constant Overhead (δ = .5, $1/\lambda=10/\mu$,N=10)



To enable a comparison of CPUF under different strategies for dynamic quantum allocation, we define a measure of strategy variability - dispersion (d), where

$$d = \frac{1}{\bar{q}} \sqrt{\frac{\sum_{i=1}^{N} (q_i - \bar{q})^2}{N}} \qquad \left(\text{and } \bar{q} = \frac{1}{N} \sum_{i=1}^{N} q_i\right).$$

The analogy between dispersion and the coefficient of variation is recognized; however, the quantum values (q_i) are not realizations of a random variable. We also refer to quantum allocation strategies as increasing, decreasing, and static (a constant quantum length regardless of the number of programs demanding CPU service).

In the following examples we explore two types of allocation strategies: (1) increasing the quantum length as the number of programs demanding service increases and (2) decreasing the quantum length as the number of programs demanding service increases. An allocation strategy is characterized by S(N,d) where N is the number of peripheral units and d is the measure of dispersion. The four "increasing" strategies are listed below. Note that the strategy average ($\bar{q}$) is 1.50 for all strategies.

S(5,.408) = (.70, .95, 1.50, 2.05, 2.30)
S(5,.739)= (.10, .45, 1.50, 2.55, 2.90)
S(10,.408)= (.50, .75, 1.00, 1.25, 1.50, 1.75, 2.00, 2.25, 2.50)
S(10,.739)= (.50, .50, .75, .75, 1.00, 1.25, 1.25, 2.00, 3.00, 4.00)

Corresponding to each of the above is a "decreasing" strategy that we designate as $\tilde{S}(N,d)$, in which the order of quantum values is reversed. For each strategy pair {S(N,d) and $\tilde{S}$(N,d)}, we obtain CPUF values for the combinations of parameter values shown in Table 1.

Demand Rate(λ)	Overhead			
	δ = .25		δ = .50	
	N=5	N=10	N=5	N=10
1/20				
1/10				
3/20				

Table 1. The Combination of Parameter Values for Each Strategy Tested (Values for Overhead and Interarrival Time in Units of $1/\mu$).

Figure 2 is comprised of four parts to show the effects of demand, overhead, and number of peripheral units on CPUF. To simplify the labelling of strategies in Figure 2, single letters are used to indicate strategies as follows:

A: constant quantum (static strategy),

D: S(N,.408) or $\tilde{D}$: $\tilde{S}$(N,.408), and

E: S(N,.739) or $\tilde{E}$: $\tilde{S}$(N.,739).

The following observations can be made:

1. As demand increases the general trend is for CPUF to increase under increasing quantum strategies (S(N,d)) and to decrease under decreasing

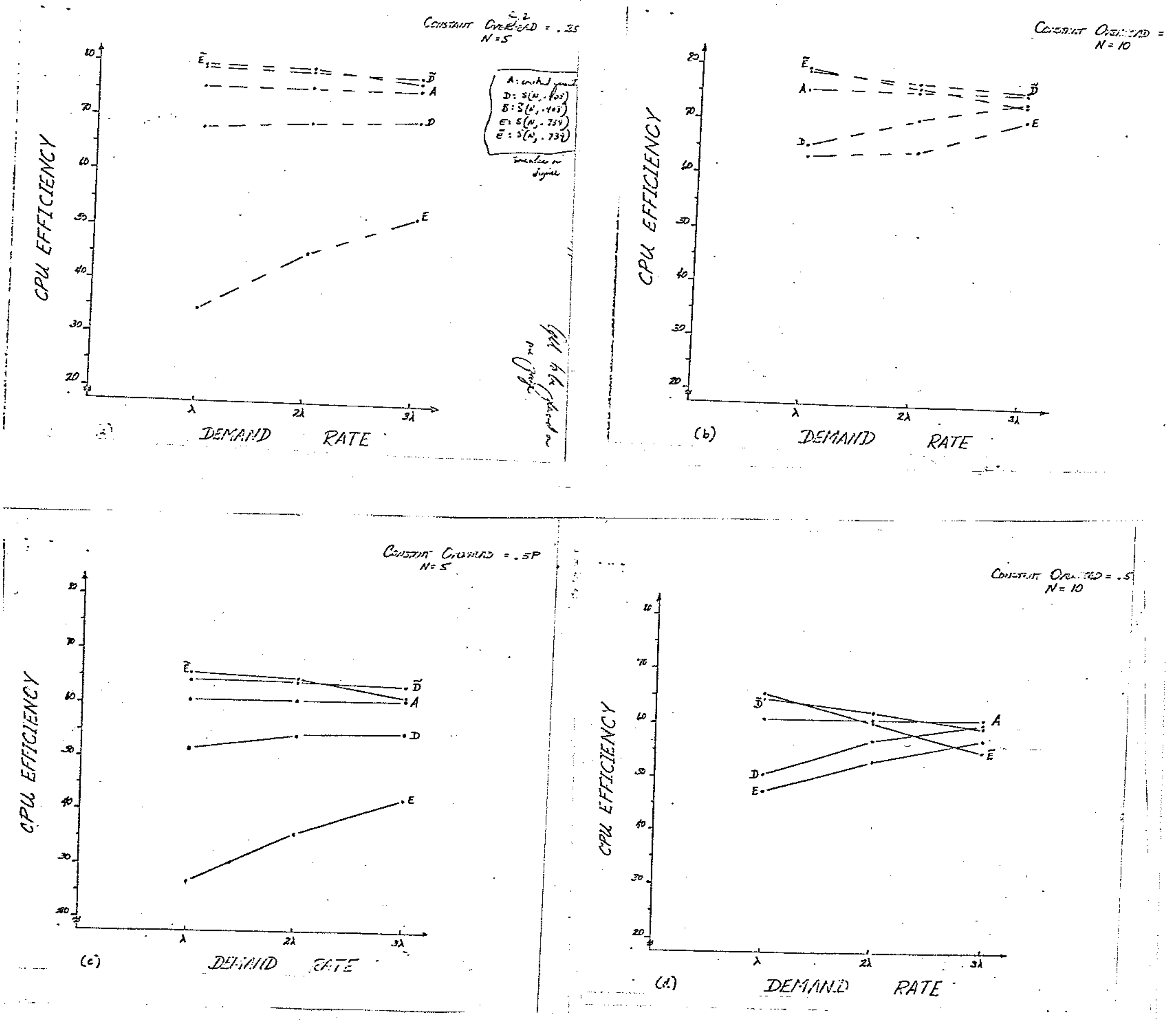
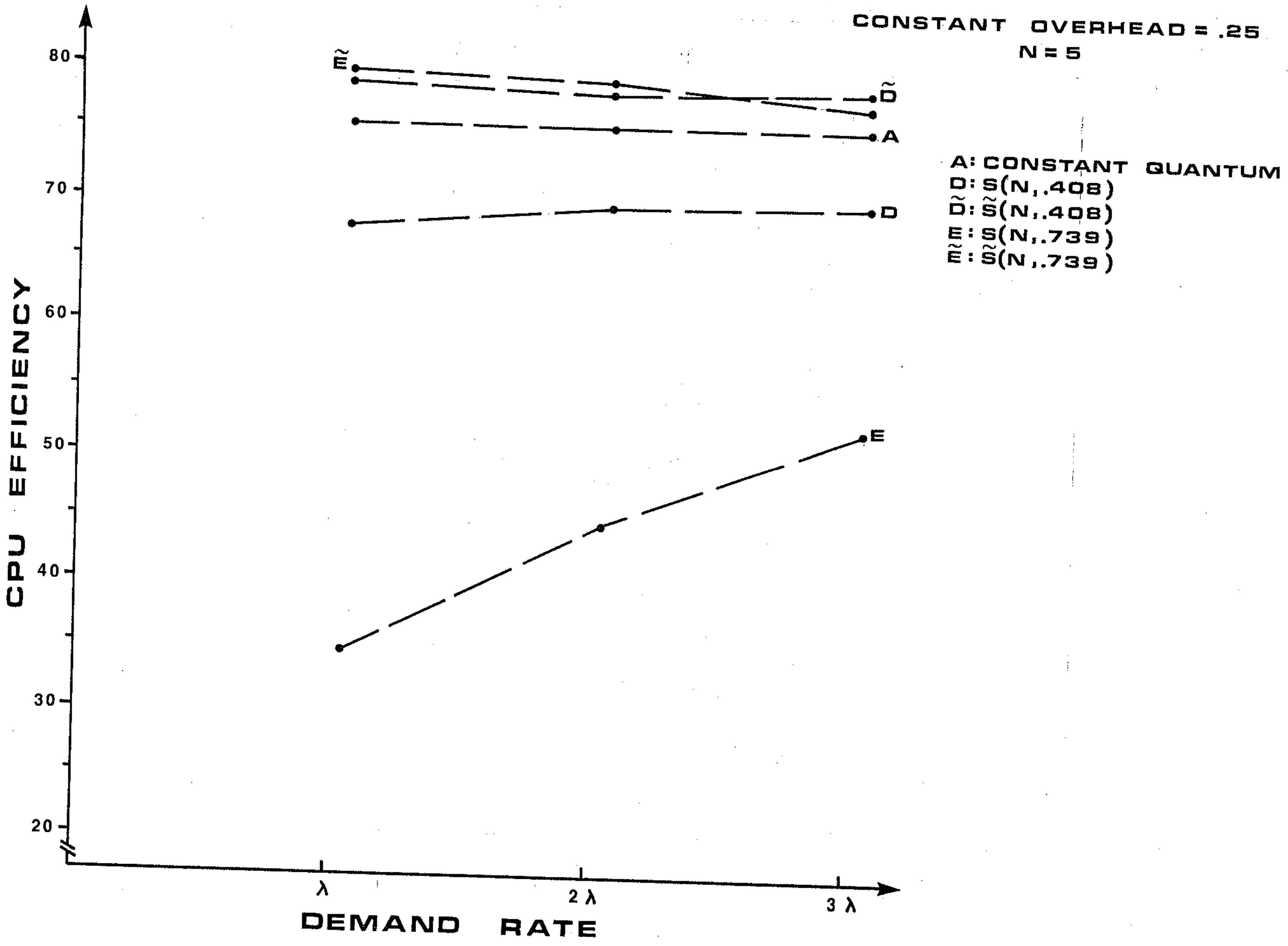


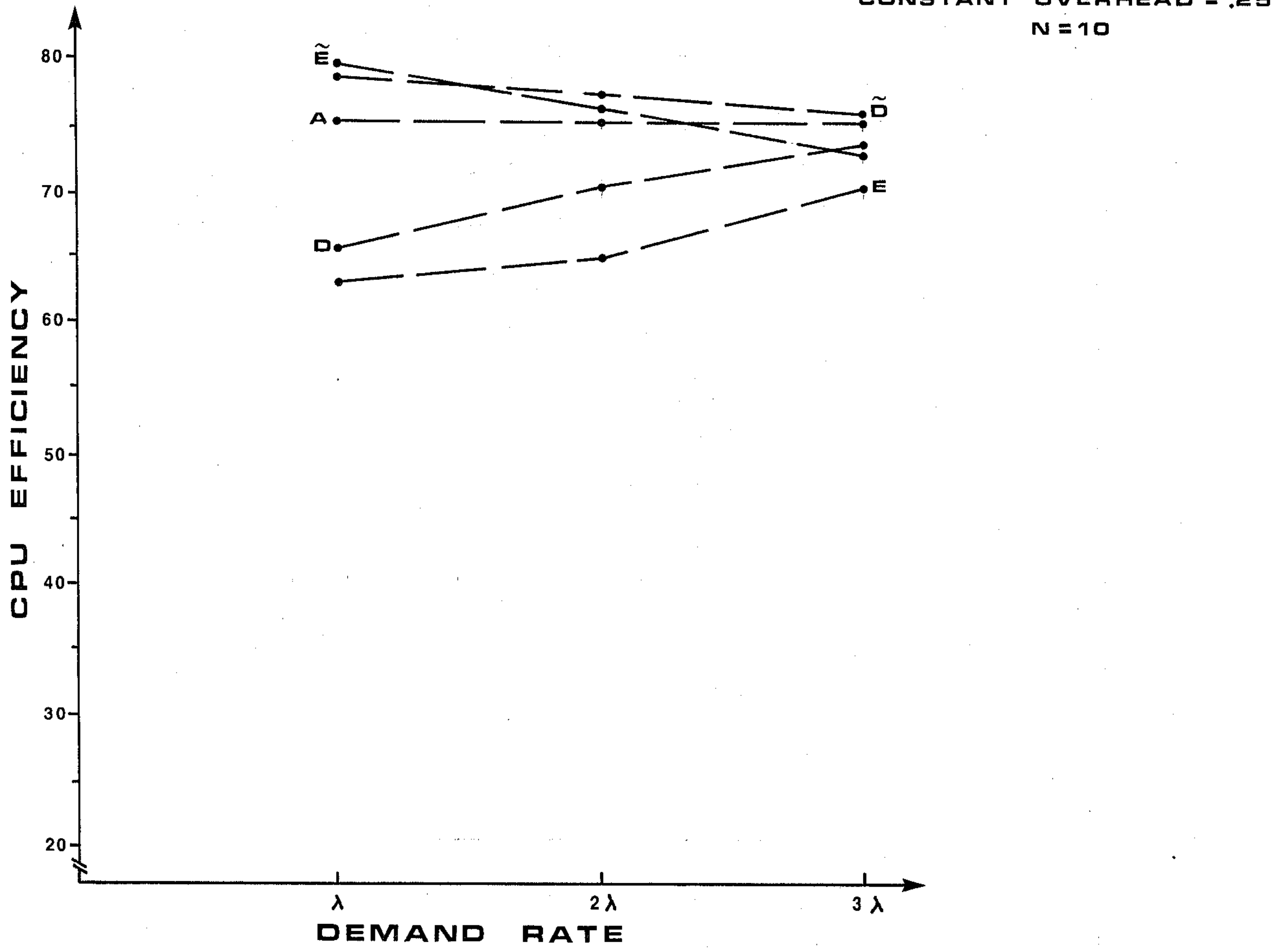
Figure 2. The Effect of Demand Rate on CPU Efficiency (CPUF) Under Quantum Allocation Strategies with Varying Dispersion Values (d)

(Overhead and Demand Rates Expressed in Units of 1/μ.)

