THE APPLICATION OF OPTIMAL STOCHASTIC CONTROL THEORY IN COMPUTER SYSTEM LOAD REGULATION

by

Samuel T. Chanson & Raymond Lo

Technical Report 81-5

June 1981

The application of Optimal Stochastic Control Theory in computer system load regulation [*]

by

Samuel T. Chanson & Raymond Lo

ABSTRACT

A method using some results and techniques of Optimal Stochastic Control Theory is introduced to compute the optimal admission policy for paged batch-interactive computer systems. The admission policy determines the optimal number of batch and terminal jobs that should be activated at each system state to maximize throughput. The system state is defined as the vector (N1,N2) where N1 and N2 are respectively the total number of terminal and batch jobs in the system. Thus the policy is adaptive to workload variation. As well, the quality of service given to each class of jobs (specifically their mean response times) can be adjusted by choosing a suitable weight for the terminal jobs. A large weight reduces the mean response time of the terminal jobs at the expense of the mean batch response time while maintaining the total system throughput at its maximum level.

Unlike most existing adaptive control algorithms, the approach is based on mathematical modelling and its extension to cover the case of more than two classes of jobs is straightforward.

[*] This work was supported by the National Science and Engineering Research Council of Canada under Grant A3554.

I. Introduction

Load control is important in paged virtual memory systems. One reason is the fact that if the workload is allowed to increase unregulated, a point will be reached when 'thrashing' (and thus reduced throughput) will occur [1]. Furthermore, it has been shown that for such systems, an optimal degree of multiprogramming exists which maximizes the system throughput rate [3]. The optimal value, however, changes as the workload characteristics vary. A good load control policy should therefore dynamically adjust to the changing workload condition to optimize system performance at all times. A number of adaptive algorithms have been proposed (see for example [2-13]). Typically, they work by regulating the load to keep some measures related to program behaviour (usually the paging behaviour) to within some predetermined bounds. Generally the bounds are set to hopefully allow the highest possible load without saturating the system.

Though these algorithms do improve system performance (usually the throughput rate), they are basically heuristics and therefore do not represent a systematic approach to tackle the problem of system load control. Furthermore, most large scale virtual memory systems nowadays support both batch and interactive jobs. For such systems, one is interested not only to maximize the system throughput rate but also to guarantee good response times to the interactive users (possibly at the expense of the batch turnaround times). Landwehr [5] proposed a scheme to activate batch jobs based on the terminal load. The aim was to maintain good response to interactive requests by activating less batch jobs when the terminal load is heavy while ensuring a minimal level of batch throughput. There was, however, no attempt to prevent the system from becoming saturated or to optimize performance. As well, there do not appear to be a simple or systematic way of determing the values of the break points. Hine et al.[14] studied the problem from a slightly different viewpoint. Their goal was to provide different response times to each class of jobs (batch and interactive) while maximizing the CPU utilization. They employed a mathematical model but optimization was achieved by an exhaustive search technique. A heuristic was also given which gives good but not necessarily optimal results.

In this paper, we propose to use optimal stochastic control theory to compute the optimal admission policy for paged batch - interactive computer systems. The policy determines the number of batch <u>and</u> interactive jobs that should be activated at each system state. The admission policy minimizes the mean weighted sum of the number of batch and interactive jobs in the system. We shall show that this will maximize the total system throughput. By adjusting the weight w in the weighted sum, one can easily control the quality of service (i.e., the response time) given to the two classes of jobs while maintaining the system throughput at its maximum level. Another advantage of this mathematical modeling approach is its generality. Once a stochastic model of a system has been developed, the theory can be used to determine optimal policies pertaining to different criteria simply by using different objective functions in the model. We now proceed to give a brief review of some relevent results from optimal stochastic control theory. The reader is referred to [15] for a more thorough treatment.

II. Review of some results of Optimal StochasticControl Theory

Let $\underline{S} = \{1,...n\}$ denotes the state space of a system. To each state $i \varepsilon \underline{S}$ and each control u in the finite control space $\underline{C}$ there corresponds a set of transition probabilities $P_{ij}(u)$, $j=1,...n$, where P_{ij} denotes the probability that the next state will be j given that the current state is i and control u is applied. Each time the system is in state $i \varepsilon \underline{S}$ and control u is applied, an expected cost $g(i,u)$ is incurred. The objective is to minimize over all admissible policies $\pi = \{u_0,u_1,...\}$ with $u_k : \underline{S} \dashrightarrow \underline{C}$, $u_k(i) \varepsilon U(i)$, $\forall i \varepsilon \underline{S}$, the average cost per stage

$$J_\pi(x_o) = \lim_{N\to\infty}(1/N)E\{\sum_{k=0}^{N-1} g[x_k,U_k(x_k)]\} \qquad ....(1)$$

for any given initial state $x_0 \varepsilon \underline{S}$. Where x_k is the state of the system at time k, $u_k(x_k)$ is the control at time k conditioning on x_k and U(i) is the set of all possible controls for state i.

Given any stationary admissible policy $\pi = \{u,u,...\}$. Let us denote by $\underline{P}_u$ the transition probability matrix having elements $P_{ij}(u(i))$:

$$\underline{P}_u = \begin{bmatrix} P_{11}(u(1)) & \cdots\cdots & P_{1n}(u(1)) \\ \vdots & & \vdots \\ P_{n1}(u(n)) & \cdots\cdots & P_{nn}(u(n)) \end{bmatrix}$$

By the definition of P_{ij}, we have

$$P_{ij}(u(i)) \geq 0, \forall i,j; \qquad \sum_{j=1}^{n} P_{ij}(u(i)) = 1, \forall i.$$

Let us now consider the value of the cost function $J_\pi(x_0)$ of Equation (1). We may use the notation:

$$J_\pi(i) = J_u(i),\ i=1,2\ldots n$$

for stationary policies $\pi = \{u,u,\ldots\}$. Denote

$$\underline{J}_u = \begin{bmatrix} J_u(1) \\ J_u(2) \\ \vdots \\ J_u(n) \end{bmatrix}, \qquad \underline{G}_u = \begin{bmatrix} g(1,u(1)) \\ g(2,u(2)) \\ \vdots \\ g(n,u(n)) \end{bmatrix}$$

With this notation it is easy to see that

$$\underline{J}_u = \lim_{N\to\infty} (1/N) \sum_{k=0}^{N-1} \underline{P}_u^k \underline{G}_u \qquad \ldots\ldots\ldots(2)$$

where $\underline{P}_u^k$ ($\underline{P}_u$ raised to the kth power) is the k-step transition probabilities corresponding to a stationary policy $\pi = \{u,u,..\}$.

It has been shown [15] that if there is a state $t \in \underline{S}$ such that for every state $i \in \underline{S}$ the probability of reaching state t from state i is strictly positive, then an optimal stationary policy exists. Furthemore, if $\pi = \{u,u,\ldots\}$ is an admissible stationary policy, then there exists a constant $\emptyset_u$ and a function $h_u : S \to R$ such that

$$J_u(i) = \emptyset_u, \quad i = 1,2,\ldots n \quad \text{and}$$

$$\emptyset_u + h_u(i) = g(i,u(i)) + \sum_{j=1}^{n} P_{ij}(u(i))h_u(j), i=1,2,\ldots n \qquad \ldots\ldots\ldots(3)$$

Equation(3) represents a system of n linear equations with (n+1) unknowns - the scalars $\emptyset_u, h_u(1), h_u(2), \ldots h_u(n)$. We may add one additional equation to this system by requiring that

$$h_u(t) = 0. \qquad \ldots\ldots\ldots(4)$$

$h_u(t)$ then becomes the base on which the other $h_u(i)$'s are computed. Since a unique solution exists for this system [15], it forms the basis for solving the average cost problem. We shall call the method of this solution the <u>Policy Improvement Algorithm</u> which is an iterative minimization process.

Let $\pi^k = \{u^k, u^k, \ldots\}$ be an admissible stationary policy obtained at the kth iteration of the algorithm. We determine the average cost per state $\emptyset_u$ corresponding to π^k by solving the system of (n+1) equations:

$$\emptyset_u^k + h_u^k(i) = g(i,u^k(i)) + \sum_{j=1}^{n} P_{ij}(u^k(i))h_u^k(j), i=1,2,\ldots n$$

$$h_u^k(t) = 0 \qquad \ldots\ldots\ldots(5)$$

Subsequently, we find a policy $\pi^{k+1}=\{u^{k+1},u^{k+1},\ldots\}$ where $u^{k+1}(i)$ is such that

$$g(i,u^{k+1}(i)) + \sum_{j=1}^{n} P_{ij}(u^{k+1}(i))h_u^k(j)$$

$$= \min_{u\varepsilon U(i)} \{g(i,u) + \sum_{j=1}^{n} P_{ij}(u)h_u^k(j)\}, \quad i = 1,2,\ldots,n \quad ..(6)$$

The average cost per stage ϕ_u^{k+1} for the (k+1)th iteration is obtained by solving Equation (5) with the superscript k replaced by k+1. It can be shown that $\phi_u^{k+1} \leq \phi_u^k$. The iteration is repeated until $(\phi_u^k - \phi_u^{k+1})$ is 0 (or in practice, less than some arbitrarily small constant). The policy $\pi^{k+1} =\{u^{k+1},u^{k+1},\ldots\}$ is optimal.

