Technical Report CS73001-R

DYNAMIC QUANTUM ALLOCATION AND SWAP-TIME VARIABILITY IN TIME-SHARING OPERATING SYSTEMS+*

U. Narayan Bhat[1]

and

Richard E. Nance[2]

April 1973

[1]Department of Computer Science and Operations Research, Institute of Technology, Southern Methodist University, Dallas, Texas

[2]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia

+Research supported by NSF Grant GK-19537.

*We wish to acknowledge the helpful suggestions and comments of Dr. James E. Kalan during the course of this research.

INTRODUCTION

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 or swap-time.

A quite natural view is that the quantum represents a period of effective processing while the swap-time is a necessary but nonproductive time requirement. 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 [6].

The quantum length decision is typical of the trade-off situations so often encountered in the design of computer hardware and software. A prime consideration is the "single user illusion", i.e. the need to make each user feel that the complete capability of the computer system is available to him. The "single user illusion" suggests a relatively short quantum so that, even in a high demand situation, each program is being assigned the CPU rather frequently. A counter consideration, however, is the increased overhead incurred for transferring CPU control. With a shorter quantum the proportion of a fixed time period devoted to overhead increases (since more swapping results). These two contrary considerations reflect the more general competitive views that we have termed the user and operator perspectives [1].

While the quantum length decision has been resolved typically by establishing a constant value that appears to work well over the varying demand periods and for the various user classes, an interesting possibility is to allocate varying length quantum dynamically. Consequently, the time for which a specific program controls the CPU can vary depending on the current CPU demand, i.e. the number of programs in queue. Also of interest is the ability to vary swap-time as well.

The model developed in this research explores the effect of dynamic quantum allocation on CPU behavior. Additionally, and even more significant, we note the change in CPU behavior stemming from swap-time variability. Computational results are included to illustrate the range of behavior for a hypothetical time-sharing system.

QUANTUM ALLOCATION AND SWAP-TIME VARIABILITY

Quantum allocation

A benefit of allocating quantum of different lengths is that higher priority programs are processed quicker than lower ones. 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 effective use of the CPU. 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 swapping the CPU between only two programs represents an inefficiency that, if removed

by running each program to completion without swapping, 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 affect the "single user illusion" detrimentally 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 [3] proposes two discrete time models that classify jobs as requiring a single quantum or multiple quanta. In the summary of his paper, Coffman [3, p. 352] notes the need for generalizing the models "to include an arbitrary 'quantum-function' of the number in the system." Heacox and Purdom [9] 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 [2] 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 a different quantum accomodates short debugging runs as the author mentions [2, p. 122].

Swap time variability

Swap time or overhead is considered negligible in some models [4,5,11] and a constant value in others [9,13]. 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 swap time for a single transfer of CPU control is dependent on the number of programs in the system, i.e. with more programs seeking service, more comparisons must be made to determine the program assigned CPU control.

The model

We begin with the assumption of a single CPU that performs all tasks related to the swapping of programs. Our objective is to determine the effective utilization of the CPU under different conditions for quantum allocation and swap time. Effective CPU utilization (U) is defined to be the ratio of the expected amount of actual processing time during a CPU busy period [1, p. 223] to the expected length of the busy period, i.e.

$$U = \frac{E\{\text{total actual processing time}\}}{E\{\text{busy period}\}} = 1 - \frac{E\{\text{total swap time}\}}{E\{\text{busy period}\}}.$$

This measure -- effective CPU utilization -- should be distinguished from that used by Gaver [7, p. 424], which he calls CPU productivity

$$\text{CPU productivity} = \frac{E\{\text{busy period}\}}{E\{\text{busy period}\} + E\{\text{idle period}\}}$$

The CPU productivity measure, subsequently termed CPU utilization by Shedler [16] and Gaver and Shedler [8], is a measure of the expected time the CPU is processing programs during an expected operating period (this is denoted as the busy cycle [1]). Note that an idle period, i.e. CPU idleness, is defined to be that period during which no programs are seeking service (the system is empty).