Figure 2. The Effect of Demand Rate on CPU Efficiency (CPUF) Under Quantum Allocation Strategies with Varying Dispersion Values (d)

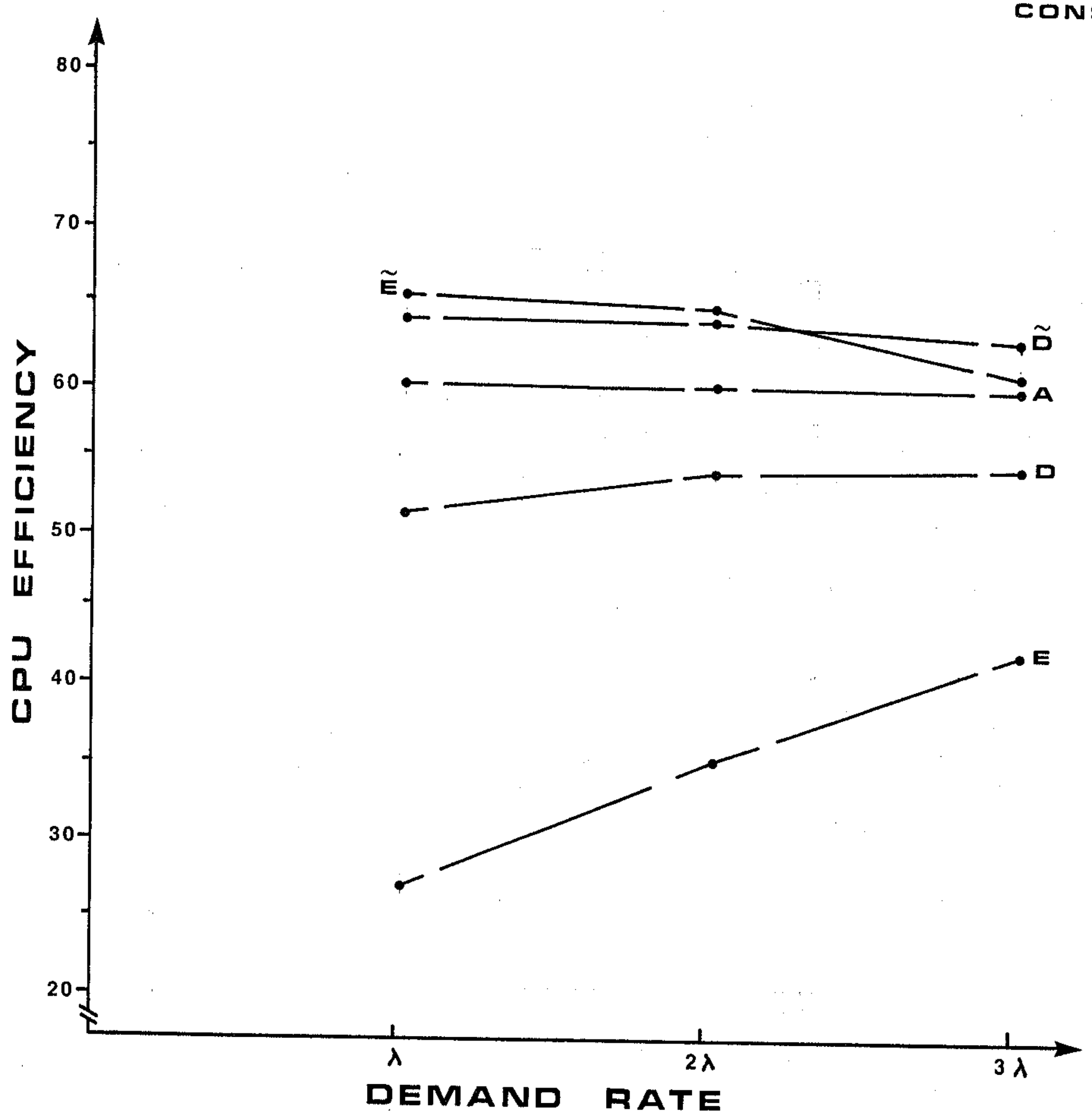
(Overhead and Demand Rates Expressed in Units of $1/\mu$)





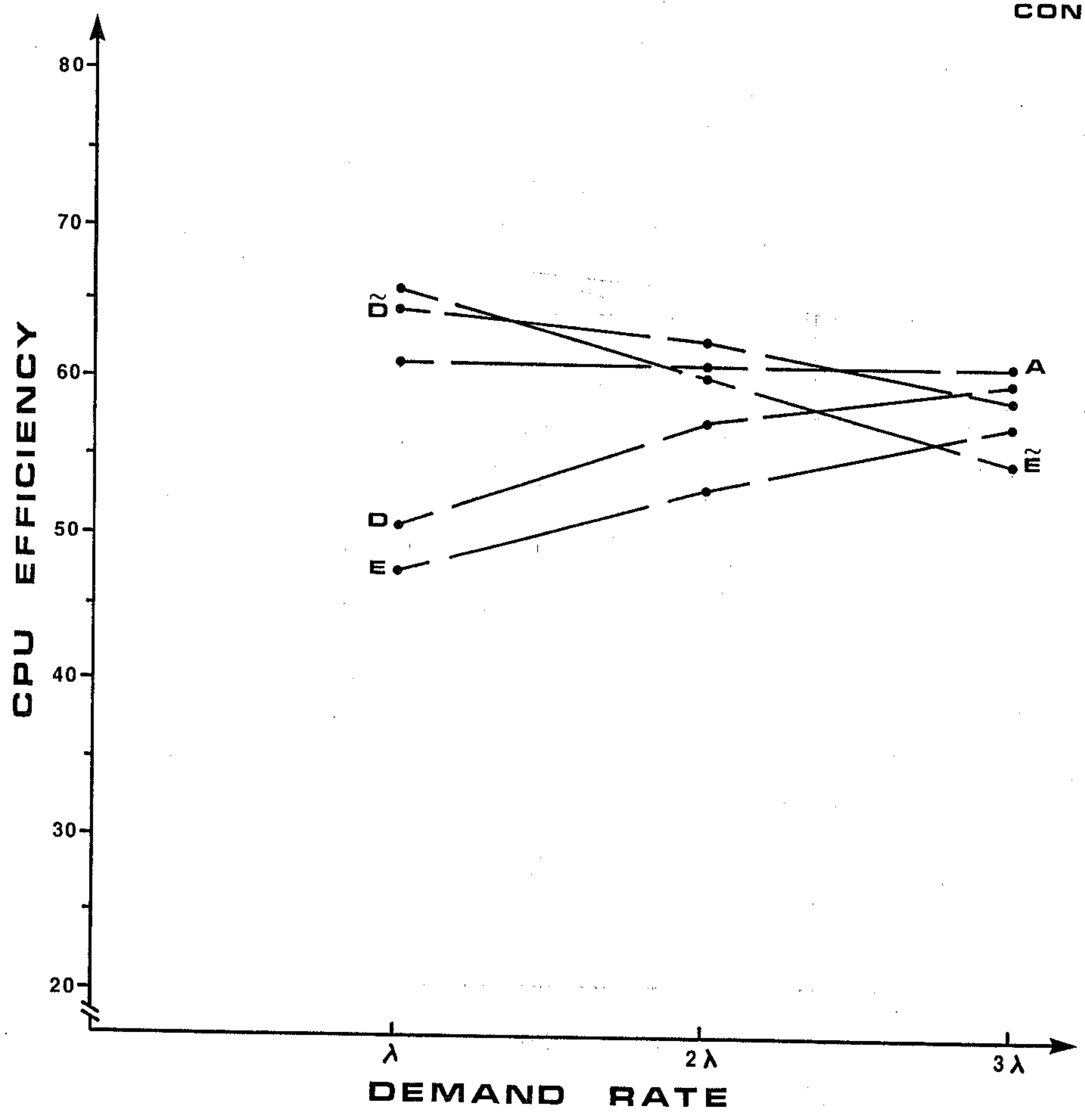
CONSTANT OVERHEAD = .25
N = 10
CPU EFFICIENCY
20
30
40
50
60
70
80
$\tilde{E}$
A
D
$\tilde{D}$
E
λ
2 λ
3 λ
DEMAND RATE

CONSTANT OVERHEAD = .5
N = 5



CONSTANT OVERHEAD = .5

N = 10



quantum strategies ($\tilde{S}(N,d)$).

2. The difference in CPUF values for corresponding increasing and decreasing quantum strategies is greater when the dispersion is higher. (Note the relative differences in D and E and $\tilde{D}$ and $\tilde{E}$ in each part of Figure 2.)
3. As demand increases, higher dispersion has a more pronounced effect on CPUF ander an increasing quantum strategy ($S(N,d)$). (Compare D and E versus $\tilde{D}$ and $\tilde{E}$.)

The significance of these results is not to be derived by comparing values of CPUF under dynamic quantum allocation strategies with the related static strategy values but rather to demonstrate the effect on CPUF of different dynamic strategies, any of which might be implemented without sufficient investigation. To elaborate briefly, we acknowledge that the desire to reduce response time can motivate the use of a dynamic strategy, but the choice of strategy should reflect consideration of the resulting effect on CPU efficiency.

Figure 3, presented in six parts reflects the effect of a saturation point in terms of the number of peripheral units. Following Kleinrock [15, p. 840] we can define a saturation point (N^*) expressed in the number of peripheral units supported by the system as

$$N^* = \frac{1/\mu + \delta + 1/\lambda}{1/\mu + \delta}$$

The resulting values for N^* are indicated on the graphs of Figure 3. With reference to these values, we make the following observations:

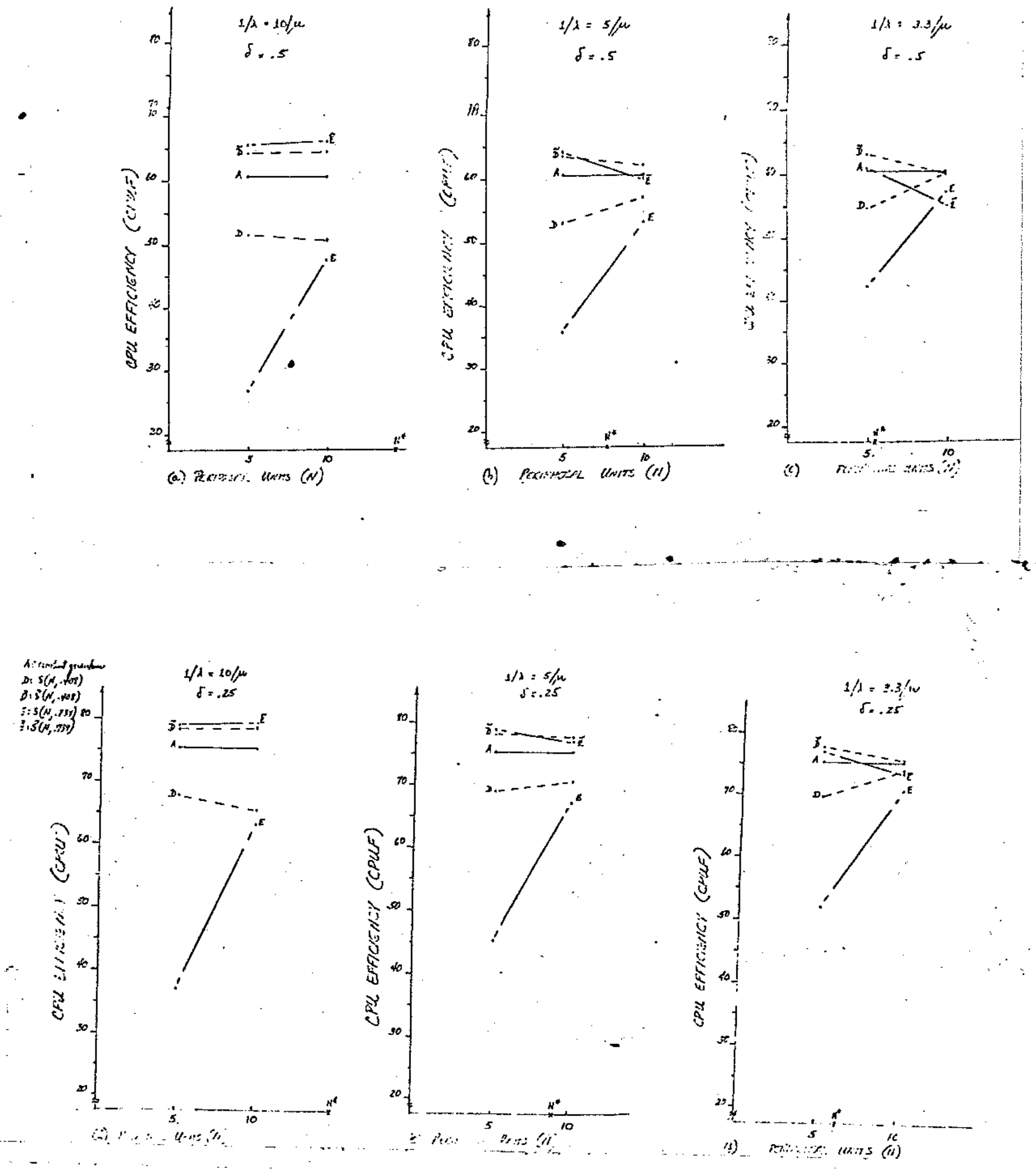
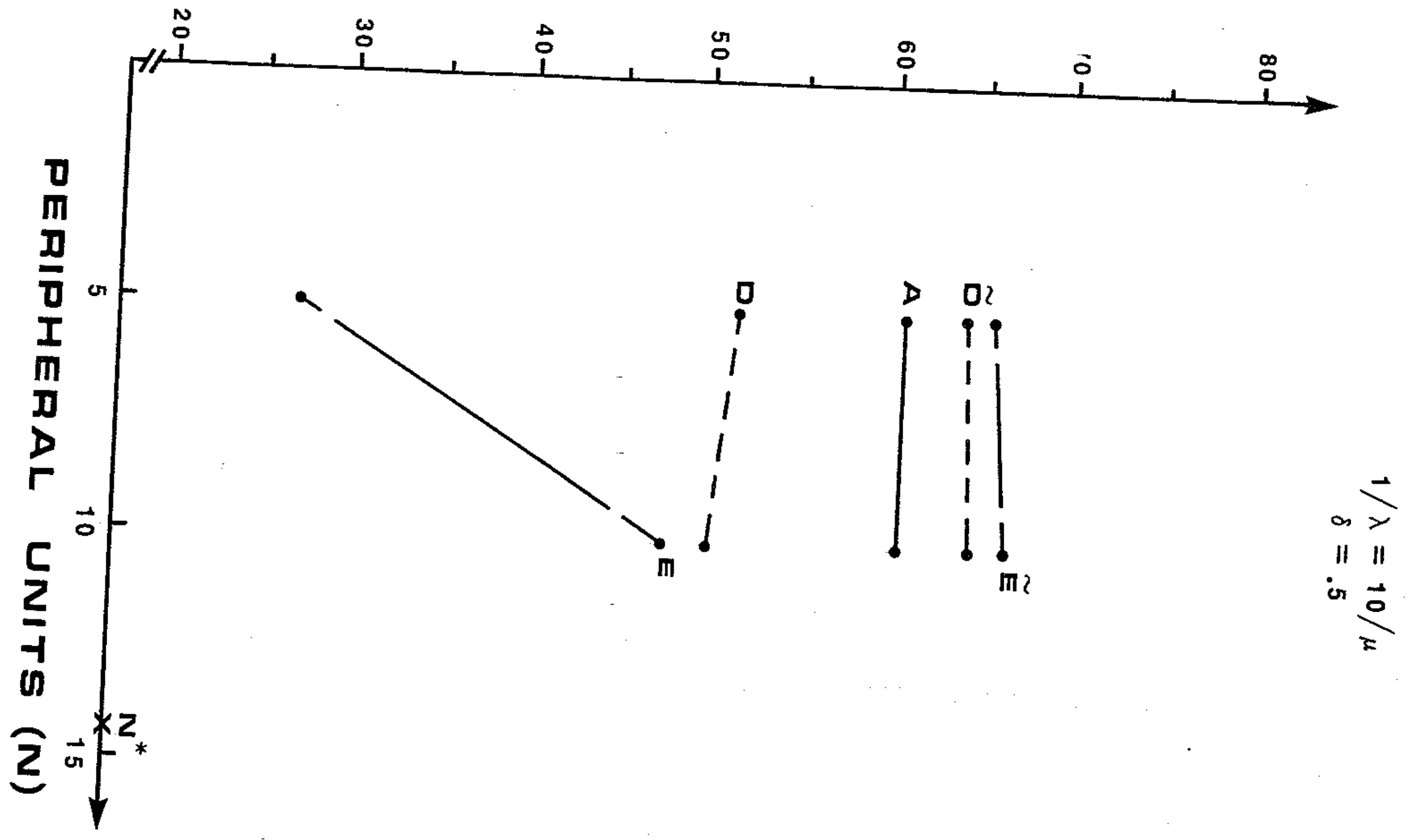


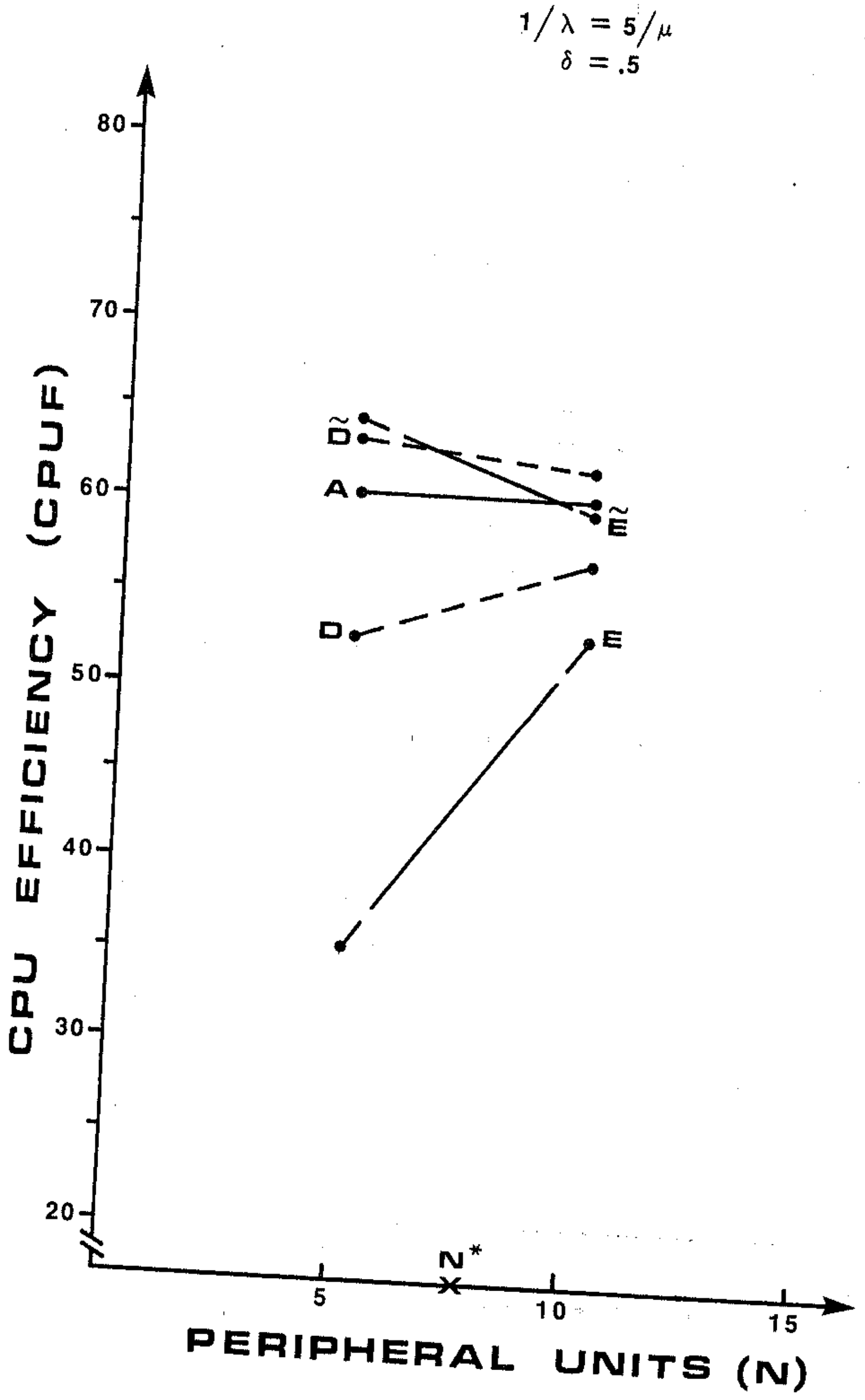
Figure 3. The Effectof the Number of Peripheral Units (N) on CPU Efficiency (CPUF) with Different Demand Rates (λ) and Overhead (δ)

(Overhead Expressed in Units of 1/μ.)

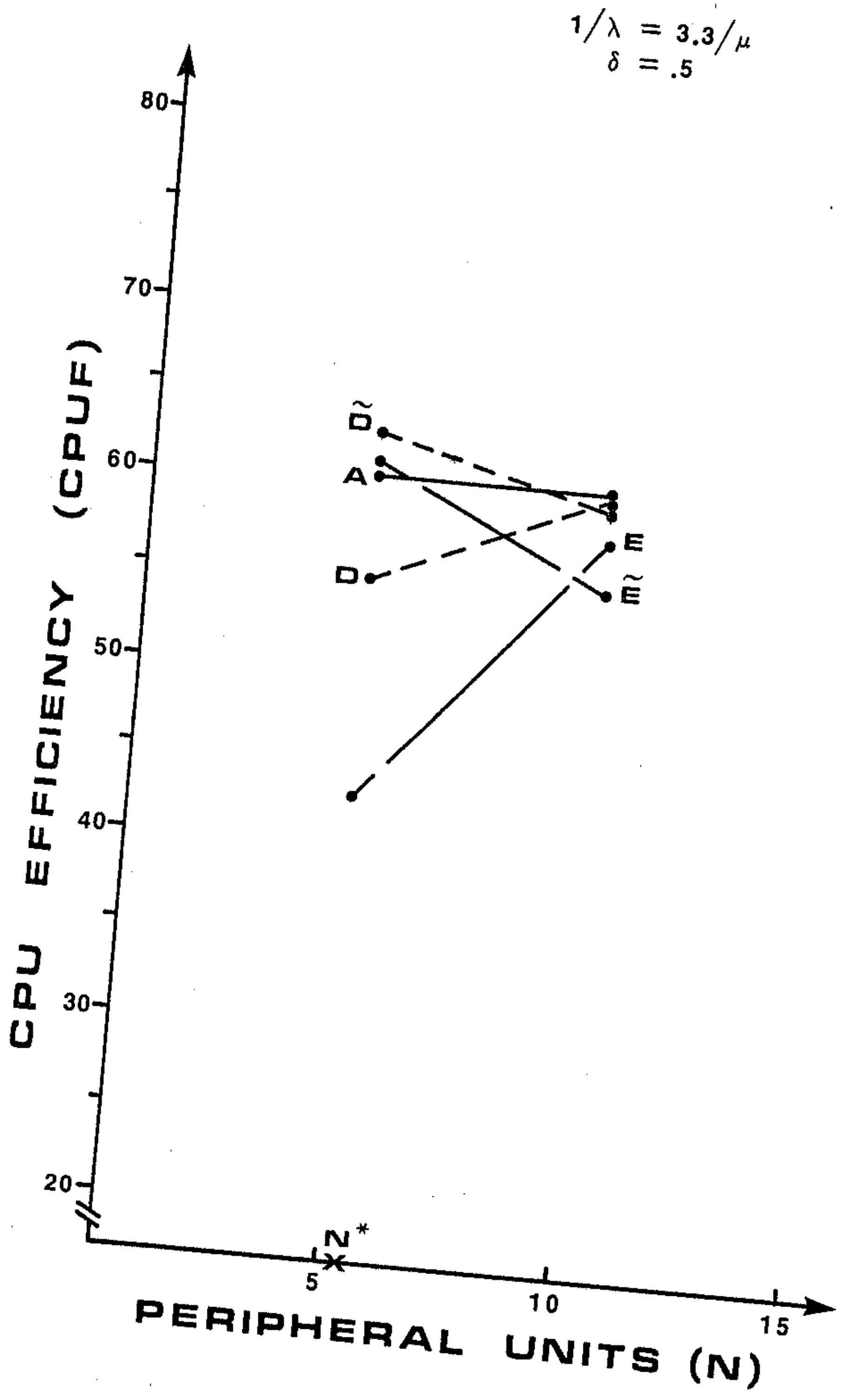
of the Number of Peripheral Units (N) on CPU Efficiency (CPUF) with Different Demand Rates (λ) and Overhead (δ)

(Overhead Expressed in Units of $1/\mu$.)