III. Model description and performance optimization

Consider a combined batch-interactive system as in Figure 1. There are N active terminals in the system and batch jobs arrive at a mean rate of ϕ_B. Upon arrival to the system, the jobs (both batch and interactive) enter the memory queues and await admission into the memory loop. An adaptive load control algorithm determines n1 and n2, the number of interactive and batch jobs respectively, to be admitted into the memory loop at each system state. The system state in turn is defined by the vector (N1,N2) where N1 and N2 are the number of interactive and batch jobs respectively in the system (i.e., those in the memory loop plus those waiting in the memory queues). The interactive user think time is assumed to be an exponentially distributed random variable with mean $1/\phi_T$. Therefore the mean arrival rate of the interactive jobs is $(N - N1)\cdot\phi_T$ when the system state is (N1,N2). For simplicity, only three service centers are considered within the memory loop. They are the CPU, drum and file disk service centers.

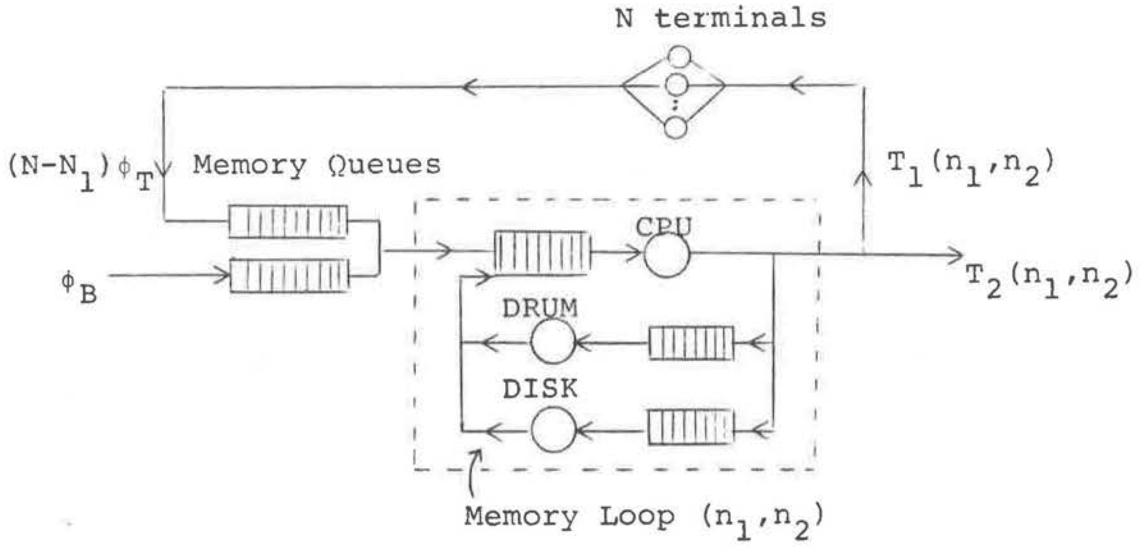

Figure 1. <u>The System Model</u>

Our problem is to find an optimal admission policy π^* such that the system throughput rate is maximized and the mean response time for interactive jobs can be controlled. The simplest way to do this is to minimize the expected weighted sum of the number of jobs in the system (i.e., E(w*N1+N2)) at each system state. (Notice N1 and N2 are the <u>total</u> number of jobs in the system including those waiting in the memory queues). The weight w reflects the importance of the interactive job relative to the batch job and is normally greater than 1. Consider the simple case when w equals 1 which implies the mean total number of jobs in the system is to be minimized. For a given job arrival rate, Little's Law says that minimizing the mean number of jobs will also minimize the mean wait time of the jobs in the system. Thus the mean system throughput rate is maximized. As w is increased, N1 will be further reduced by admitting the interactive jobs sooner thus cutting down their memory queue wait time and hence the interactive response time. This however will result in the batch jobs having to wait longer to be admitted in order to maintain the degree of multiprogramming at the optimal value. Thus the mean response time of batch jobs suffers but the mean system throughput rate remains constant at its maximum level. The above argument is validated by the results reported in Section V.

An admission policy determines the number of interactive and batch jobs (n1 and n2 respectively) to be admitted into the memory loop at each system state. Thus the optimal admission policy is a mapping of (N1,N2) to (n1,n2) i.e.,

$$\pi^* = (N_1,N_2) \rightarrow (n_1,n_2) \quad \text{such that}$$
$$z = \lim_{N\rightarrow\infty} (1/N)\ E\ \{ \sum_{k=0}^{N-1} g(x_k) \}$$

is minimized, where x_k = (N1,N2) is the state of the system at time k and the cost function $g(x_k)$ is (w*N1+N2). This problem fits into the framework of the basic problem of optimal stochastic control and hence can be solved by the techniques described in Section II.

Because of the blocking which occurs when a job cannot be allocated main memory, the model in Figure 1 cannot be solved exactly. We use the decomposition technique [16] to obtain an approximate solution. The method is to solve the memory loop as an independent closed central-server network, then replace it by a single composite server whose service rate is the same as the throughput rate of the memory loop. As the state-transition rate within the memory loop greatly exceeds the interactions between it and the rest of the system, the error introduced by the approximation should be small [16].

3.1 The memory loop as a closed central-server network

Our task is to compute the throughput rates of the interactive and batch jobs when there are n1 interactive and n2 batch jobs in the memory loop (denoted as $T_1(n1,n2)$ and $T_2(n1,n2)$ respectively). We make the usual assumptions to simplify the computation:

i) The virtual time between successive page-faults and file disk requests for class r jobs are exponentially distributed with means $L_r(p)$ and $1/d_r$ respectively, where

$$L_r(p) = \frac{2b_r}{1 + (a_r/p)^2} \qquad \text{..........(7)}$$

is the expected lifetime for class r jobs when the main memory allocation is p pages. The lifetime function in Equation (7) was first proposed by Chamberlin et al.[17] (Figure 2) where b is the expected virtual inter page-fault time when a class r job is allocated a_r pages of main memory and a_r is the number of pages that provides the job with half of its longest possible lifetime.

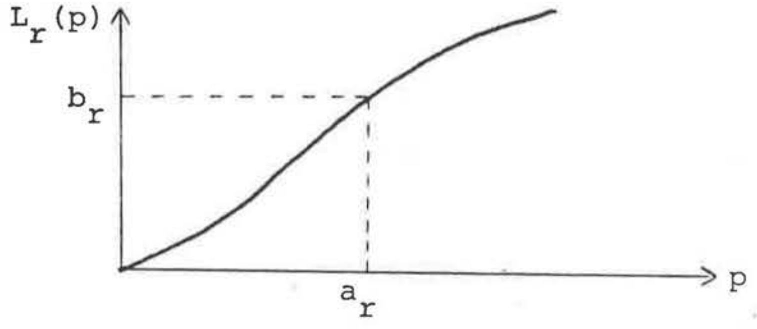

Figure 2. <u>The Lifetime Function</u>

ii) The CPU time request for class r jobs is exponentially distributed with mean $1/C_r$.

iii) The mean drum and disk service rates (μ_2 and μ_3 respectively) are the same for all classes of jobs.

iv) The service discipline of the CPU, for which the service rate μ_{1r} is class-dependent, is processor sharing.

From the above assumptions, it is easy to see that

$$\mu_{1r} = d_r + C_r + 1/L_r(p), \quad r = 1,2 \qquad \cdots\cdots (8)$$

Also, the transition probabilities, $q_{ij}(r)$, of a class r job going from server i to server j are given by:

$$q_{12}(r) = 1/(\mu_{1r}L_r(p_r))$$

$$q_{13}(r) = d_r/\mu_{1r} \qquad \cdots\cdots\cdots (9)$$

$$q_{31}(r) = q_{21}(r) = 1$$

The state of the model is defined by the number of jobs in each class at each server. Thus if k_{ir} is the number of class r jobs at server i, the state $\underline{K}$ of our model is given by the vector $(k_{11}, k_{12}, k_{21}, k_{22}, k_{31}, k_{32})$. Following Baskett et al.[18], the stationary probability $P_{\underline{n}}(\underline{K})$ of state $\underline{K}$ when there are n1 interactive and n2 batch jobs in the system ($\underline{n}$ is the vector (n1,n2)) is given by:

$$P_{\underline{n}}(\underline{K}) = \frac{1}{G(\underline{n})} k_1!k_2!k_3! \prod_{r=1}^{2} \left\{ \frac{X_{1r}^{k_{1r}}}{k_{1r}!} \cdot \frac{X_{2r}^{k_{2r}}}{k_{2r}!} \cdot \frac{X_{3r}^{k_{3r}}}{k_{3r}!} \right\} \quad \cdot\cdot(10)$$

the x_{ir} 's are the solutions of the transition equations:

$$\sum_{i=1}^{3} q_{ij}(r) U_{ir} X_{ir} = U_{jr}.X_{jr}, \qquad \begin{matrix} j = 1,2,3 \\ r = 1,2 \end{matrix} \quad \cdots\cdots(11)$$

and $G(\underline{n})$ is the normalizing constant which normalizes the sum of all the probabilities to 1. It is easy to see that

$$\begin{aligned} X_{1r} &= 1 \\ X_{2r} &= \frac{q_{12}(r) \cdot \mu_{1r}}{\mu_2} = \frac{1}{\mu_2 L_r(P_r)} \qquad \cdots\cdots\cdot(12) \\ X_{3r} &= \frac{q_{13}(r) \cdot \mu_{1r}}{\mu_3} = \frac{d_r}{\mu_3} \end{aligned}$$

Assuming the memory is equally partitioned among all the jobs, the mean value of P_r is Y/(n1+n2) where Y is the capacity of the main memory. Extending the results of Buzen [19] to the multi-class case, an efficient method for computing $G(\underline{n})$ was derived.