By defining effective CPU utilization to reflect only the loss of time due to swapping, we have excluded some additional sources of CPU idleness. For instance, the case where the CPU is idle because all programs are completing I/O activities is ignored. In effect we are counting only a portion of the time the system is in supervisor mode [15, p. 313] as contributing to a loss in effectiveness. This is justified since we wish to investigate the effect of quantum allocation and swap-time assumptions on the CPU behavior. Time required for tasks other than swapping might remain in an environment with no time-sharing, e.g. a serial batch operating system. Moreover, the time in supervisor mode not related to swapping is dependent on the individual program and the mix of programs rather than on decisions related to swap time and/or quantum 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 the CPU, i.e. submit a single program at a time to the CPU,

(3) demand for the CPU by a "free" input device is Poisson, i.e. for a "free" input device at time t

Pr{input device demands the CPU during $(t,t+dt]$}

$$= \lambda dt + o(dt),$$

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

(5) the probability of more than one event in the interval $(t,t+dt]$ is negligible, i.e. $o(dt)$.

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 further service. In either case a swap time 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 (Q_i+D_i), which are called tasks.

Within a CPU 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 $n^{\underline{th}}$ task, conditioned on i programs seeking the CPU on completion of the $(n-1)^{\underline{st}}$ task, i.e. for $J_{n-1}=i$

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

Then the time devoted by the CPU to the $n^{\underline{th}}$ task ($B_i(t)$) for specific realizations of Q_i and D_i, i.e. $Q_i=q_i$ and $D_i=\delta_i$, is given by

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

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

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

Let

$$Y_i(x) = \Pr\{D_i\leq x\},$$

$$\bar{\delta}_i = E\{D_i\}, \text{ and}$$

$$\sigma_i(\theta) = \int_0^\infty e^{-\theta x} dY_i(x) .$$

Then

$$\beta_i(\theta) = \frac{\sigma_i(\theta)\left[\mu+\theta e^{-(\theta+\mu)q_i}\right]}{\theta+\mu}$$

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

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

where

$$\eta_i = \bar{\delta}_i + \mu^{-1}\left(1-e^{-\mu q_i}\right).$$

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 swap-time:

(1) constant quantum and swap-time,

(2) constant swap-time, variable quantum, and

(3) constant quantum, variable swap-time.

All four cases (the fourth considering both as state-dependent random variables) have physical significance in the modeling of time-sharing systems.

In the first case, the effective utilization of the CPU (U) is determined simply as

$$U = 1 - \frac{\delta}{\eta} \qquad \delta \leq \eta$$

(since the swap-time and quantum are both constant neither δ nor η is subscripted). The result in this case is developed in a paper by Kleinrock [12], 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 three cases require a more extensive analysis which we develop in terms of the most general conditions, i.e. both swap-time and quantum assumed to be random variables. Development of the initial relationships below is presented in more detail in previous papers [1,14].

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

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

For the case where $Q_i=q_i$ and $D_i=\delta_i$, these transition probabilities are determined as

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

$$dA_{ij}(x) = \begin{cases} \mu e^{-\mu(x-\delta_i)}\binom{N-i}{j-i+1}(1-e^{-\lambda x})^{j-i+1} \\ \qquad \cdot (e^{-\lambda x})^{N-j-1}dx & \delta_i < x < \delta_i + q_i \\ e^{-\mu(\delta_i+q_i)}\binom{N-i}{j-i}\left(1-e^{-\lambda(\delta_i+q_i)}\right)^{j-i}\left[e^{-\lambda(\delta_i+q_i)}\right]^{N-j} \\ \qquad \cdot h(\delta_i+q_i-x) & i=1,2,\ldots,N;\ i\leq j \end{cases}$$

where $h(z)$ is the Dirac delta function, i.e.

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

In the more general case, with Q_i and D_i as random variables, we have

$$\beta_i(\theta) = \sigma_i(\theta)\frac{[\mu+\theta\gamma_i(\theta+\mu)]}{\theta+\mu}$$

where $\gamma_i(\theta)$ is the LS transform of the distribution function for terminal access during the quantum period, denoted by $C_i(y)$. Then

$$dA_{i,i-1}(x) = \int_{y=0}^{\infty}\int_{t=0}^{\min(y,x)}\mu e^{-\mu t}dY_i(x-t)(e^{-\lambda x})^{N-i}dC_i(y) \qquad x\geq 0$$

$$dA_{ij}(x) = \int_{y=0}^{\infty}dC_i(y)\int_{t=0}^{\min(y,x)}\mu e^{-\mu t}dY_i(x-t) \cdot\binom{N-i}{j-i+1}(1-e^{-\lambda x})^{j-i+1}(e^{-\lambda x})^{N-j-1}$$

$$+\int_{y=0}^{x}dC_i(y)e^{-\mu y}dY_i(x-y) \cdot\binom{N-i}{j-i}(1-e^{-\lambda x})(e^{-\lambda x})^{N-j} \qquad j\geq i,\ x\geq 0$$