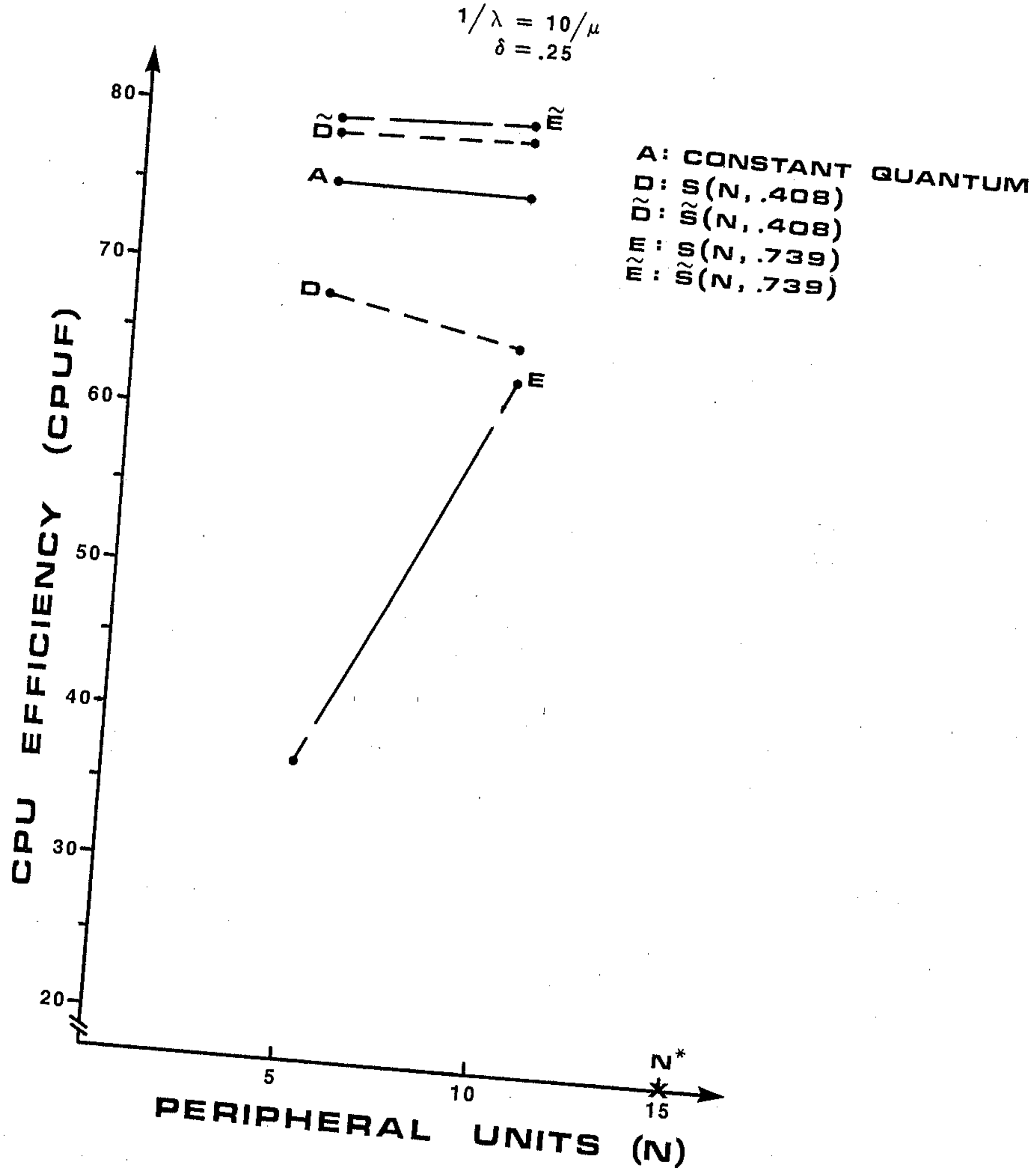


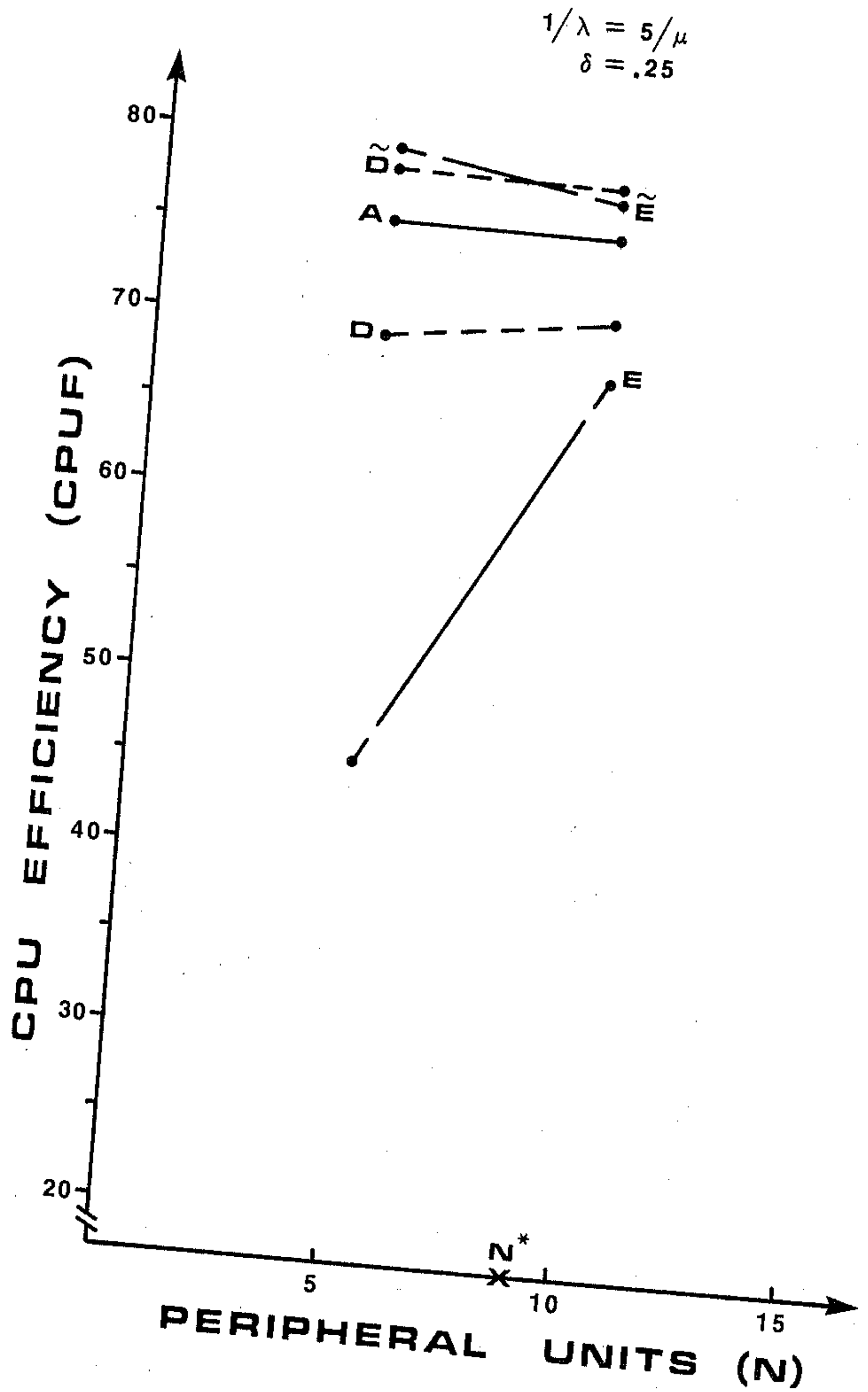


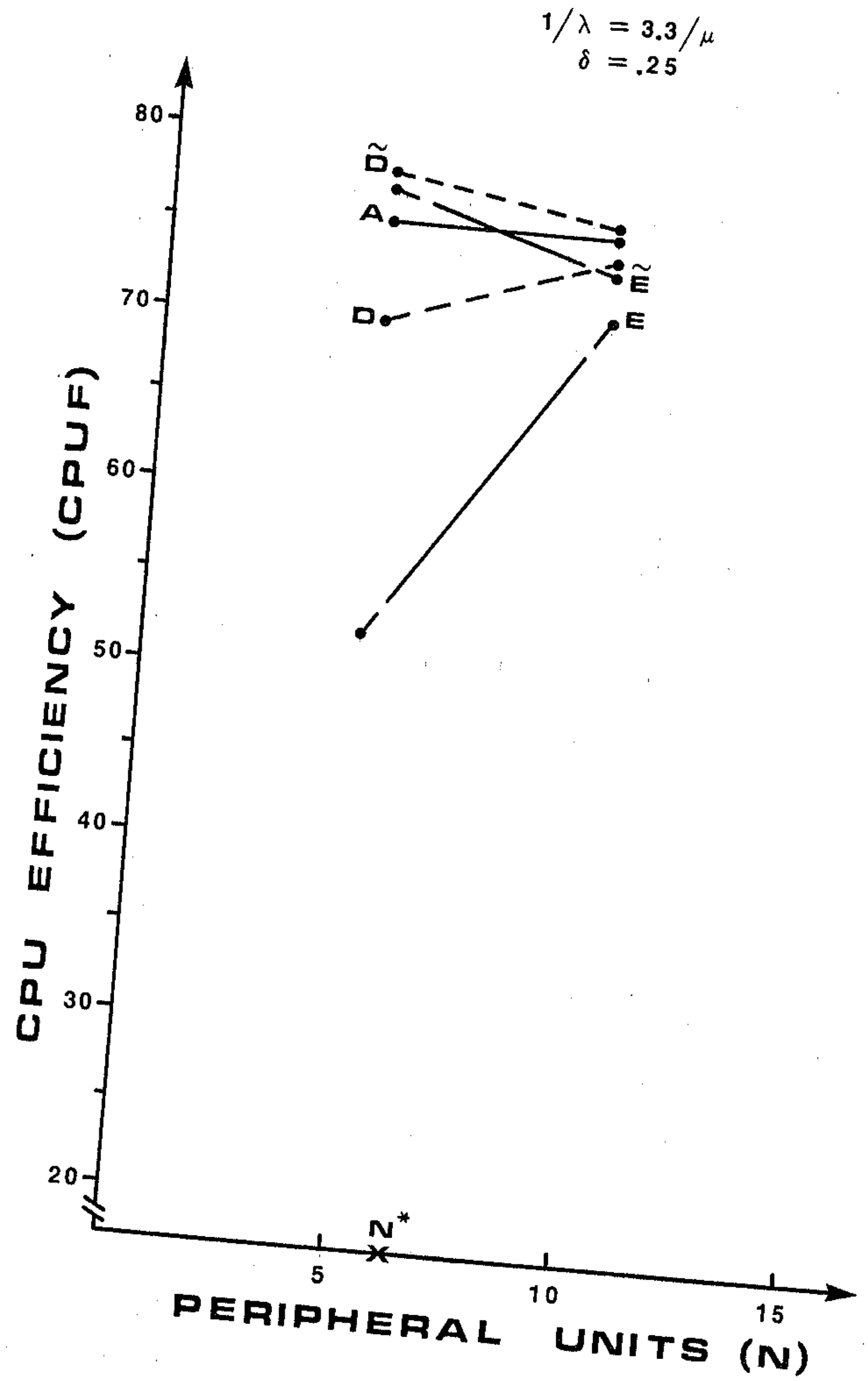
$1/\lambda = 5/\mu$
$\delta = .5$
CPU EFFICIENCY (CPUF)
80
70
60
50
40
30
20
$\tilde{D}$
A
D
$\tilde{E}$
E
PERIPHERAL UNITS (N)
5
N^*
10
15



$1/\lambda = 3.3/\mu$
$\delta = .5$
CPU EFFICIENCY (CPUF)
PERIPHERAL UNITS (N)
20
30
40
50
60
70
80
5
10
15
N^*
$\tilde{D}$
A
D
E
$\tilde{E}$


$1/\lambda = 10/\mu$
$\delta = .25$
A: CONSTANT QUANTUM
D: S(N, .408)
$\tilde{D}$: $\tilde{S}$(N, .408)
E: S(N, .739)
$\tilde{E}$: $\tilde{S}$(N, .739)
$\tilde{D}$
$\tilde{E}$
A
D
E
CPU EFFICIENCY (CPUF)
20
30
40
50
60
70
80
PERIPHERAL UNITS (N)
5
10
15
N*


$1/\lambda = 5/\mu$
$\delta = .25$
CPU EFFICIENCY (CPUF)
20
30
40
50
60
70
80
PERIPHERAL UNITS (N)
5
10
15
N^*
$\tilde{D}$
$\tilde{E}$
A
D
E



$1/\lambda = 3.3/\mu$
$\delta = .25$
$\tilde{D}$
A
D
$\tilde{E}$
E
N^*
CPU EFFICIENCY (CPUF)
PERIPHERAL UNITS (N)
20
30
40
50
60
70
80
5
10
15

4. An increasing quantum allocation strategy ($S(N,d)$) with high dispersion is distinctly inferior under non-saturation conditions ($N < N^*$).

5. Differences in the corresponding increasing ($S(N,d)$) and decreasing ($\tilde{S}(N,d)$) strategies are small under saturation conditions ($N^* \leq N$).

Overhead Variability

Unlike the quantum length decision, the amount of overhead required to maintain the time-sharing environment is often viewed as a consequence of the demand load and beyond the control of the system operator. Strictly speaking, overhead can be controlled to some extent by employing scheduling algorithms that differ in the amount of information and the level of job/system analysis used. Blevins and Ramamoorthy [2] extend this idea further in their investigation of dynamic models of CPU scheduling.

We treat the effect on CPU of overhead variability represented by two functions: (1) overhead increasing with the number of active tasks and (2) overhead decreasing with the number of active tasks.

The relationship between a constant overhead and CPUF is given directly by equation (6). Figure 4 pictures this behavior for quantum lengths of 2.5 and 7.5 (in units of $1/\mu$). The difference between the two cases appears to be slight.