We define an auxiliary function

$$g(n_1,n_2,m) = \sum_{k \varepsilon S(n_1,n_2,m)} \prod_{i=1}^{m} (X_{i1})^{k_{i1}} \cdot (X_{i2})^{k_{i2}} \cdot \frac{(k_{i1}+k_{i2})!}{k_{i1}!k_{i2}!} \quad \cdot\cdot(13)$$

$$m = 1,2,\ldots,M,$$

where M is the total number of service centers in the system (which is 3 in our case). It can be shown [20] that

$$G(\underline{n}) = g(n1,n2,M).$$

The recursive relationship

$$g(n_1,n_2,m) = \sum_{p=0}^{n_1} \sum_{q=0}^{n_2} \left\{ \frac{(p+q)!}{p!q!} (X_{m1})^p \cdot (X_{m2})^q \cdot g(n_1-p,n_2-q,m-1) \right\} \quad \cdot\cdot(14)$$

together with the initial condition

$$g(n_1,n_2,1) = (X_{11})^{n_1} \cdot (X_{12})^{n_2} \cdot \frac{(n_1+n_2)!}{n_1!n_2!} \quad \cdots\cdots(15)$$

can be used to compute $G(\underline{n})$ efficiently (see [20] for details).

The utilization of server m when the memory loop is in state $\underline{n}$ can be shown to be equal to

$$u_n(m) = 1 - \frac{g(n_1,n_2,m-1)}{g(n_1,n_2,m)} \quad \cdots\cdots\cdot(16)$$

and the partial utilization of server m for class r jobs is given by

$$u_{\underline{n}}(m,r) = \frac{1}{g(n_1,n_2,m)} \sum_{p=1}^{n_1} \sum_{q=0}^{n_2} \left\{ \frac{p}{p+q} \cdot \frac{(p+q)!}{p!q!} \cdot \right.$$

$$\left. (X_{m1})^p \cdot (X_{m2})^q \cdot g(n_1-p,n_2-q,m-1) \right\} \quad \cdots\cdots(17)$$

The throughput rate of class r jobs in the memory loop $T_r(n1,n2)$ can now be computed from Equations (17), (15) and (14) using the familiar relationship:

$$T_r(n_1,n_2) = u_{\underline{n}}(1,r) \cdot (1-q_{12}(r) - q_{13}(r)) \cdot \mu_{1r} \qquad \ldots\ldots(18)$$

3.2 Formulation of the optimal control problem

The memory loop in Figure 1 is replaced by an aggregate server (Figure 3) whose service rate is workload dependent and is equal to T (n1,n2) when there are n1 class 1 and n2 class 2 jobs in the memory loop (Equation (18)).

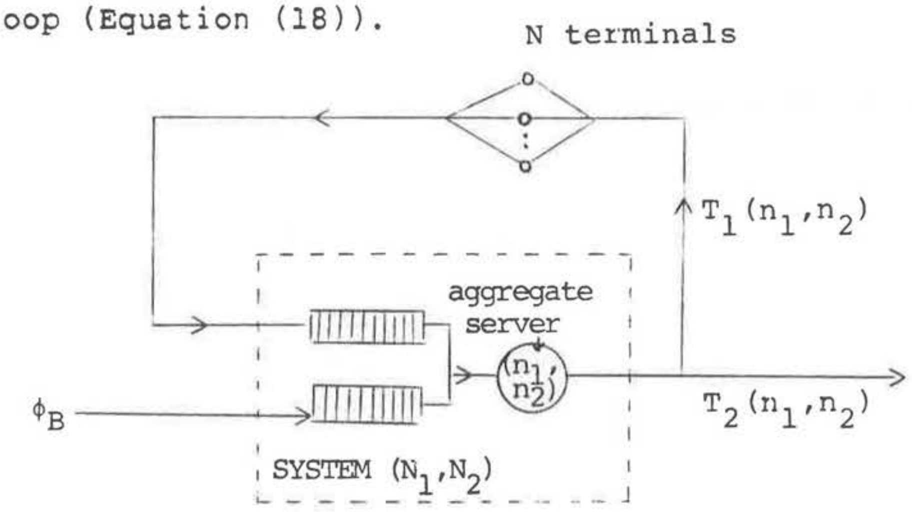

Figure 3. The system model with the memory loop replaced by an aggregate server.

In a small interval h, the transition probability $P_{\underline{ij}}(\underline{n})$ of the system going from state $\underline{i}$ to state $\underline{j}$ when the load of the memory loop is $\underline{n}$ is given by equation:

$$P_{\underline{ij}}(\underline{n}) = \begin{cases} \phi_B h, & \text{for } \underline{i} = (N_1,N_2),\ \underline{j} = (N_1,N_2+1),\ N_2 \le m_2-1 \\ (N-N_1)\phi_T h, & \text{for } \underline{i} = (N_1,N_2),\ \underline{j} = (N_1+1,N_2) \\ T_1(\underline{n})h, & \text{for } \underline{i} = (N_1,N_2),\ \underline{j} = (N_1-1,N_2),\ N_1 > 0 \\ T_2(\underline{n})h, & \text{for } \underline{i} = (N_1,N_2),\ \underline{j} = (N_1,N_2-1),\ N_2 > 0 \\ 1 - (\text{sum of the above 4 cases}) & \text{for } \underline{i} = \underline{j} \\ 0 & \text{otherwise} \end{cases} \qquad \cdots\cdots (19)$$

If the control decision is made each time a job arrives or departs from the system, the system state vector (N1,N2) can be described by a two dimensional semi-Markov decision process. The set of possible control for each state $\underline{i}$=(N1,N2), is {$\underline{n}$=(n1,n2), 0 <= n1 <= N1, 0 <= n2 <= N2}. It is easy to see that the transition probability, $P_{\underline{ij}}(\underline{n})$, of going from state $\underline{i}$ to state $\underline{j}$ when control $\underline{n}$ is applied is the same as those given in Equation (19).

The problem of finding an optimal admission policy is formulated as a minimization problem of the average cost per unit time over all admissible policies

$$\min\{J_\pi = \lim_{N\to\infty} (1/N) \sum_{k=0}^{N-1} P_\pi^k\, g_\pi\}$$

where P_π is the transition probability matrix having element $P_{\underline{ij}}$, and

$$g_\pi = \begin{bmatrix} g(0,0) \\ g(1,0) \\ \vdots \\ g(N_1,N_2) \\ \vdots \\ g(N,m_2) \end{bmatrix}, \qquad g(N_1,N_2) = w * N_1 + N_2$$

The optimal admission policy is solved by the Policy Improvement Method described in Section II.

IV. Interpreting the results

Using the model as described in the previous section, the values of certain performance indices under a given admission policy $\pi = \{u, u, \ldots u\}$ can be computed by using an appropriate g function in Equation (5). The performance indices chosen are the mean response times and throughput rates for each class of jobs.

To compute the mean response time R for interactive jobs, we first compute the mean number of interactive jobs Z1 in the system by using g(i,u(i)) = N1 in Equation (5) and then apply Little's formula to obtain :

$$R_1 = \frac{Z_1}{\phi_T(N_1 - Z_1)}$$

Similarly, the mean turnaround time for batch jobs is given by:

$$R_2 = \frac{Z_2}{\phi_B}$$

where Z2 is the mean number of batch jobs in the system and is computed by using g(i,u(i)) = N2 in Equation (5). The mean throughput rates for interactive and batch jobs are computed by substituting T_1(n1,n2) and T_2(n1,n2) for g(i,u(i)) respectively in Equation (5).

4.1 The efficts of the weight w on performance

Since the optimal admission policy minimizes the mean _weighted_ sum of the number of jobs in each class in the system, the effects of w on the performance indices deserve examination.

Figures 5(a) through 8(a) are some samples plots of the mean response times for each class of jobs versus the weight (w) under different workloads with and without load control. The lower-bound mean response time for interactive jobs (i.e., when the batch stream is absent and the optimal control is applied) is also plotted on each of these graphs.