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

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

Then from above we have, after some simplifications,

$$\alpha_{i,i-1}(\theta) = \mu\sigma_i[\theta+\lambda(N-i)] \left\{\frac{1-\gamma_i[\theta+\mu+\lambda(N-i)]}{\theta+\mu+\lambda(N-i)}\right\}$$

and

$$\alpha_{ij}(\theta) = \binom{N-i}{j-i+1} \sum_{k=0}^{j-i+1} (-1)^k \binom{j-i+1}{k} \mu\sigma_i[\theta+(N-j+k-1)\,]$$

$$\cdot \left\{\frac{1-\gamma_i[\theta+\mu+(N-j+k-1)\lambda]}{\theta+\mu+(N-j+k-1)\lambda}\right\}$$

$$+\binom{N-i}{j-i+1} \sum_{k=0}^{j-i} (-1)^k \binom{j-i}{k} \sigma_i[\theta+(N-j+k)\lambda]$$

$$\cdot \gamma_i[\theta+\mu+(N-j+k)\lambda] \qquad j\geq i$$

constituting the LS transforms of the transition probabilities for the subdiagonal and general terms respectively.

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

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

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

$$A = \begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} & \cdots & \alpha_{0N} \\ \alpha_{10} & \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1N} \\ \alpha_{20} & \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2N} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \alpha_{N0} & \alpha_{N1} & \alpha_{N2} & \cdots & \alpha_{NN} \end{bmatrix}$$

which we partition accordingly

$$A = \left[\begin{array}{c|ccc} \alpha_{00} & \alpha_{01} & \alpha_{02} & \cdots & \alpha_{0N} \\ \hline \alpha_{10} & & & & \\ \alpha_{20} & & & & \\ \vdots & & H & & \\ \alpha_{N0} & & & & \end{array}\right]$$

From the theory of finite Markov chains [10], we know that the $(i,j)^{\underline{th}}$ 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 ($\bar{\delta}_j$ in swapping and v_j in processing) before the subsequent task completion. Thus we have

$$(I-H)^{-1}\ (\underline{\eta}-\underline{\delta}) = \underline{v}$$

where $\underline{\eta}$ is the vector of expected task completion times (determined by the values assumed by Q_i and D_i) and $\underline{v}$ is the vector of expected processing times. The elements of the vector $\underline{v}$ are therefore obtained from the relation

$$\underline{\eta}-\underline{\delta} = (I-H)\underline{v}.$$

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

$$\underline{\delta} = (I-H)\underline{b} \quad .$$

Note that the elements of $\underline{v}$ and $\underline{b}$ corresponds to the number of programs seeking control of the CPU at the initiation of a busy period. Extending our previous statement regarding effective utilization of the CPU to a vector, we have

$$U_i = v_i/(v_i+b_i) \qquad i=1,2,\ldots,N.$$

COMPUTATIONAL RESULTS

For computational comparisons we assume a system with ten peripheral devices accessing a single CPU. The mean access rate (λ) is .005/ms and the mean processing rate (μ) is .05/ms. Beginning with a constant swap-time of 10 ms and constant quantum of 150 ms, we derive the expected CPU utilization to range from 63.968 to 66.654 (all values given as percent utilization). Beginning with values near 65 percent seems warranted with respect to current systems although these values can vary widely.

The effect of choosing different constant quantum values is shown in Figure 1. By decreasing the quantum from 150 ms to 50 ms, a drop of approximately two percent effective CPU utilization is experienced. However, by increasing the quantum from 150 ms to 250 ms, no practical improvement is realized. To prefer 50 ms to 150 ms for an operational value requires some considerations of the user perspective, but that no advantage resides with the choice of 250 ms over 150 ms is obvious.