In testing the effect of variable overhead, we identify an increasing overhead function by $O(N,d,\bar{\delta})$, where N is the number of input devices, d is the dispersion (computed in the same way as for a quantum strategy), and $\bar{\delta}$ is the

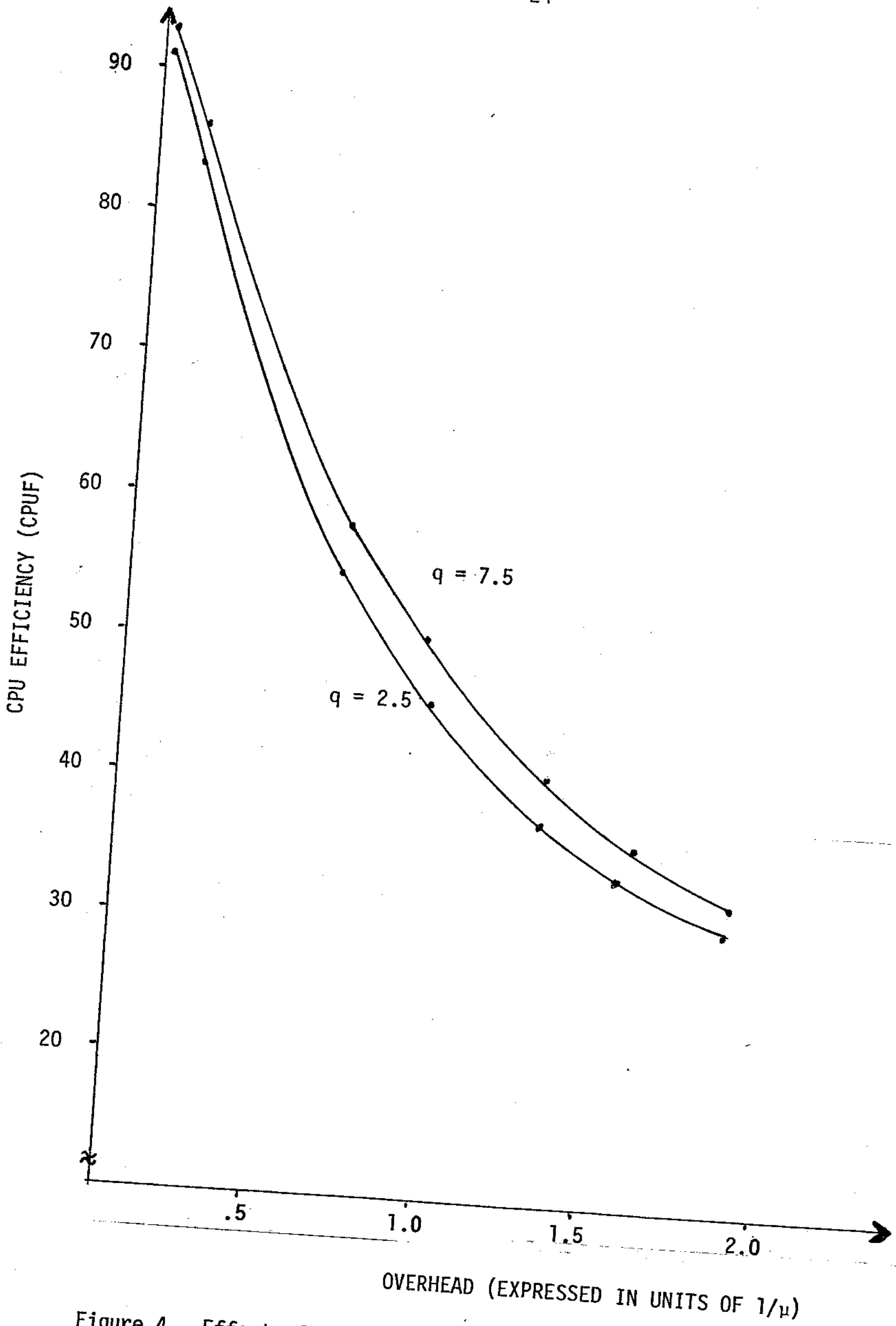


Figure 4. Effect of Constant Overhead Using Two Quantum Values ($N=10$, $\lambda=\mu/2$)

"average" overhead ("average" in the same sense as $\bar{q}$). The overhead functions used in the reported results are shown in Table 2. similar to the testing of quantum allocation strategies, $\tilde{O}(N,d,\bar{\delta})$ indicates a decreasing overhead function.

Table 3 gives the complete results for CPUF under the following specifications:

1. the number of input devices (N) set at five and ten,
2. quantum lengths of 1.5 and 3.0 (in units of $1/\mu$),
3. the demand rate per input device set at .05, .10, and .15 (in units of $1/\mu$),
4. overhead function dispersion values of .283 and .633, and
5. "average" overhead values of .25, .50, and .75.

The most interesting patterns of behavior are identified in several figures composed from the results shown in Table 3. The discussion attempts to interpret the behavior more fully.

Table 3 shows that an increasing overhead function, especially with a high dispersion, causes a considerable increase in CPUF over a constant overhead value except in situations of extremely high demand. Figure 5 illustrates the effect of high demand -- a tendency to obscure any differences among CPUF for different functions. Figure 5(a) shows that the decreasing overhead function can produce higher values of CPUF since the states representing higher congestion are encountered more frequently.

Although increasing CPU efficiency is a prime consideration, stability of response contributes significantly to user satisfaction. The results of Table 3 show that a decreasing overhead function is always less sensitive to demand than an increasing overhead function. This fact only suggests an element of

$O(5, .283, .25) = \{.15, .20, .25, .30, .35\}$
$O(5, .633, .25) = \{.05, .10, .25, .40, .45\}$

$O(5, .283, .50) = \{.30, .40, .50, .60, .70\}$
$O(5, .633, .50) = \{.10, .20, .50, .80, .90\}$

$O(5, .283, .75) = \{.45, .60, .75, .90, 1.05\}$
$O(5, .633, .75) = \{.15, .30, .75, 1.20, 1.35\}$

$O(10, .283, .25) = \{.15, .15, .20, .20, .25, .25, .30, .30, .35, .35\}$
$O(10, .633, .25) = \{.05, .05, .10, .10, .25, .25, .40, .40, .45, .45\}$

$O(10, .283, .50) = \{.30, .30, .40, .40, .50, .50, .60, .60, .70, .70\}$
$O(10, .633, .50) = \{.10, .10, .20, .20, .50, .50, .80, .80, .90, .90\}$

$O(10, .283, .75) = \{.45, .45, .60, .60, .75, .75, .90, .90, 1.05, 1.05\}$
$O(10, .633, .75) = \{.15, .15, .30, .30, .75, .75, 1.20, 1.20, 1.35, 1.35\}$

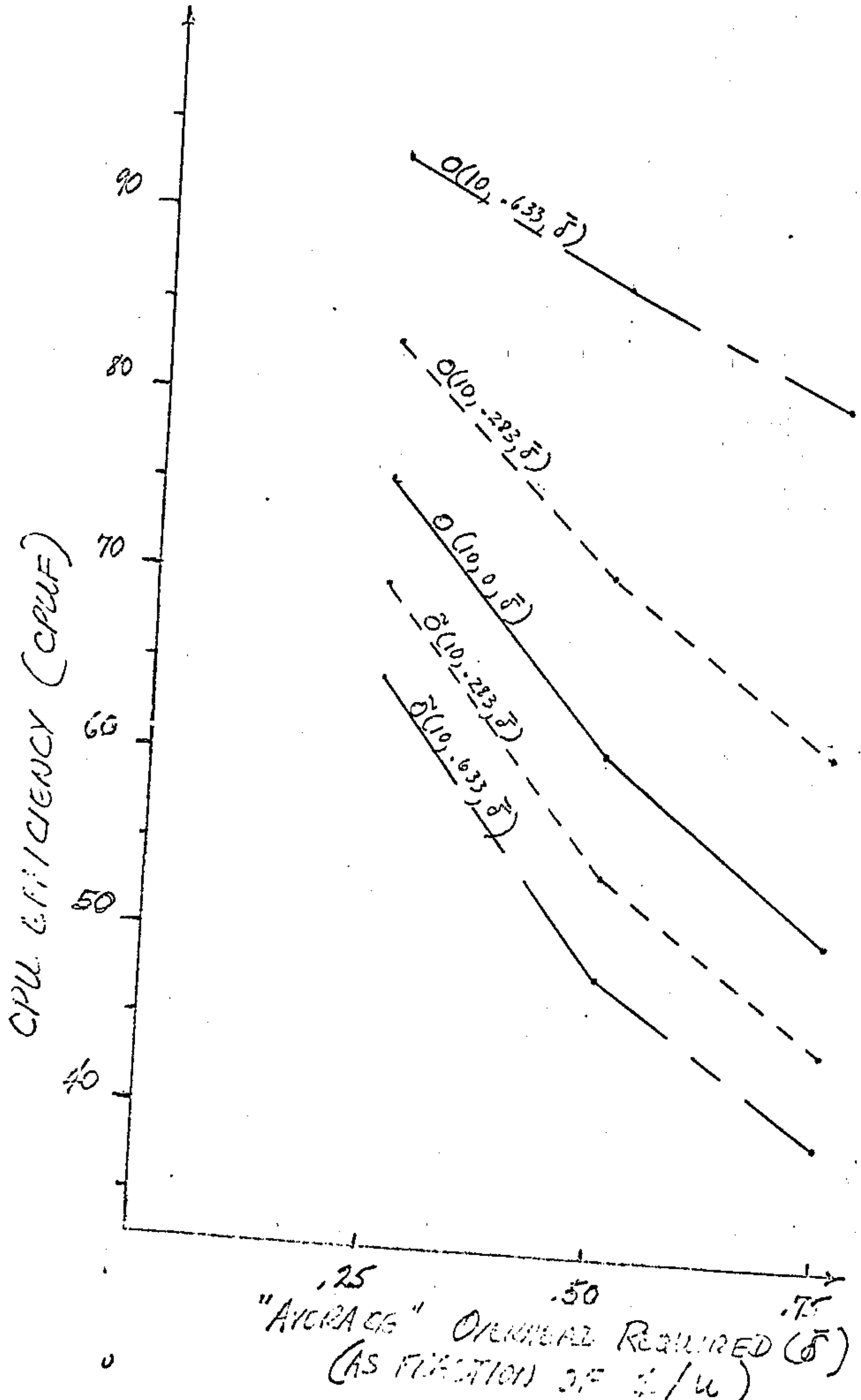
Table 2. The Overhead Functions $O(N,d;\bar{\delta})$ Used in the Testing of Overhead Variability (All Overhead Values in Units of $1/\mu$)

		Quantum Length (q) = 1.5					
		N=5					
		Dispersion (d)					
	λ/μ	d = .283			d = .633		
Overhead (δ)	(μ = 1)	Const.	Incr.	Decr.	Const.	Incr.	Decr.
	.05		83.24	69.40		93.09	63.89
δ = .25	.10	75.65	82.40	70.09	75.65	91.52	64.98
	.15		81.39	70.94		89.44	66.45
	.05		71.16	53.30		86.95	47.20
δ = .50	.10	60.84	69.66	54.43	60.84	83.87	49.00
	.15		67.88	55.79		79.77	51.37
	.05		62.02	43.40		81.44	37.64
δ = .75	.10	50.88	59.97	44.86	50.88	76.88	40.01
	.15		57.58	46.56		70.89	43.01
		N = 10					
	.05		83.14	69.55		93.27	64.08
δ = .25	.10	75.65	80.68	71.73	75.65	88.74	67.79
	.15		77.57	74.26		81.22	72.90
	.05		70.79	53.80		87.04	48.00
δ = .50	.10	60.84	65.87	57.66	60.84	76.78	54.79
	.15		61.12	61.02		62.76	61.77
	.05		61.25	44.37		80.99	39.27
δ = .75	.10	50.88	54.19	49.21	50.88	64.21	47.96
	.15		49.32	52.56		47.55	55.30

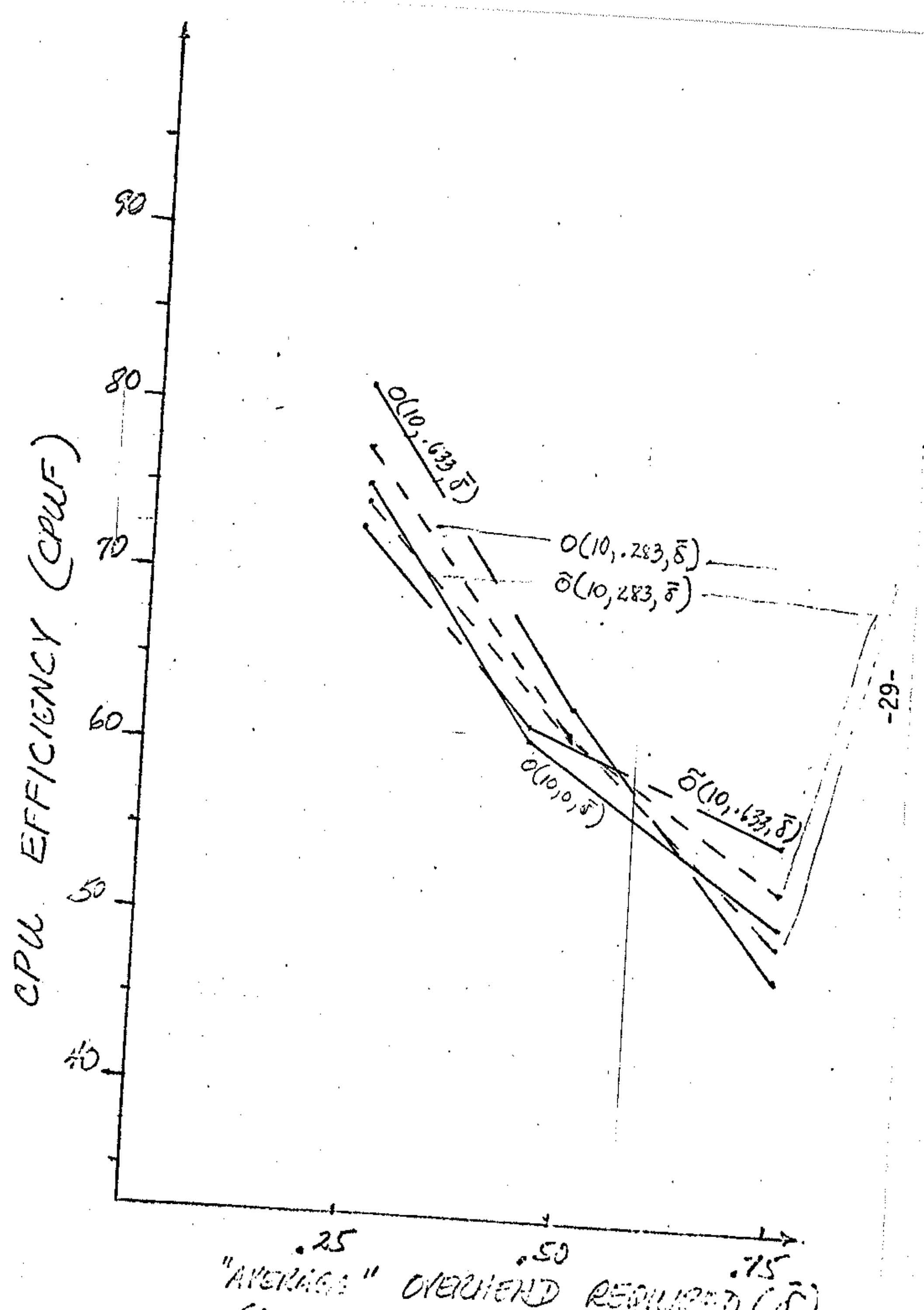
Table 3. Complete Results for Tests of the Increasing and Decreasing Overhead Functions (Values of q and δ in Units of 1/μ).