Figures 5(b) to 8(b) are the corresponding throughput plots with the total system throughput (sum of batch and interactive throughtputs) also given. The different workload conditions are simulated by varying the CPU requirements for each class of jobs systematically over a range of values and keeping the other workload parameters constant. This is justified on the ground that changing the other parameter values has the same net effect as far as affecting the residence times of the jobs in the system (and thus the system throughput rate) is concerned.

These figures show that the optimal admission policy improves the mean response times and throughputs for each class of jobs considerably over those with no control. For a given system hardware configuration and characteristics, the exact improvements depend on the workload of the system. Generally, the heavier the workload the greater the improvements. When the system load is light the improvement is only marginal. This leads to the logical conclusion that there is no need to control the system when the expected workload is light.

From the figures, the following interesting effects of w on the performance indices can be observed:

i) An increased value of w increases the interactive throughput and decreases the batch throughput. It also decreases the mean interactive response time and increases the batch turn-around time. In other words, the larger the value of w the better the quality of service is given to the interactive jobs.

ii) When w exceeds a certain value, the rates of change of the values of the performance indices with respect to w decrease as w increases and become insignificant for large w. This is an important property because it implies that the batch throughput will not continue to drop as w increases but is guaranteed to exceed some minimum level.

iii) The total system throughput does not vary appreciably as w varies. Recall that when $w=1$, the optimal admission policy minimizes the total number of jobs in the system which is equivalent to maximizing the total system throughput. This property implies that the optimal admission policy always produces maximum or near-maximum total system throughput regardless of the value of the weight chosen.

iv) The mean interactive response time becomes very close to the lower-bound when w exceeds some value (denoted by w*). The value of w* depends on the relative loads of interactive and batch jobs. The heavier the interactive load relative to the batch load the greater the value of w*. This property suggests that we can choose a suitably large value of w such that the mean interactive response time is acceptably low (say within 10 percent of the lower bound).

From the above observations, we can see that an optimal admission policy can be obtained by choosing a suitable value of w which,

1) gives good response to the interactive jobs,
2) maximizes the total system throughput,
3) guarantees a minimum batch throughput.

Since the optimal admission policy improves the mean interactive response time considerably over that with no control (especially when the load is heavy) we expect that more terminals can be supported. To verify this we plot the interactive response times (with and without control) versus the number of terminals (Figure 9). It shows that the saturation point (according to Kleinrock's definition* [21]) when control is applied is about twice that with no control under the given workload.

----

* defined as the intersection of the mean normalized response time curve asymptote and the horizontal line corresponding to the mean response time when there is only one terminal.

4.2 The optimal admission policy

Tables (1) through (9) are some samples of optimal admission policies obtained by the Policy Improvement Method.

It is obvious that the weight (w) of the objective function affects the optimal admission policies. For a given set of parameter values, different weights in the objective function result in different optimal admission policies. Tables (1) to (3) are the policies obtained by using different weights ranging from 1 to 3 and keeping the other parameters constant. The other parameter values used are: $\phi_T=0.05$, $\phi_B=0.1$, Y=60 pages, N=6, $m_2=5$, $b_T=0.018$, $b_B=0.012$, $a_B=10$, $a_T=20$, $T_{IO}=20$, $B_{IO}=10$, $F_s=30$, $p_s=80$, cpuT=0.3, cpuB=0.3 (all time units are in seconds) where

$1/\phi_T$=	mean terminal "think" time
ϕ_B=	mean batch arrival rate
Y=	total memory size
N=	number of terminals
m_2=	maximum number of batch jobs to be considered
b_T, b_B, a_T, a_B=	parameters of the life time functions for batch and terminal jobs
T_{IO}=	mean interactive I/O request rate
B_{IO}=	mean batch I/O request rate
p_s=	mean paging rate
F_s=	mean I/O service rate
CPUT=	mean interactive CPU service rate

CPUB= mean batch CPU service rate

As expected these tables show that the larger the weight used in the objective function the higher the priority given to the interactive jobs. This is reflected by the fact that at some of the system states the policy with the larger weight admits more interactive jobs into the memory loop than those by policies with a smaller weight.

The policies in Tables (4) - (6) have the same set of parameter values and weight except for cpuB, which varies from 0.2 to 0.6. Varying cpuB and keeping the other parameters constant simulates different relative loads between interactive and batch jobs. These tables show that the lighter the load of one class of jobs relative to the other the better service it will receive. For example, as cpuB increases (hence the batch load decreases) the optimal policy admits more batch and less interactive jobs into the memory loop for the same system states.

Tables (7)-(10) are the optimal admission policies for different workloads which give mean interactive response time close to the lower bound. By comparing these policies it is observed that they are very similar to one another. Very often, when the number of interactive jobs in the system exceeds some number (3 in our numerical example) the policies will admit only that number of interactive jobs into the memory loop regardless of how many batch and interactive jobs there are in the system. This confirms the results of our earlier work [22] which solves the load control problem using queuing theory and optimization technique. From Table (11) it is observed that that number is equal to the number of interactive jobs which produces the highest partial interactive CPU utilization.

The similarity of the optimal admission policies would lead one to expect that a policy that produces optimal performance for a given workload to give close to optimal performance for a range of workload conditions (especially when w is large). If this is actually the case then in practice we need only to implement one standard policy for a range of workload instead of one for each workload condition. We choose the policy in Table (9), (which gives optimal performance at cpuT=0.4, cpuB=0.3) as the standard policy in our example. The performances of this policy for different workloads are computed and compared with the optimal values (a typical result is given in Table (12)). It is observed that the performances are quite close to the optimal performances in all cases.

V. Conclusion

We have described a method using some results and techniques of optimal stochastic control theory to compute the load control policy of a combined batch-interactive computer system. The policy determines the optimal number of batch and terminal jobs that should be admitted depending on the current state of the system. A mathematical model of the system was developed and the theory applied to compute the optimal admission policy. The objective function to be minimized per unit time is the mean weighted sum of the number of jobs in each class in the system (E(w*N1+N2)).

The policy thus obtained was shown to exhibit the following properties:
i) It gives good mean response time for interactive jobs (close to the lower bounds if sufficiently large weight w is used).
ii) It maximizes the total system throughput.
iii) It guarantees some minimum level of batch throughtput.
iv) A policy that produces an optimal performance for a given workload provides near-optimal performance for a range of different workload conditions, especially when w is large.
v) The quality of service given to each class of jobs can be easily controlled by choosing some suitable weight w.

The extension to cover more than two classes of jobs is straightforward.

REFERENCES

[1] Denning, P.,"Thrashing: its causes and prevention", Proc., AFIPS, Vol.33, 1968, 915-922.

[2] Badel, M., Gelenbe, E., Leroudier, J., Potier, D.,"Adaptive optimization of a timesharing system's performance", Proc. IEEE, Vol.63, 1975, 958-965.

[3] Badel, M., Leroudier, J., "Adaptive multiprogramming systems can exist", Performance of Computer Installations, D. Ferrari (ed.), North-Holland, 1978, 115-135.

[4] Denning, P., Kahn, K., Leroudier, J., Potier, D., Suri, R., "Optimal multiprogramming", Acta Informatica, Vol.7, No.2, 1976, 197-216.

[5] Landwehr, C.,"An endogenous priority model for load control in a combined batch-interactive computer system", Proc. Int'l Symp. On Comp. Perf. Modelling, Meas. and Eval., March 1976, 282-293.

[6] Leroudier, J. and Potier, D.,"Principles of optimality for multiprogramming", Proc., Int'l Symp. On Comp. Perf. Modelling, Meas. and Eval., March 1976, 211-218.

[7] Gelenbe, E., Kurinckx, A., Mitrani, I., "The rate control policy for virtual memory management", Operating Systems: Theory and Practice, D. Lanciaux (ed.), North-Holland, 1979, 247-264.

[8] Gelenbe, E. and Kurinckx, A., "Random injection control of multiprogramming in virtual memory", IEEE Trans. On Software Engineering, Vol.4, 1978,2-17.

[9] Bunt, R. and Hume, J.,"Adaptive Processor scheduling based on approximating demand distribution", INFOR, Vol.15,No.2, June 1977, 135-147.

[10] Geck, A., "Performance improvement by feedback control of the operating system", Proc. Of the 4th Int'l Symp. On Modelling and Perf. Eval. Of Computer Systems, Vienna, Feb. 1979, 459-471.

[11] Brandwajn, H., and Hernandez, J., "A study of a mechanism for controlling multiprogrammed memory in an interactive system", Perf. Of Computer Installations, D. Ferrari (ed.), North-Holland, 1978, 487-500.

[12] Kritzinger, P., Krzesinski, A., Teunissen, P.,"Design of a control system for a timesharing computer system", Perf. Of Computer Installations, D. Ferrari (ed.), North-Holland, 1978, 103-114.

[13] Denning, P. and Kahn, K., "An L=S criterion for optimal multiprogramming", Proc. Int'l Symp. On Computer Perf. Modelling, Meas. and Eval., March 1976, 219-229.

[14] Hine, J., Mitrani, I., Tsur, S., "The control of response times in multi-class systems by memory allocation", Comm. ACM, Vol.22, No.7, July 1979, 415-423.

[15] Bertrekas, D.,Dynamic Programming and Stochastic Control, Academic Press, 1976.