By implementing dynamic allocation of quantum length, some interesting results can be observed. Figure 2 reveals that by varying the quantum value directly with the number of tasks in queue, we incur almost a two percent drop in effective CPU utilization assuming initially that the busy period begins with a single program in queue. This disadvantage is removed as the number of programs initially seeking CPU control increases, i.e. for heavy demand environments, no advantage is apparent in increasing the quantum with an increasing demand. However, by adopting the strategy of reducing the quantum with increasing demand we effect an increase in effective CPU utilization. Although

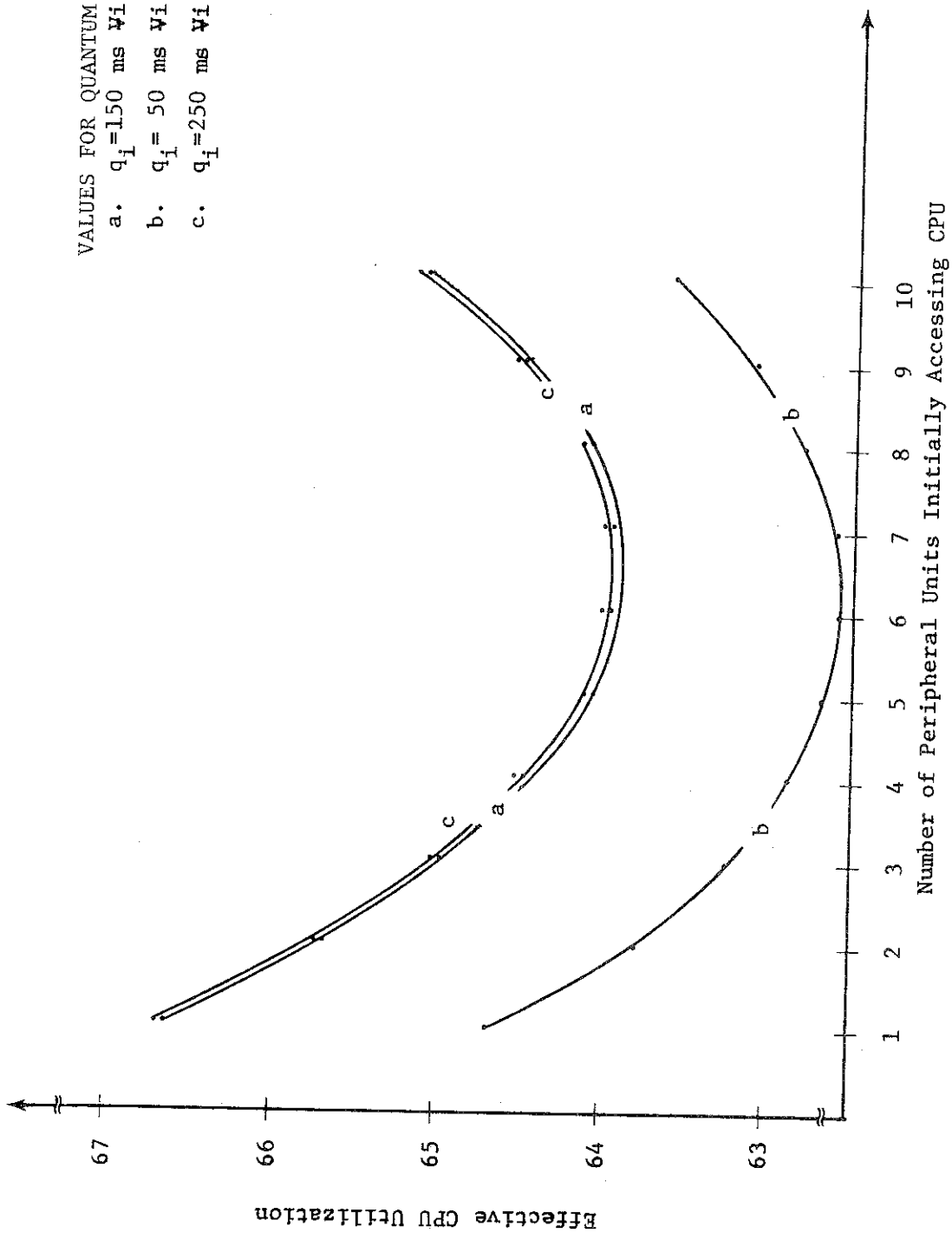

FIGURE 1. EFFECT OF CHOICES FOR CONSTANT QUANTUM WITH SWAP-TIME CONSTANT (δ_i = 10 ms).