		Quantum Length (q) = 3.0					
		N=5					
		Dispersion (d)					
	λ/μ	d = .283			d = .633		
Overhead (δ)	(μ = 1)	Const.	Incr.	Decr.	Const.	Incr.	Decr.
δ = .25	.05	79.17	86.11	73.30	79.17	94.62	68.13
	.10		85.60	73.74		93.76	68.77
	.15		84.91	74.34		92.48	69.75
δ = .50	.05	60.84	75.54	57.93	60.84	89.74	51.78
	.10		74.60	58.70		88.03	52.92
	.15		73.31	59.75		85.44	54.64
δ = .75	.05	50.88	67.24	47.97	50.88	85.30	41.88
	.10		65.90	49.02		82.74	43.47
	.15		64.05	50.42		78.81	46.00
		N = 10					
δ = .25	.05	75.65	86.01	73.43	75.65	94.66	68.29
	.10		84.18	75.09		91.65	70.95
	.15		81.60	77.30		85.86	75.24
δ = .50	.05=	60.84	75.25	58.30	60.84	89.62	52.30
	.10		71.50	61.51		82.88	57.71
	.15		67.01	64.72		70.89	64.12
δ = .75	.05	50.88	66.66	48.68	50.88	85.06	42.99
	.10		60.92	53.04		73.40	50.49
	.15		55.69	56.42		56.63	57.56

Total 3. Continued



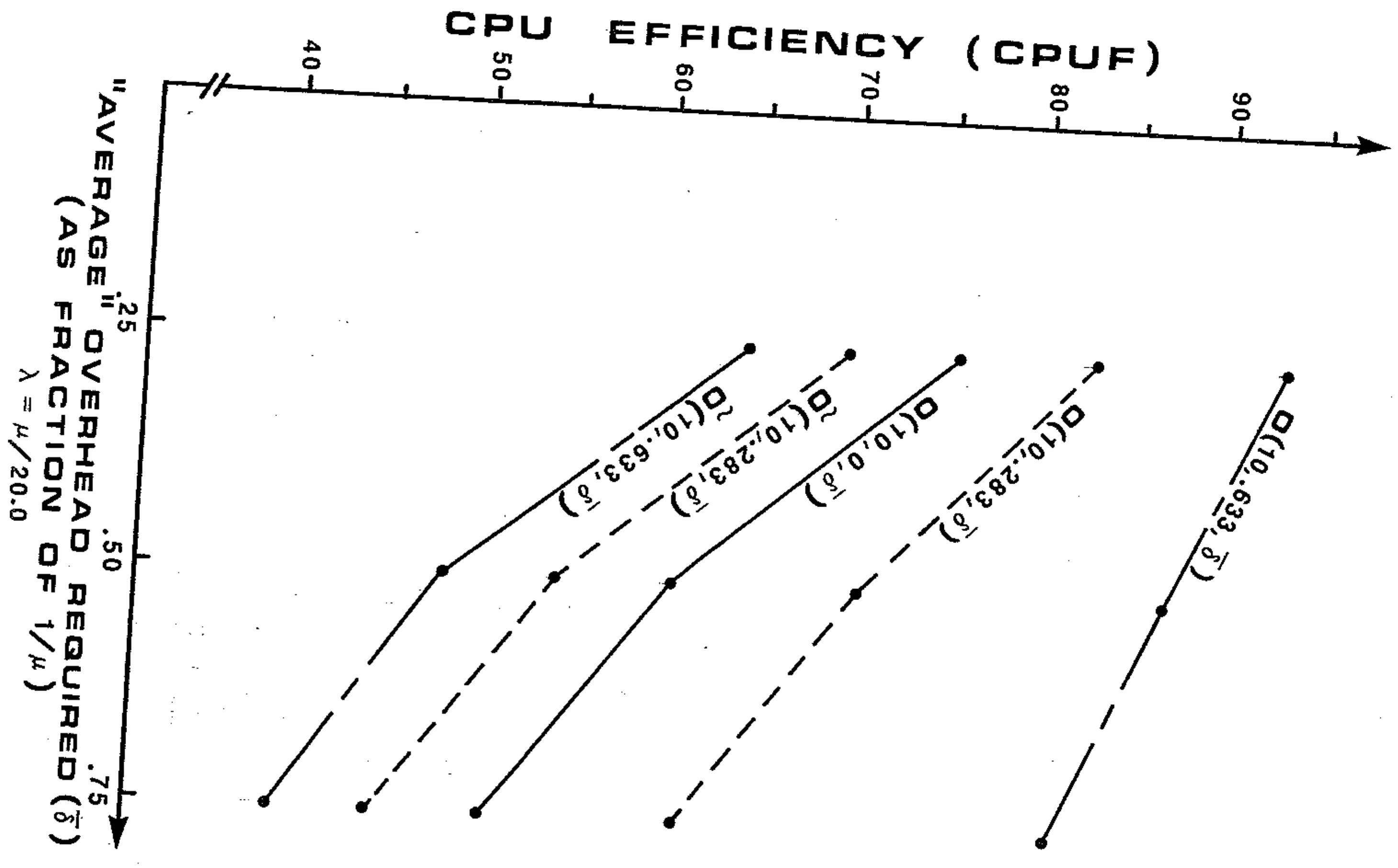
(a) $\lambda = \mu/20.0$

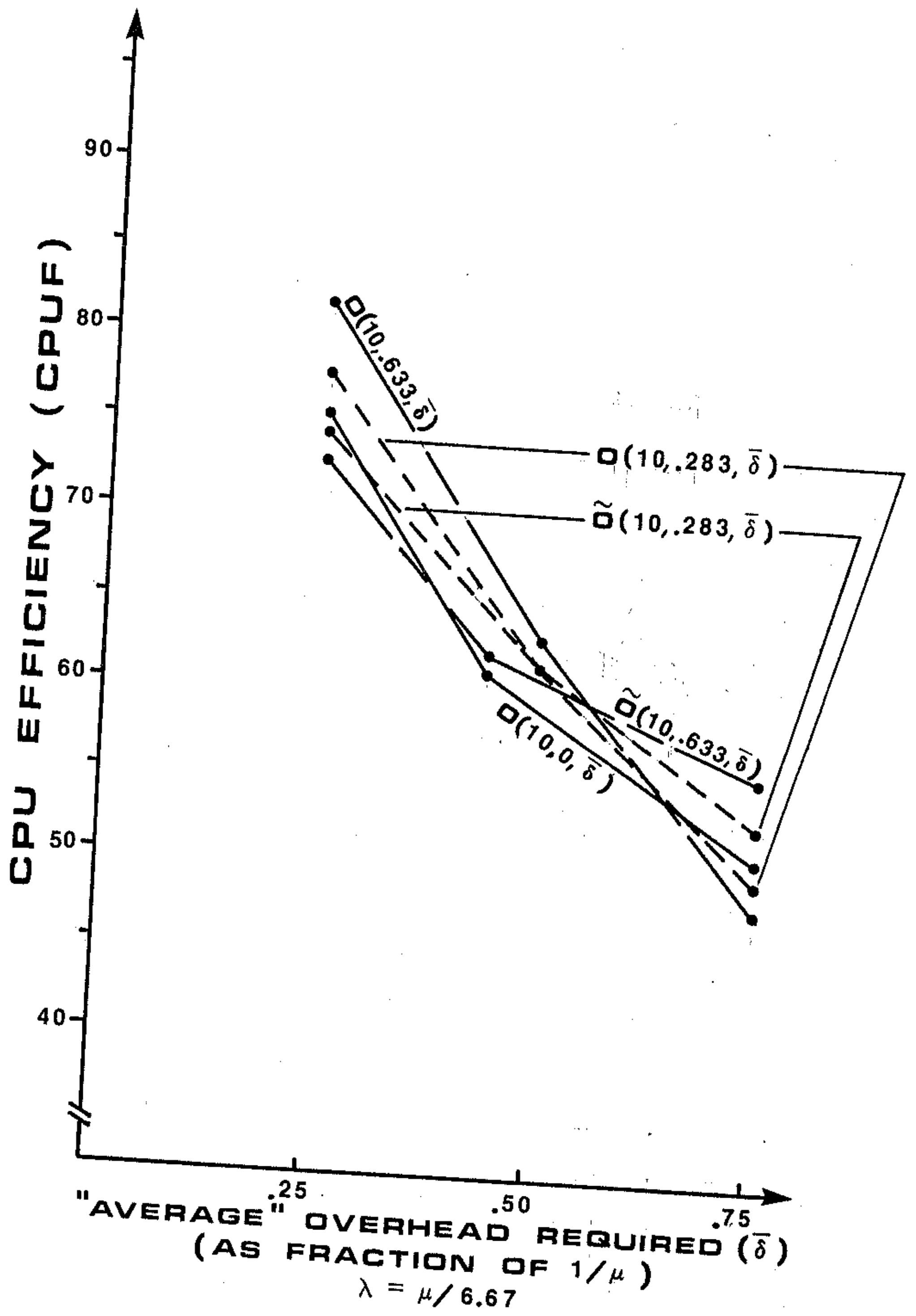


(b) $\lambda = \mu/$

Figure 5. The Effect of Demand Ratio on CPU Efficiency.

igure 5. The Effect if Demand Rate on CPU Efficiency.

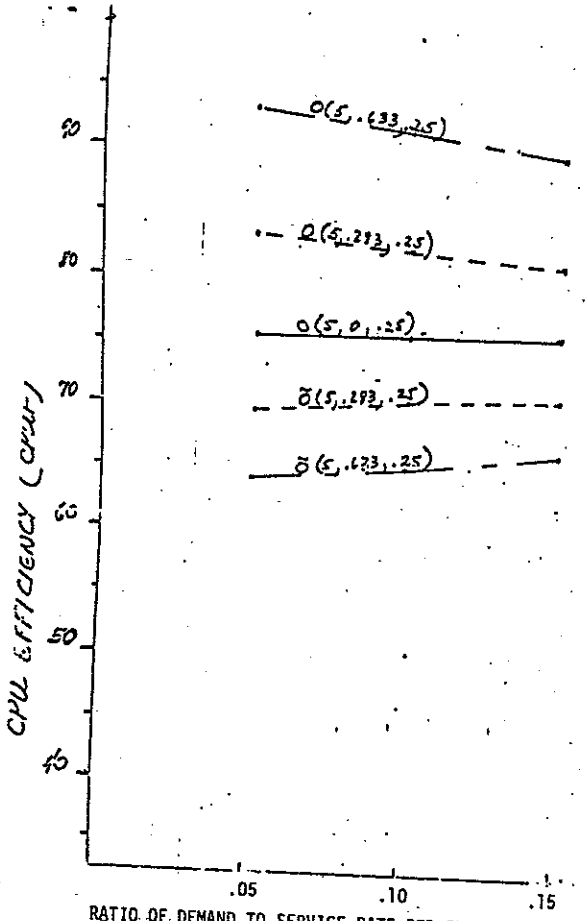



CPU EFFICIENCY (CPUF)
40
50
60
70
80
90
"AVERAGE" OVERHEAD REQUIRED ($\bar{\delta}$)
(AS FRACTION OF $1/\mu$)
$\lambda = \mu/6.67$
.25
.50
.75
$\square(10,.633,\bar{\delta})$
$\square(10,.283,\bar{\delta})$
$\tilde{\square}(10,.283,\bar{\delta})$
$\tilde{\square}(10,.633,\bar{\delta})$
$\square(10,0,\bar{\delta})$

caution in utilizing an increasing overhead function.

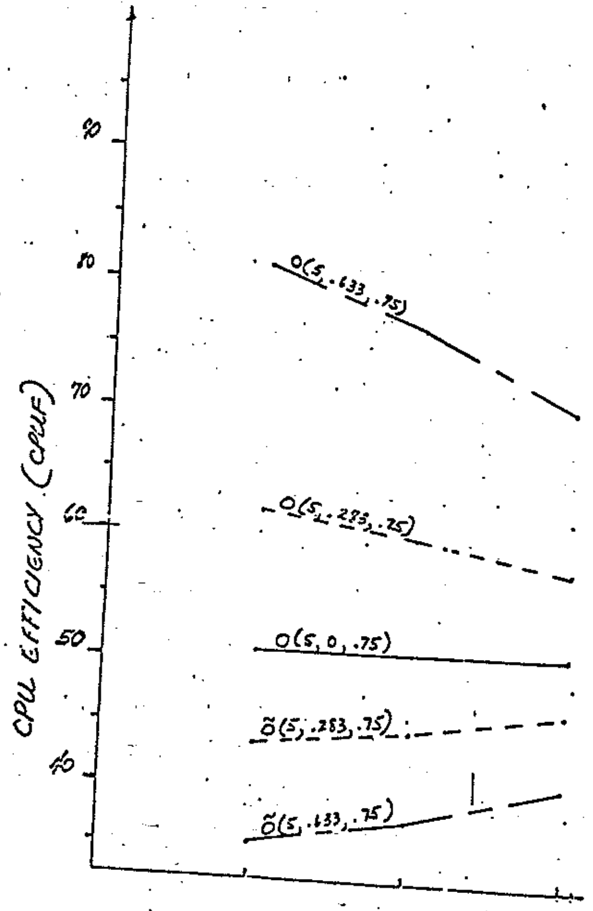
The interaction between demand and the amount of overhead incurred is depicted in the four parts of Figure 6. The effect of increasing the "average" overhead by a factor of three serves to separate the CPUF values in Figure 6(a) and 6(b). The degree of separation is quite large. But when the number of input devices is increased to ten, the pattern of behavior undergoes marked changes. The drop in CPU efficiency for the increasing overhead function O(10, .633, .75) is over 33 percentage points contrasted with only 12 percentage points for O(10, .283, .25) and less than 11 for O(5, .633, .75).