[16] Courtois, P., Decomposabality -Queueing and Computer System Applications, Academic Press, 1977.

[17] Chamberlin, D., Fuller, S., Liu, L., "An analysis of page allocation strategies for virtual memory systems", IBM

Journal of Research and Development, Vol.17, 1973, 404-412.

[18] Baskett, F., Chandy, K., Muntz, R., Polaius-Gomez, J., "Open, closed, and mixed networks of queues with different classes of customers", J.ACM, Vol.22, No.2, April 1975, 248-260.

[19] Buzen, J.,"Computational algorithms for closed queueing networks with exponential servers", Comm. Of ACM, Vol.16, No.9, Sept. 1973, 527-531.

[20] Lo, R.,"The application of optimal stochastic control theory to adaptive performance control of computer systems", M.S. Thesis, Dept. Of Computer Science, University of British Columbis, June 1980.

[21] Kleinrock, L., "Certain analytic results for timeshared processors", Proc. IFIPS Congress 68, 1968, 838-845.

[22] Chanson, S. and Sinha, P.,"Adaptive load control in batch-interactive computer systems", Proc. Of 16th Computer Performance Evaluation Users Group, Oct. 1980, 207-213.

FIGURES 5(a) to 8(a):

_N_ : Mean interactive response time (No control)

_*_ : " " " " (with control)

_B_ : Mean batch response time (No control)

-+- : " " " " (with control)

___ : Lower bound of mean interative response time

Figures 5(b) to 8(b):

___ :Total system throught rates (jobs/s)

-*- :Interactive throughput rate (with control)

_I_ : " " " (No control)

-+- : Batch throuhgput rate (with control)

_B_ : " " " (NO control)

Fig. 4. Legend for Figures 5(a) to 8(a)
and Figures 5(b) to 8(b)

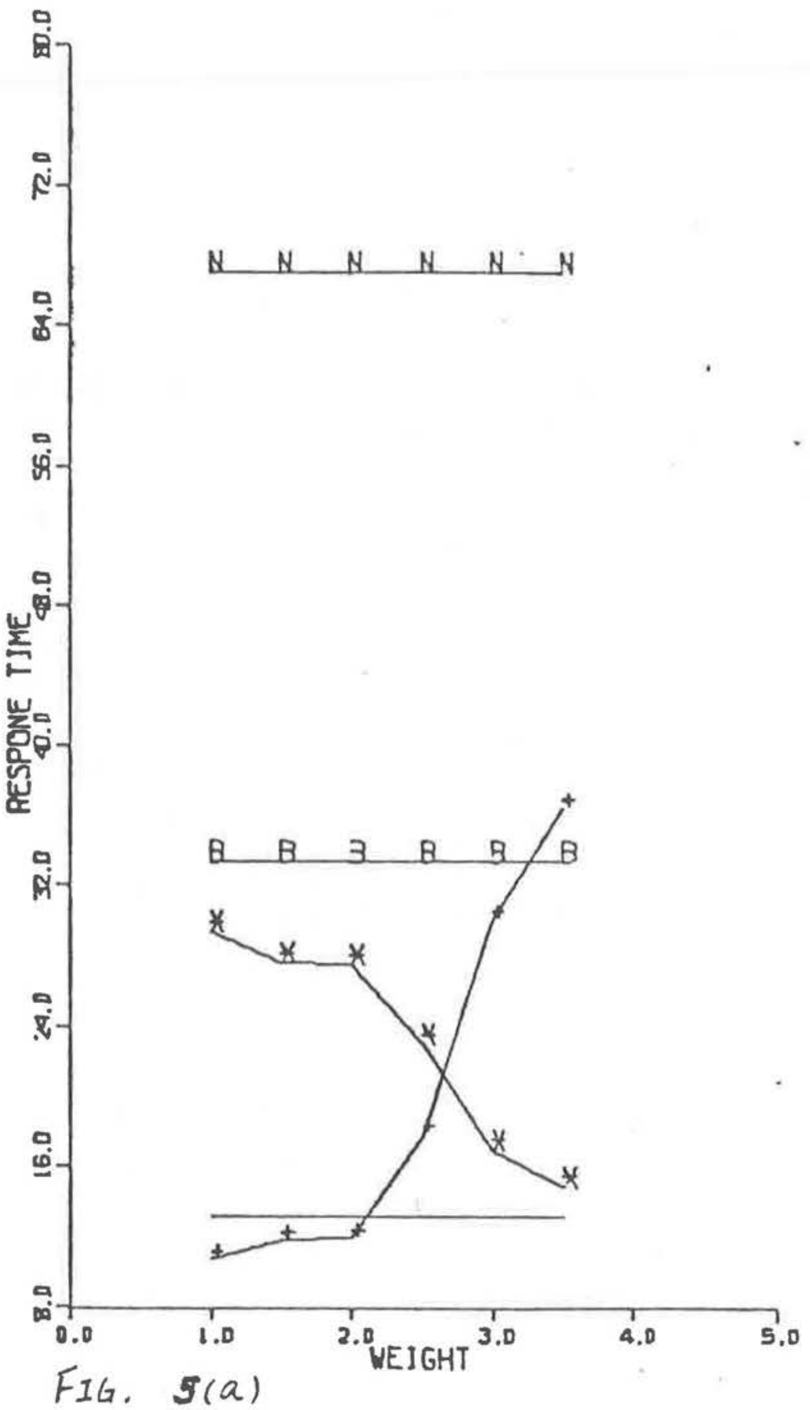

FIG. 5(a)

cpuT=0.3, cpuB=0.3

$\phi_T=0.05$, $\phi_B=0.1$, M=60, N=6, m2=5, $b_T=0.018$

$b_B=0.012$, $c_B=10$, $c_T=20$, $T_{IO}=20$, $B_{IO}=10$, $F_s=30$, $P_s=80$

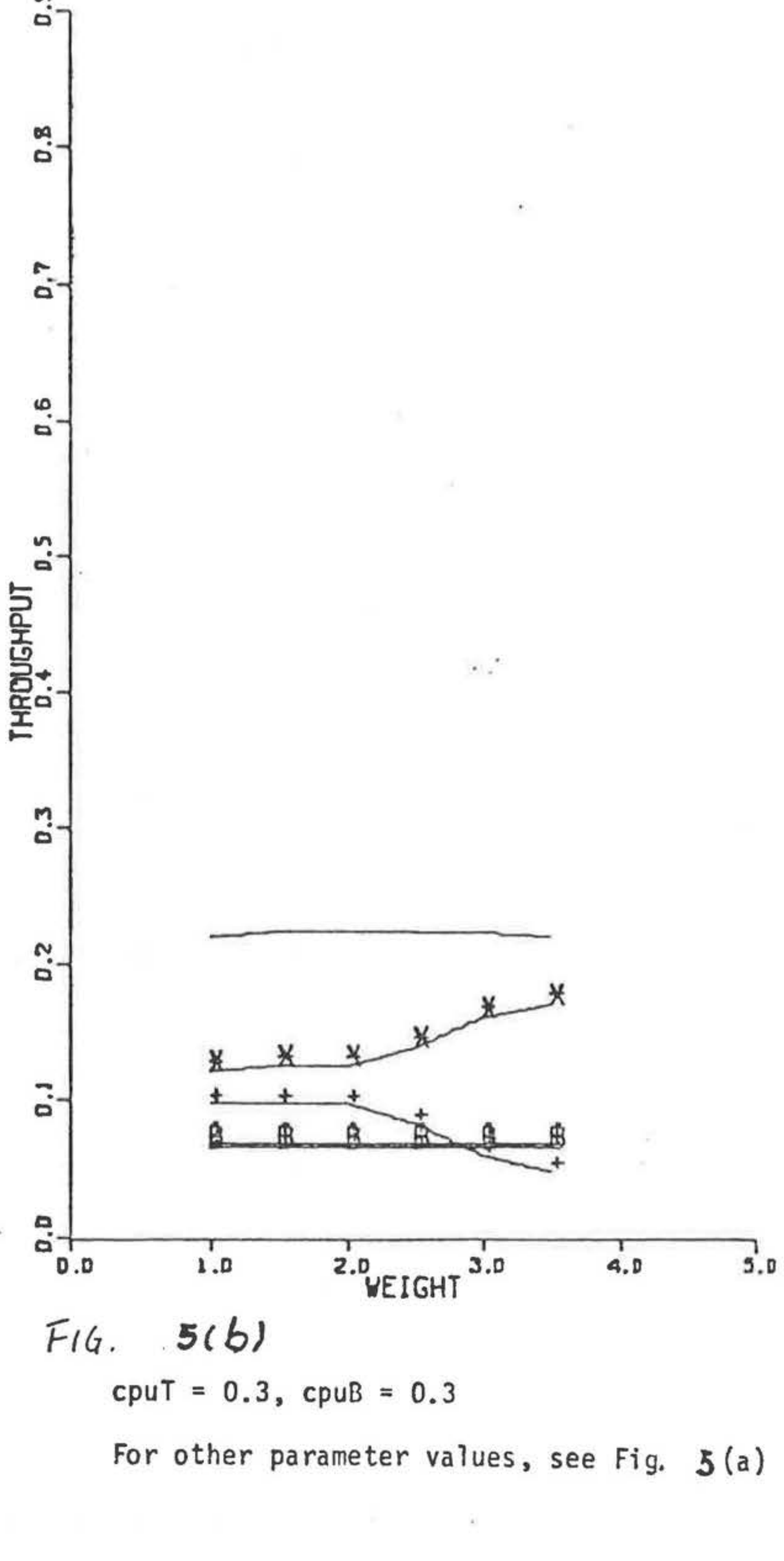

FIG. 5(b)

cpuT = 0.3, cpuB = 0.3

For other parameter values, see Fig. 5(a)