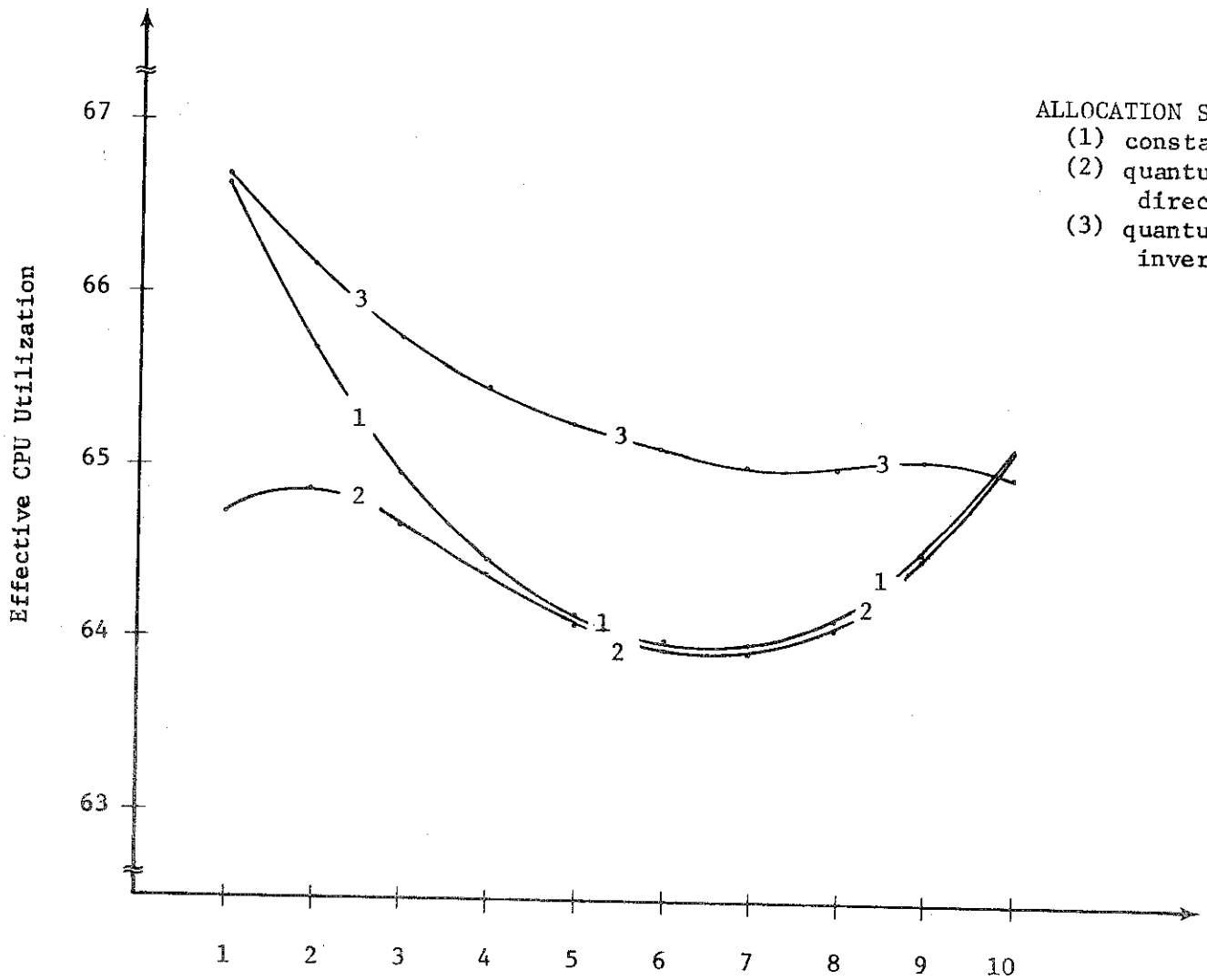

FIGURE 2. COMPARISON OF QUANTUM ALLOCATION STRATEGIES

small, this increase is interesting since the strategy of reducing quantum length with increasing demand is also beneficial to the user. In this instance at least the user and operator perspectives seem not to be competitive. Note that the average quantum length is the same in all three cases.