As a final observation we note that a higher dispersion value (with a fixed "average" overhead) produces larger differences between the corresponding increasing and decreasing overhead functions. The claim that variation in overhead can significantly affect CPU efficiency is clearly warranted.

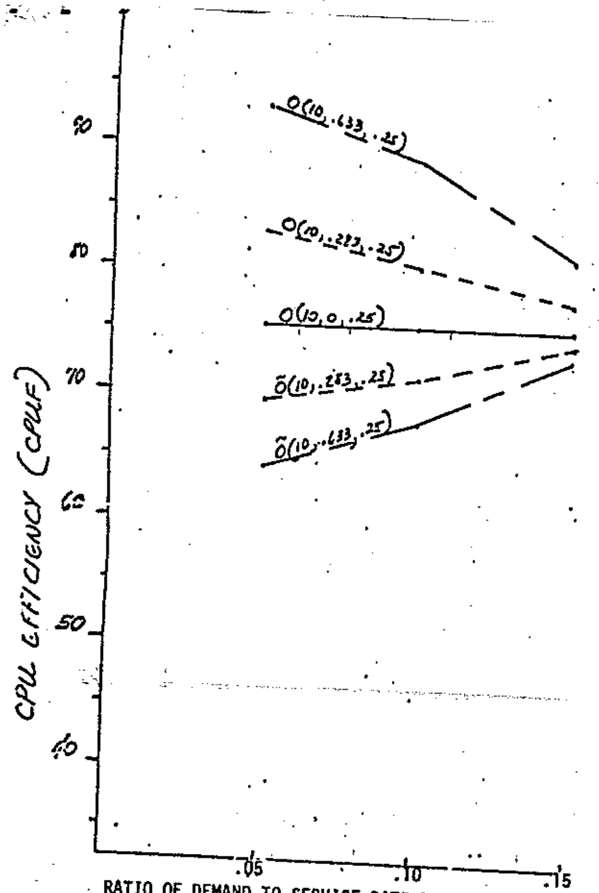


RATIO OF DEMAND TO SERVICE RATE PER INPUT DEVICE

a) With Five Input Devices and Overhead of .25 μ^{-1}.

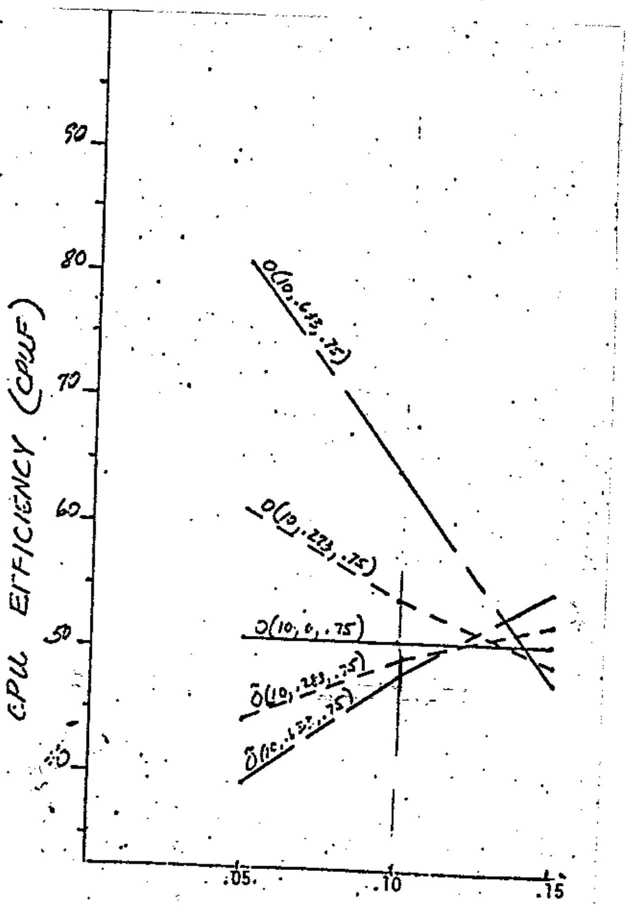


(b) With Five Input Devices and Overhead of .75 μ^{-1}.



RATIO OF DEMAND TO SERVICE RATE PER INPUT DEVICE

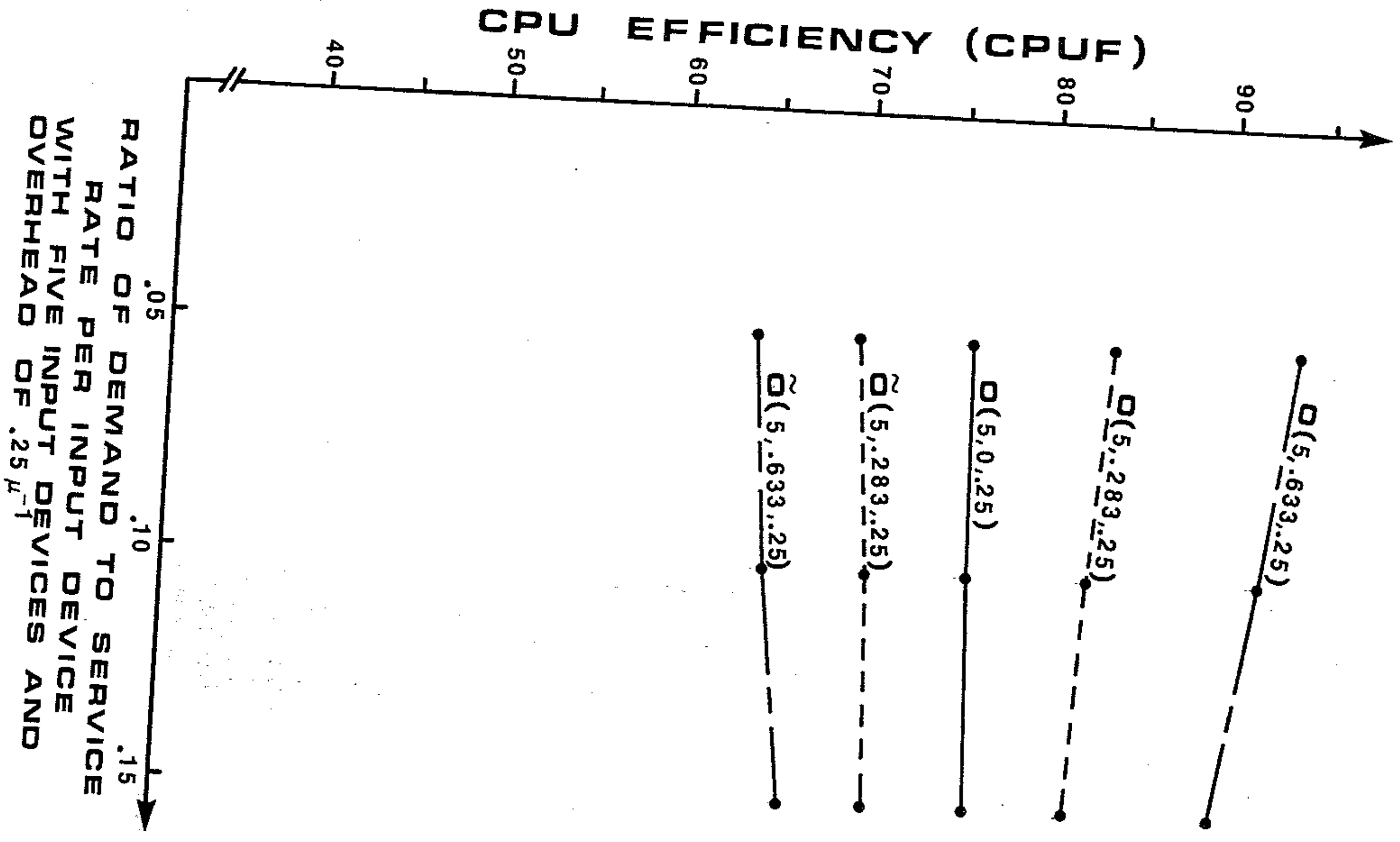
(c) With Ten Input Devices and Overhead of