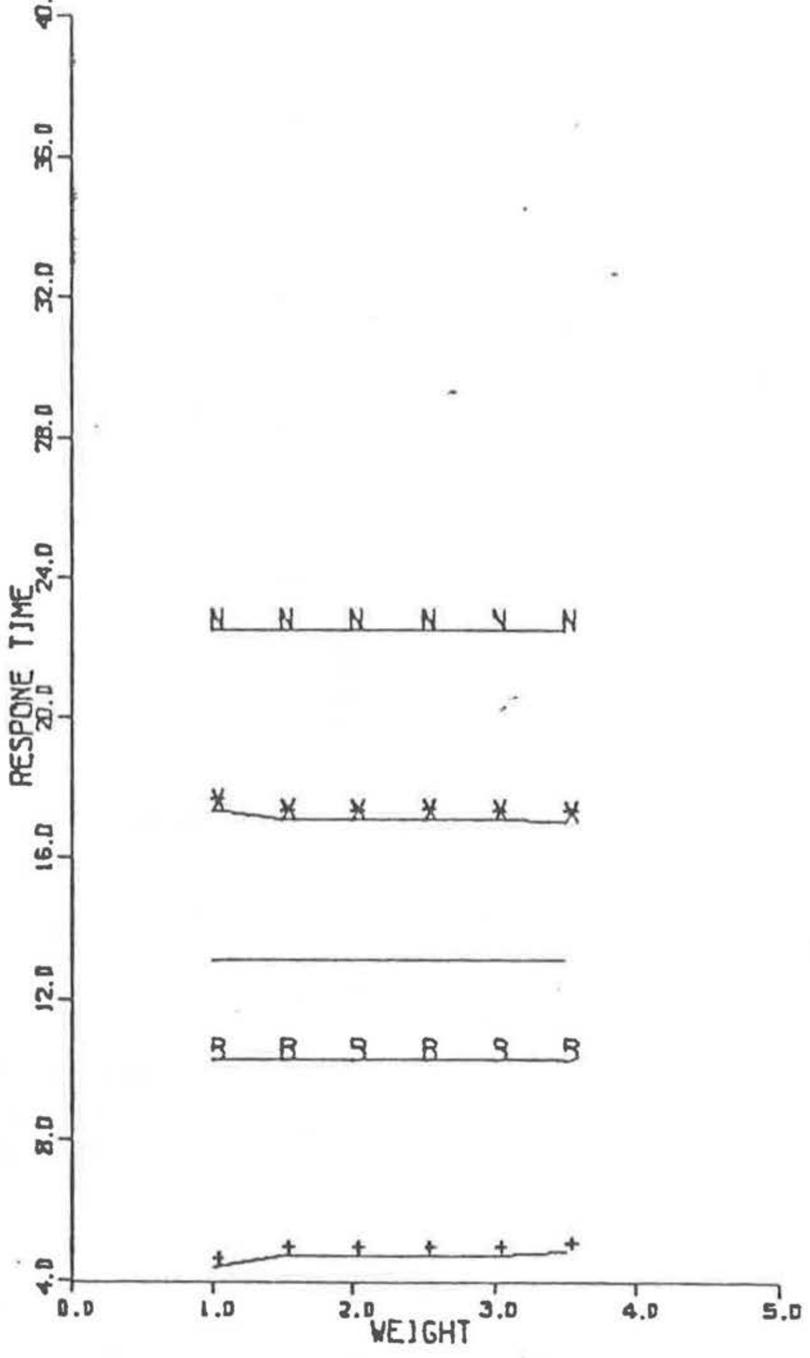

FIG. 6(a)

cpuT = 0.3, cpuB = 0.7

For other parameter values, see Fig. 5(a)

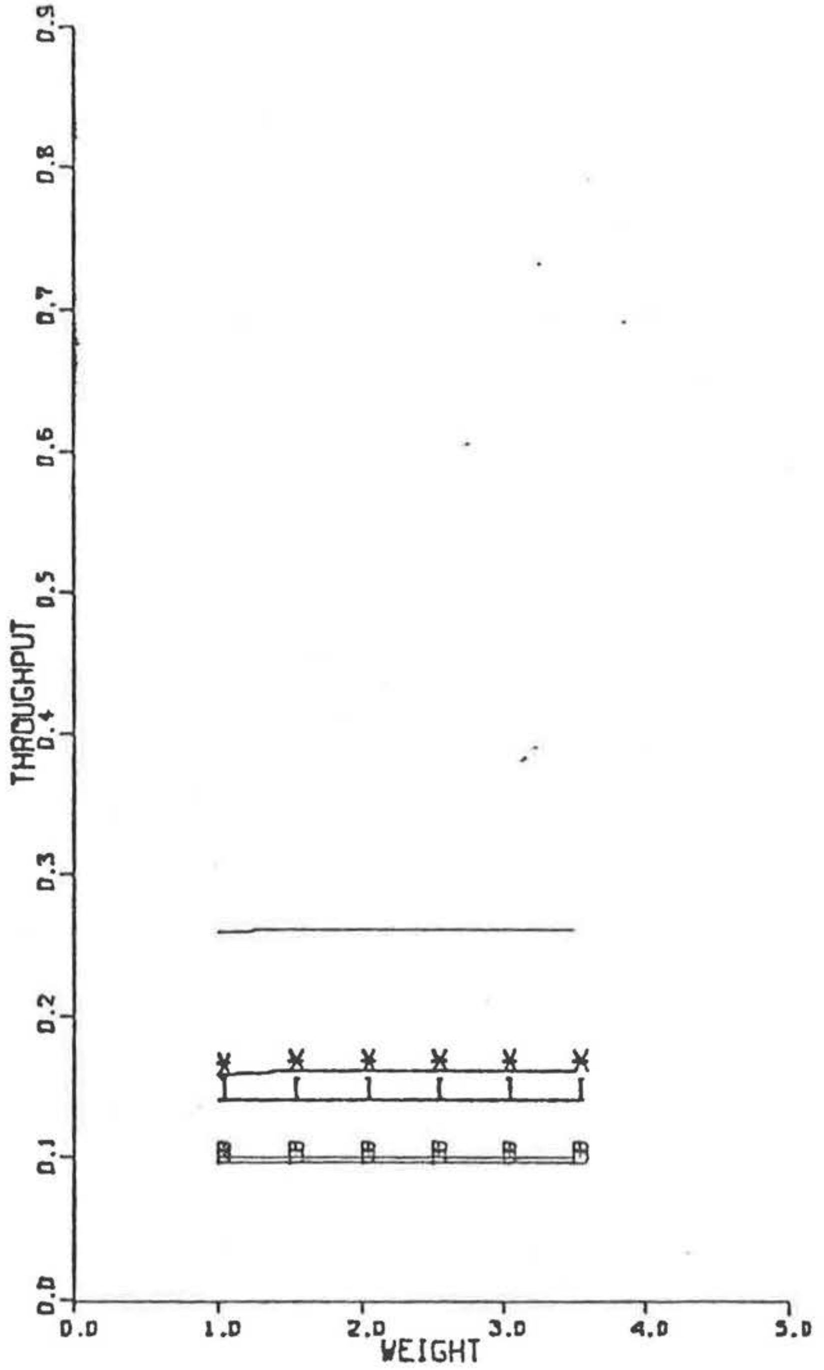

FIG. 6(b)

cpuT = 0.3, cpuB = 0.7

For other parameter values, see Fig. 5(a)