Further investigation of the results presented in Figure 2 is aided by reference to Table 1. The components that determine effective CPU utilization are shown in Table 1, and although the increase in expected swap-time with reduced quantum length is evident, a more than compensating increase in expected processing time is realized.

Figure 3 shows the pronounced effect of varying swap-time. In one instance, swap-time increases with the increase in demand, possibly reflecting the use of more complex scheduling algorithms with increased load. In the second case the swap-time decreases with demand, indicative of a contrary strategy -- employing simpler scheduling algorithms with an increasing load. With either strategy an effect is induced. Note that the average swap-time is the same in both cases (10 ms) and equal to the value in the constant case (designated as c). The intersection of the three curves at a single point is interesting, but we can offer no obvious interpretation of this behavior. It might suggest that this is an inherently stable point of behavior, i.e. the value of effective CPU utilization which is insensitive to variations in swap-time strategies.

Numerical results for the fourth model, the general case where both swap-time and quantum length vary, offer little new information. The dominance of the swap-time is confirmed, and the dynamic allocation of quantum exerts only a minor effect. Again, the allocation strategy for both swap-time and quantum is constructed to produce average values of 10 ms and 150 ms respectively.

Number of Tasks in Queue Initiating the Busy period	Constant Quantum (1: q_i=150 ∀i)			Varying Quantum (2: Increasing)				Varying Quantum (3: Decreasing)			
	Expected Swap Time	Expected Processing Time	Effective CPU Utilization	Quantum Allocation (ms)	Expected Swap Time	Expected Processing Time	Effective CPU Utilization	Quantum Allocation (ms)	Expected Swap Time	Expected Processing Time	Effective CPU Utilization
1	7.512	15.015	66.65	50	7.372	13.535	64.74	250	8.563	17.126	66.67
2	10.405	19.931	65.70	75	10.246	18.936	64.89	225	11.725	22.936	66.17
3	11.956	22.177	64.97	100	11.833	21.662	64.67	200	13.347	25.663	65.79
4	13.133	23.854	64.49	125	13.051	23.607	64.40	175	14.580	27.669	65.51
5	14.340	25.698	64.18	150	14.289	25.582	64.16	150	15.882	29.886	65.30
6	15.820	28.129	64.00	150	15.782	28.060	64.00	150	17.496	32.710	65.15
7	17.824	31.643	63.97	175	17.798	31.607	63.98	125	19.696	36.683	65.07
8	20.784	37.152	64.13	200	20.764	37.135	64.14	100	22.296	42.685	65.06
9	25.708	46.774	64.53	225	25.692	46.770	64.54	75	28.145	52.511	65.10
10	35.708	66.763	65.15	250	35.692	66.770	65.17	50	37.701	70.043	65.01

Table 1. Individual Components of Effective CPU Utilization for Three Quantum Allocation Strategies.