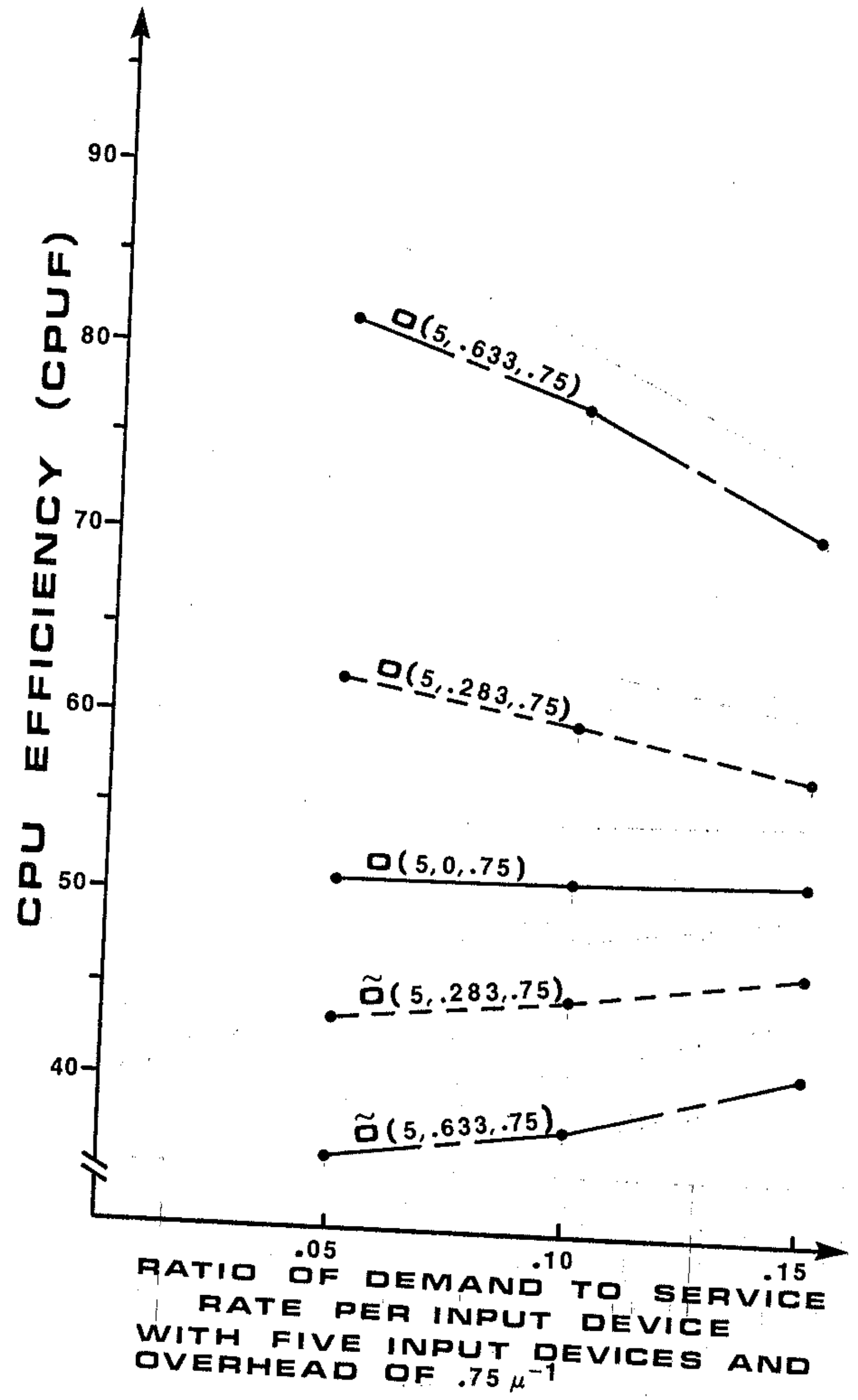


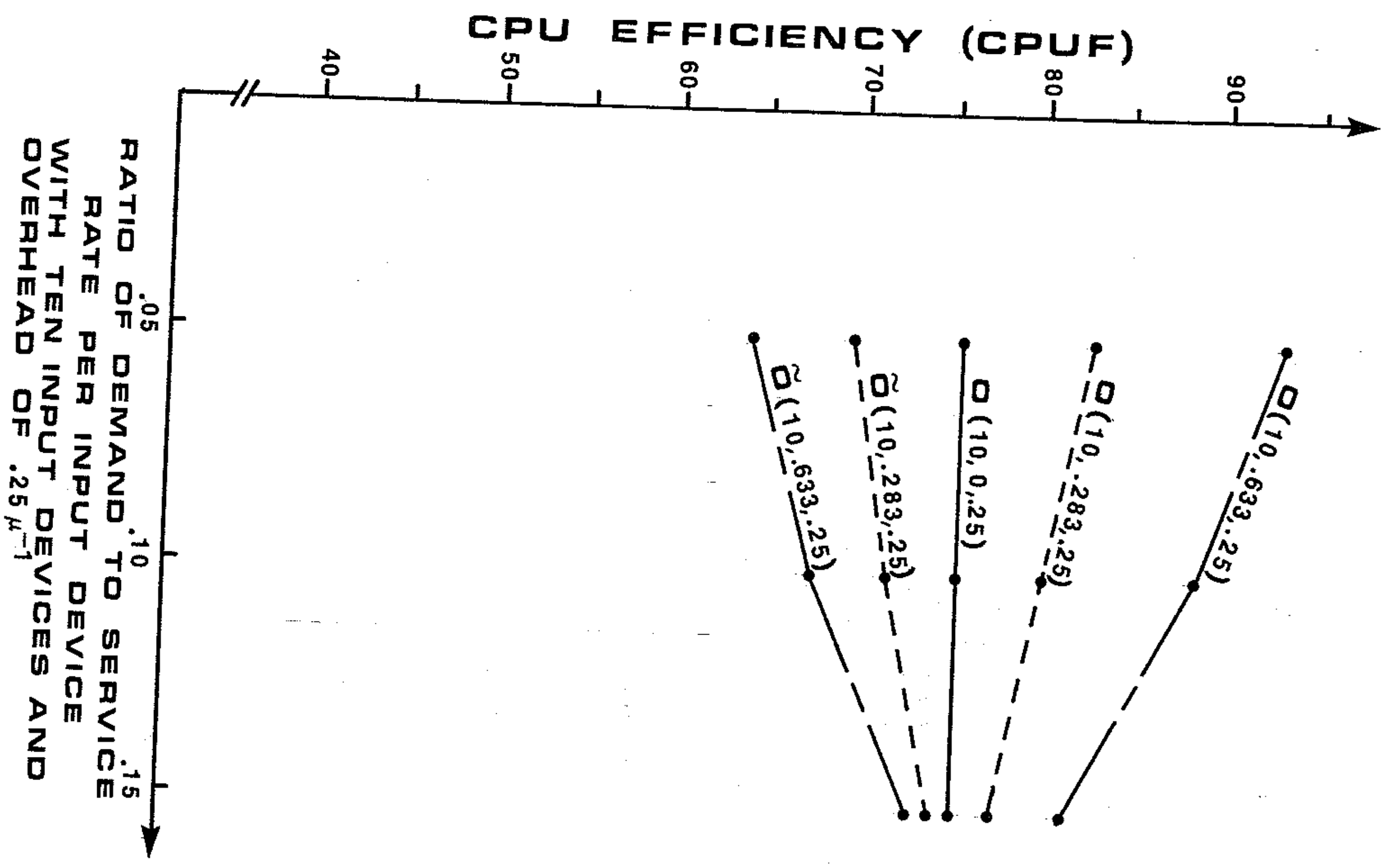
(d) With Ten Input Devices and Overhead of .75 μ^{-1}.

Figure 6. The Effect on CPU Effieiency Caused by Interaction of Demand and the Amount of Overhead.

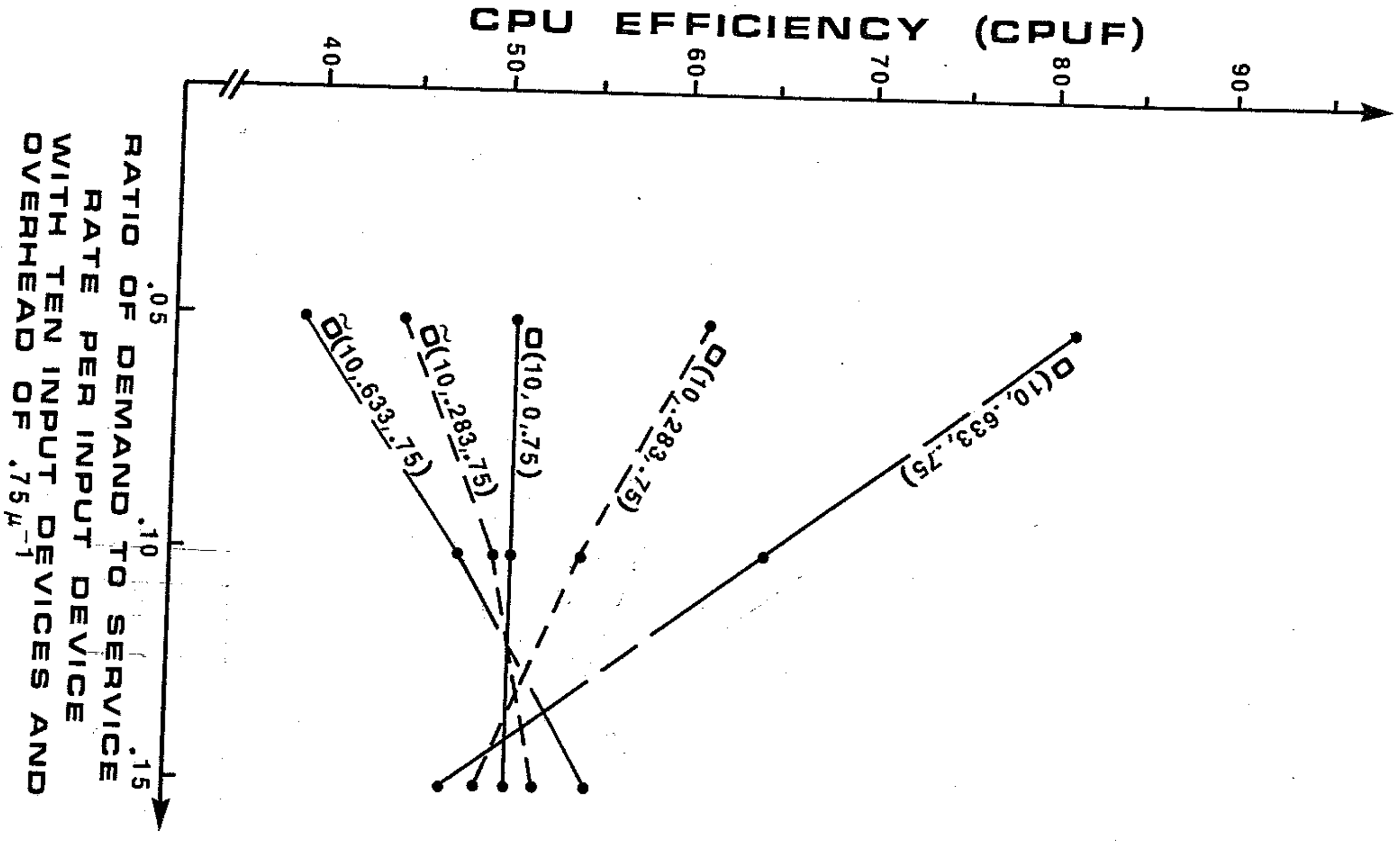
Figure 6. The Effect on CPU Efficiency Caused by Interaction of Demand and the Amount of Overhead.




CPU EFFICIENCY (CPUF)
90
80
70
60
50
40
O(5,.633,.75)
O(5,.283,.75)
O(5,0,.75)
Õ(5,.283,.75)
Õ(5,.633,.75)
.05
.10
.15
RATIO OF DEMAND TO SERVICE RATE PER INPUT DEVICE WITH FIVE INPUT DEVICES AND OVERHEAD OF .75 μ⁻¹



CPU EFFICIENCY (CPUF)
40
50
60
70
80
90
RATIO OF DEMAND TO SERVICE RATE PER INPUT DEVICE WITH TEN INPUT DEVICES AND OVERHEAD OF .25 μ^{-1}
.05
.10
.15
$\tilde{D}(10,.633,.25)$
$\tilde{D}(10,.283,.25)$
D(10,0,.25)
D(10,.283,.25)
D(10,.633,.25)


CPU EFFICIENCY (CPUF)
40
50
60
70
80
90
RATIO OF DEMAND TO SERVICE RATE PER INPUT DEVICE WITH TEN INPUT DEVICES AND OVERHEAD OF .75μ^{-1}
.05
.10
.15
$\tilde{D}(10,.633,.75)$
$\tilde{D}(10,.283,.75)$
D(10,0,.75)
D(10,.283,.75)
D(10,.633,.75)

CONCLUSIONS AND SUMMARY

Several observations are noted in the preceding pages. In repeating the most significant of these observations below, we wish to recall two specific points with regards to this research: (1) the emphasis is directed toward relative strategy comparisons, and (2) while all scheduling policies give the same "average" quantum assignment (q) in the sense of assuming equal state probabilities, the time-average is not equal to $\overline{q}$. However, the time-average quantum assignment can be obtained only by solving the model. The conclusions shown below are limited to the conditions specified for computational results:

1. With increasing demand, increasing quantum strategies cause an increase in CPU efficiency while the reverse is true for decreasing strategies.
2. Higher dispersion among quantum allocations results in larger differences in CPU efficiencies between the corresponding increasing and decreasing strategies.
3. Increasing quantum strategies with higher dispersion have a more pronounced effect on CPU efficiency and are distinctly inferior under low to moderate demand (non-saturation).
4. An increasing overhead function, especially with high dispersion, causes a considerable increase in CPU efficiency over that of constant overhead except in high demand situations.
5. In terms of CPU efficiency, a decreasing overhead function is less sensitive to demand than an increasing function.
6. The interaction between demand (in terms of the number of users served and the rate of job submissions) and the amount of overhead has a significant effect on CPU efficiency.

The models developed during this research offer tools that are both powerful and practical. Specific system configurations can be conveniently investigated. The effect of variability in quantum allocation strategies and overhead functions on CPU efficiency is being investigated further. Extensions of this research are directed toward the identification of "best" strategies which include both the user and operator perspectives.

REFERENCES

1. Bhat, U. Narayan and Richard E. Nance, "Busy Period Analysis of a Time-Sharing System Modeled as a Semi-Markov Process," J.ACM, 18, (2): April 1971, pp. 221-238.

2. Blevins, Parker R. and C.V. Ramamoorthy, "Aspects of a Dynamically Adaptive Operating System," IEEE Trans. on Comp., C-25 (7): July 1976, pp. 713-725.

3. Bunt, R.B. and J.N.P. Hume, "Self-Regulating Operating Systems," INFOR, 10 (3): October 1972, pp. 232-239.

4. Chang, W. "A Queueing Model for a Simple Case of Time-Sharing," IBM Syst. J., 5 (2): 1966.

5. Clark, S.R. and T.A. Rourke, "A Simulation Study of the Effects of Various Job-Scheduling Algorithms in Computer Systems," INFOR, 10 (3): October 1972, pp. 205-220.

6. Coffman, Edward G., Jr. "Analysis of Two Time-Sharing Algorithms Designed for Limited Swapping," J.ACM, 15 (3): July 1968, pp. 341-353.

7. Coffman, Edward G., Jr. and Leonard Kleinrock, "Feedback Queueing Models for Time-Shared Systems," J.ACM, 15 (4): October 1968, pp. 549-576.

8. Coffman, Edward G., Jr. and R.R. Muntz, "Models of Pure Time-Sharing Disciplines for Resource Allocation," Proc. ACM National Conf. 1969, pp. 217-228

9. Coffman, Edward G., Jr. and Peter J. Denning, Operating Systems Theory, Prentice-Hall, 1973.

10. Gaver, Donald P., "Probability Models for Multiprogramming Computer Systems," J.ACM, 14 (3): July 1967, pp. 423-438.

11. Gaver, Donald P. and G.S.Shedler, "Processor Utilization in Multiprogramming Systems Via Diffusion Approximations," Research Report NPS55GV720514, Naval Postgraduate School, Monterey, California, May 1972.

12. Heacox, Harry C. and Paul W. Purdom, Jr. "Analysis of Two Time-Sharing Queueing Models," J.ACM, 19 (1): January 1972, pp. 70-91.

13. Hoare, C.A.R. and R.H. Perrott (Eds), Operating Systems Techniques, Academic Press, 1972.

14. Kemeny, J.G. and J.L. Snell, Finite Markov Chains, D. Van Nostrand, 1960

15. Kleinrock, Leonard, "Certain Analytic Results for Time-Sharing Processors," Proc. IFIP Conf. 1968, Vol. 2, Amsterdam, pp. 838-845.

16. Kleinrock, Leonard, "Swap-Time Considerations in Time-Shared Systems," IEEE Trans. Computers, C-19 (6): June 1970, pp. 534-540.

17. Krishnamoorthi, B. and Roger C. Wood, "Time-Shared Operations with Both Interarrival and Service Time Exponential," J.ACM, 13 (3): July 1966, pp. 317-338.

18. Mullery, A.P. and G.C. Driscoll, "A Processor Allocation Method for Time-Sharing," C.ACM, 13 (1): January 1970, pp. 10-14.

19. Nance, Richard E., U. Narayan Bhat and Billy G. Claybrook, "Busy Period Analysis of a Time-Sharing System: Transform Inversion," J.ACM, 19 (3): July 1972, pp. 453-463.

20. Nance, Richard E. and U. Narayan Bhat, "A Processor Utilization Model for a Multiprocessor Computer System," Opns. Res., to appear.

21. Potier, D., E. Gelenbe and J. Lenfant, "Adaptive Allocation of Central Processing Unit Quanta," J.ACM, 23 (1): January 1976, pp. 97-102.

22. Sayers, Anthony P. (ed.), Operating Systems Survey, Auerbach Publishers, 1971.

23. Shedler, G.S., "A Cyclic-queue Model of a Paging Machine," IBM Research Report RC-2814, IBM Watson Research Center, Yorktown Heights, New York, 1970.