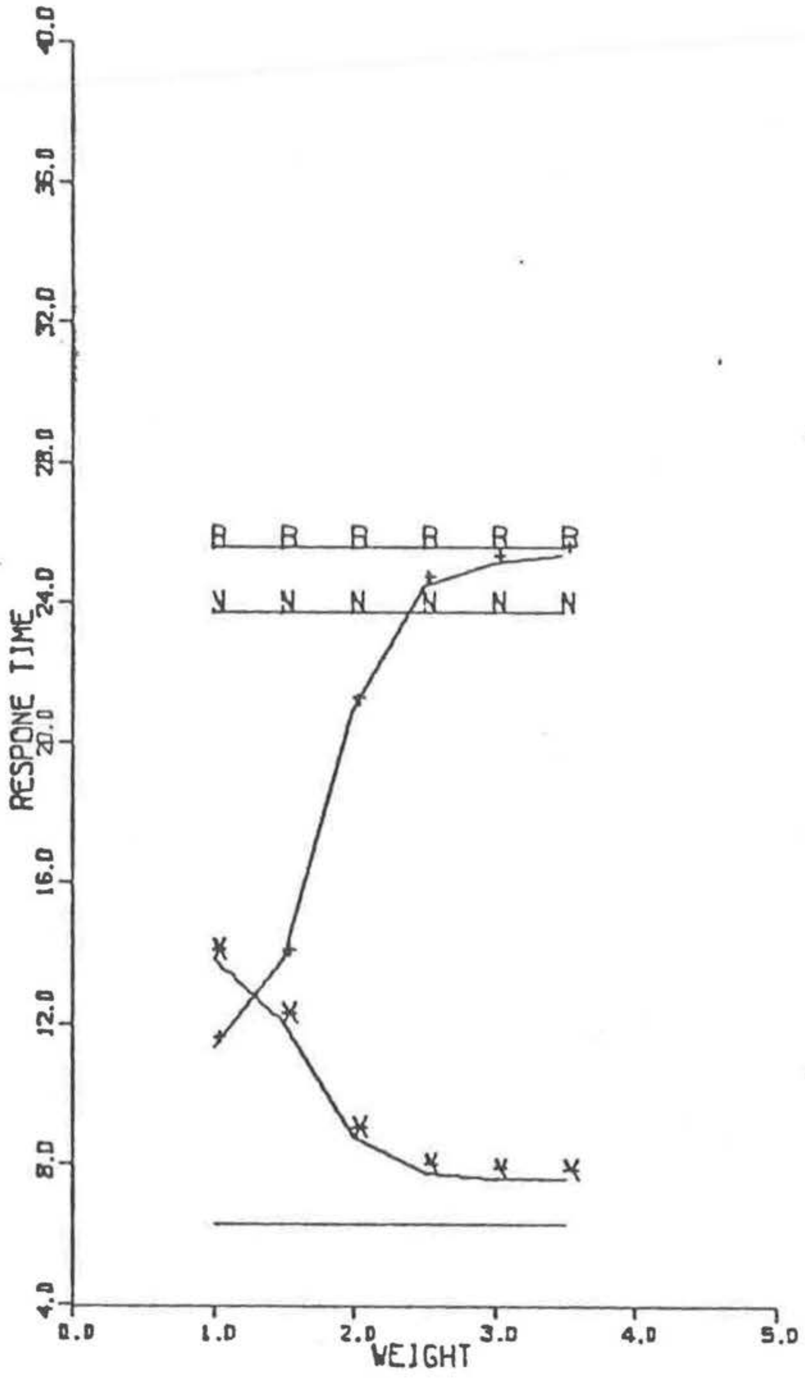

FIG 7(a)

cpuT = 0.5, cpuB = 0.3

For other parameter values, see Fig. $(a)

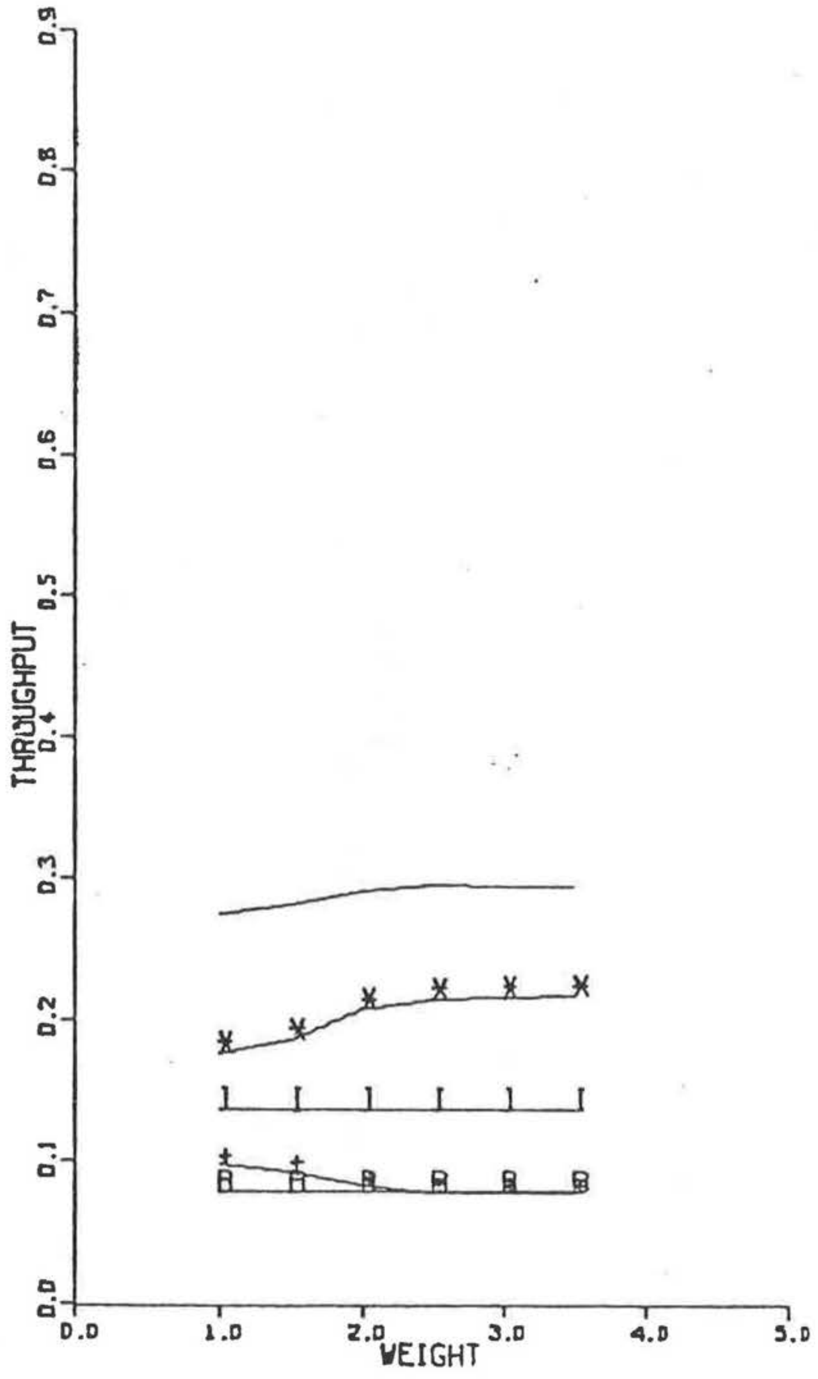

FIG. 7(b)

cpuT = 0.5, cpuB = 0.3

For other parameter values, see Fig. 5(a)

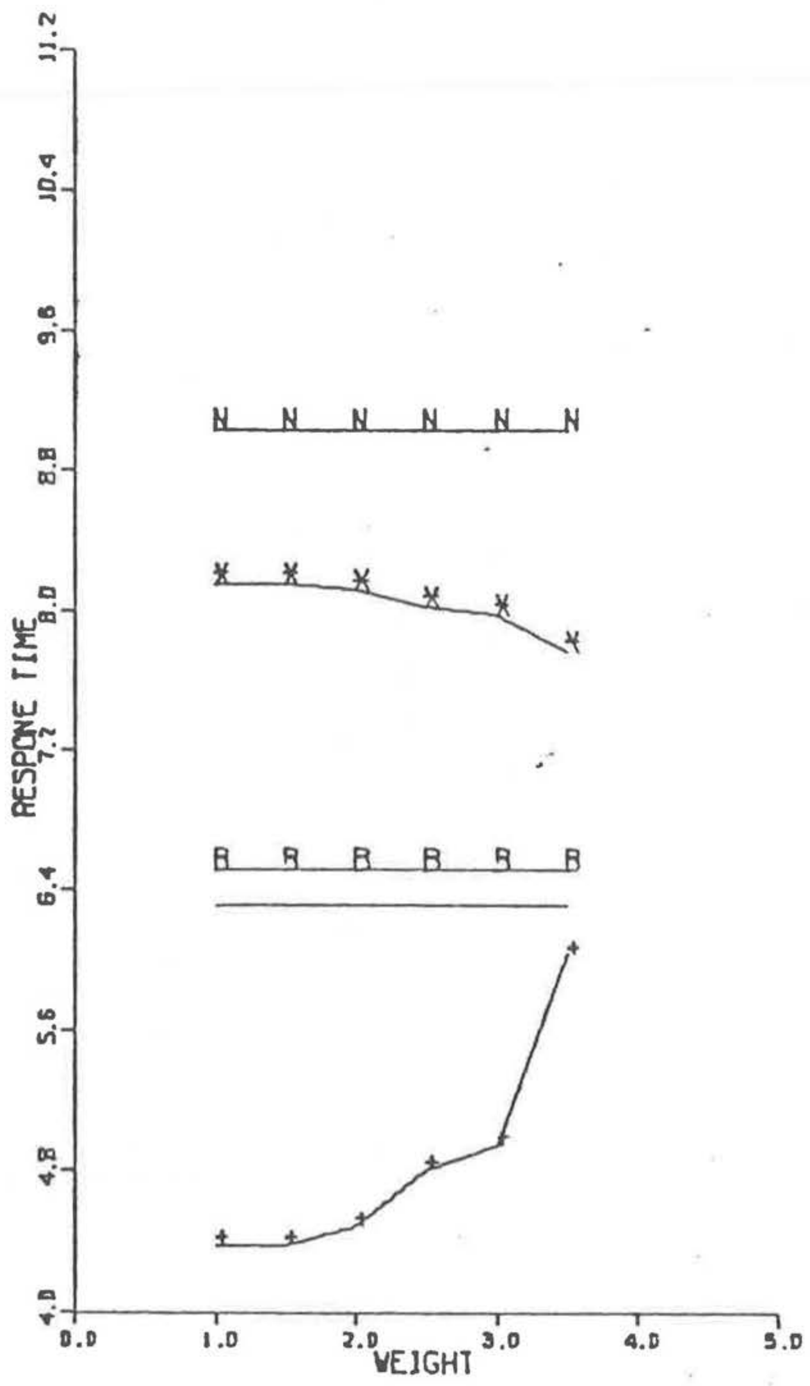

FIG. 8(a)

cpuT = 0.5, cpuB = 0.7

For other parameter values, see Fig. 5(a)