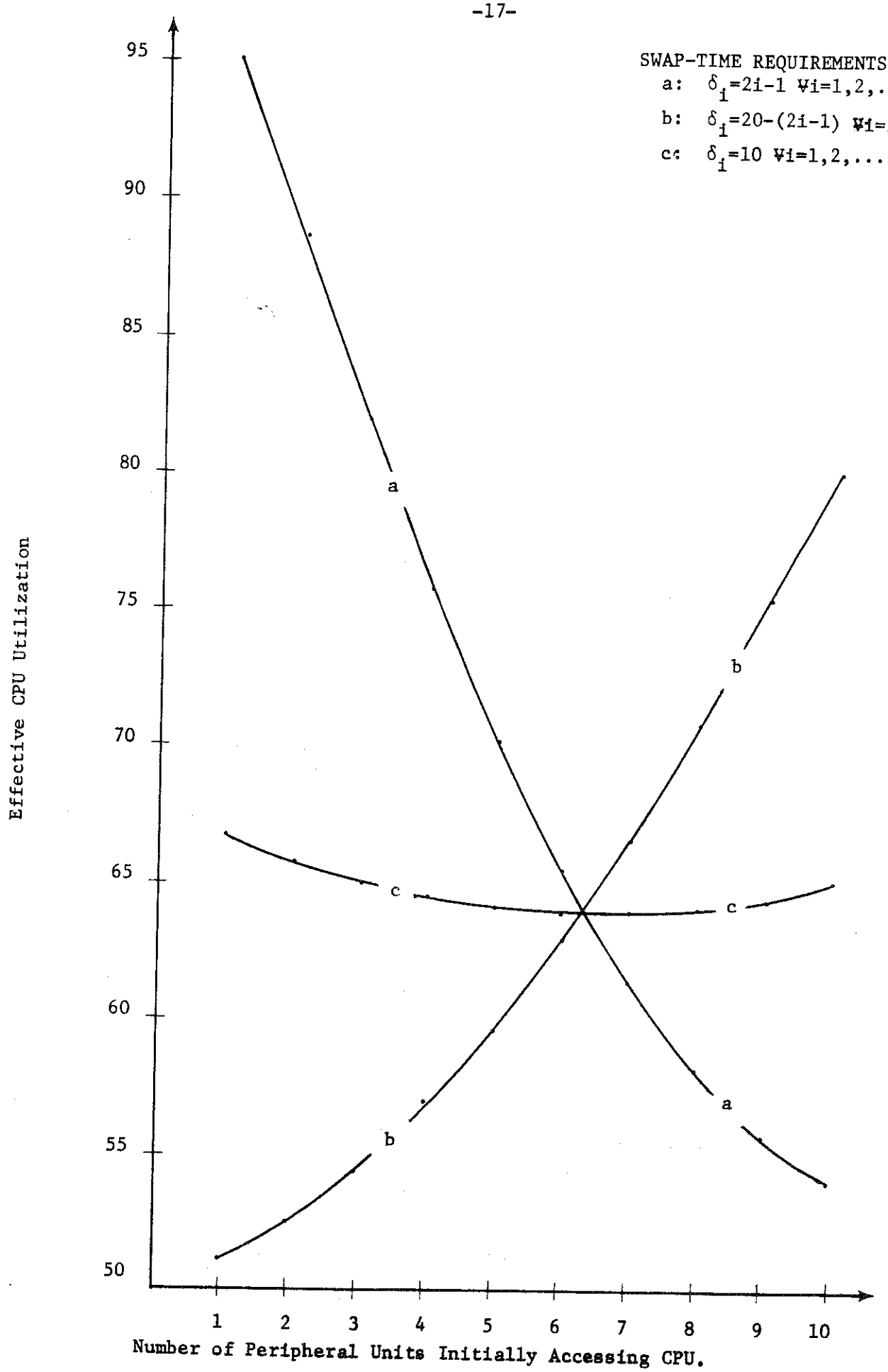

FIGURE 3. PRONOUNCED EFFECT OF VARYING SWAP-TIME WITH A FIXED QUANTUM (q_i=105 ms)

CONCLUSIONS AND SUMMARY

Our investigation of dynamic quantum allocation and swap-time variability has led to the following conclusions, all of which are limited to the conditions specified for computational analysis:

(1) With a fixed quantum strategy, at some point an increase in quantum length results in no practical improvement in the effective CPU utilization.

(2) With a fixed swap-time, an improvement in effective CPU utilization can be realized by a strategy of decreasing quantum length with increasing demand.

(3) The effect of swap-time variability on the effective CPU utilization is pronounced.

(4) While the dynamic allocation of quantum seems attractive in concept, it seems to offer only a minor advantage in terms of effective CPU utilization.

The fourth conclusion must be accepted only with respect to the operator perspective as reflected by the effective CPU utilization. For the user the advantage might be considerable.

The models developed during this research offer tools that are both powerful and practical. Specific system configurations can be conveniently investigated using the FORTRAN programs that produced the computational results. These programs are also being used to investigate further strategies involving both dynamic quantum allocation and swap-time variability. Extensions of this research are directed toward a more comprehensive sensitivity analysis and the identification of "best" strategies.

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

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

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

5. 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.

6. Coffman, Edward G., Jr. and Peter J. Denning, Operating Systems Theory, Prentice-Hall, to appear.

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

8. 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.

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

10. Kemeny, J. G. and J. L. Snell, Finite Markov Chains, D. Van Nostrand, Princeton, New Jersey, 1960.

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

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

13. 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.

14. 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.

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

16. 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.

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

16. 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.