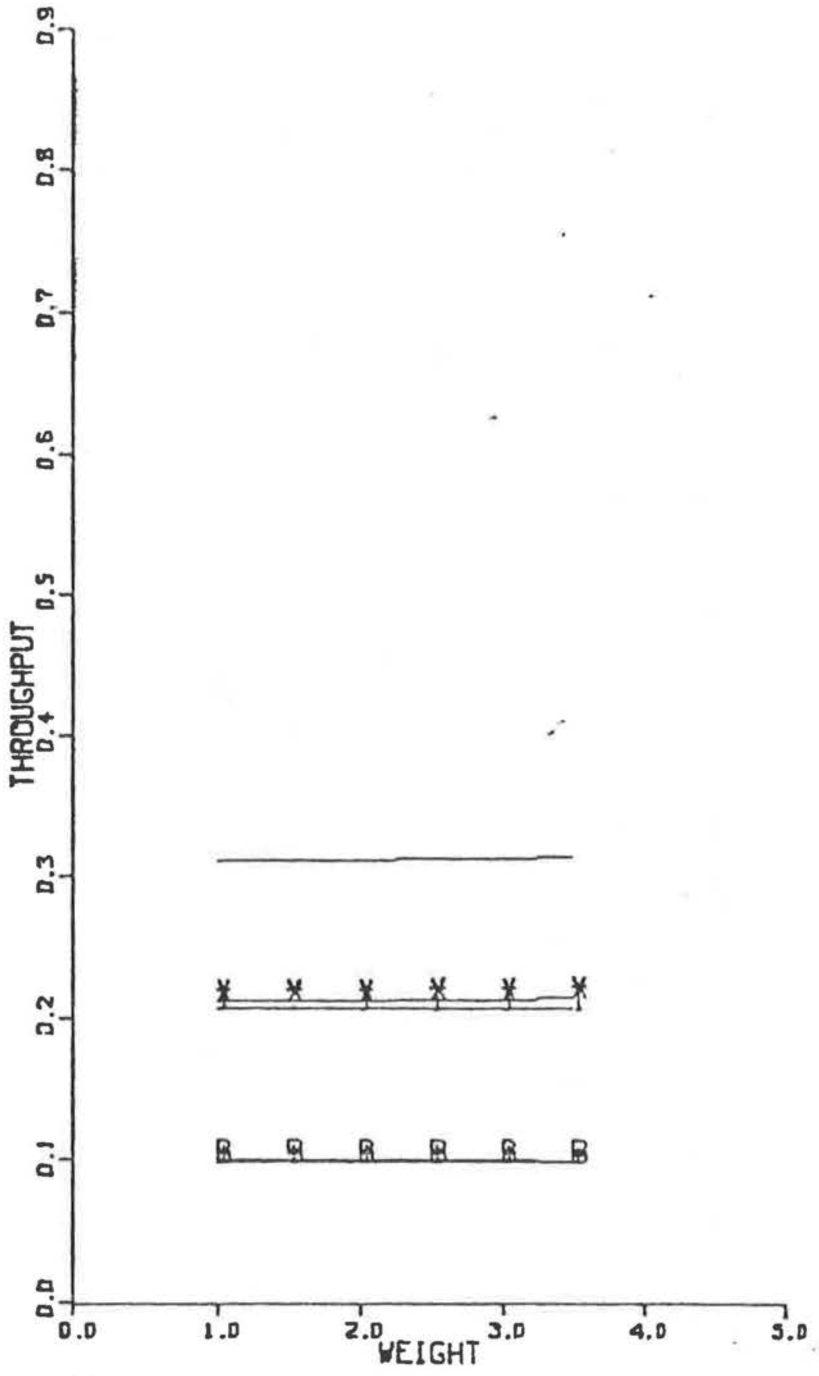

FIG. 8(b)

cpuT = 0.5, cpuB = 0.7

For other parameter values, see Fig 5(a)

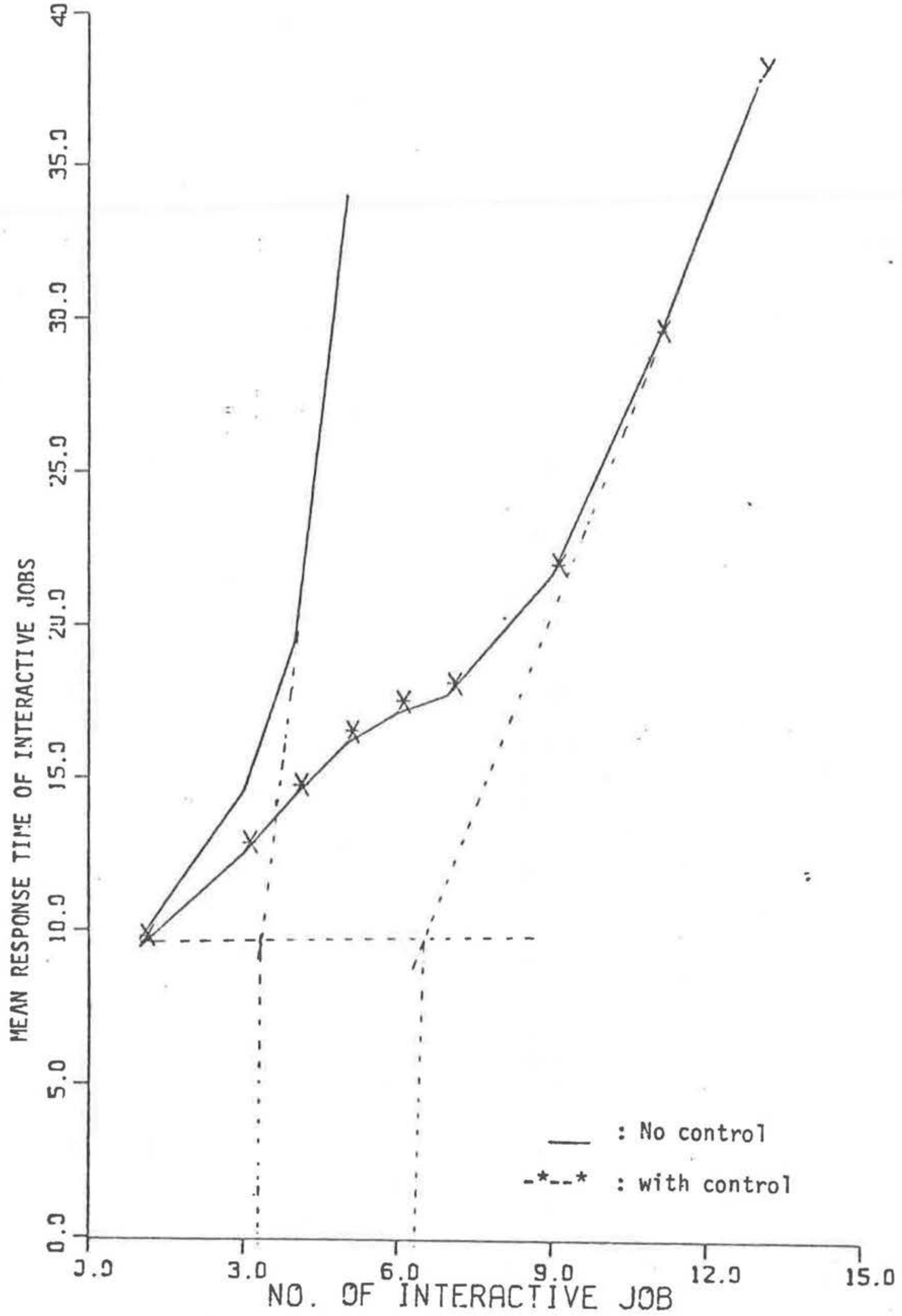

FIG. 9

cpuT = 0.3, cpuB = 0.3

For other parameter values, see Fig. 5(a)

IN TABLES (1) TO (10), ENTRY (I,J) CORRESPONDS TO THE SYSTEM STATE WITH I INTERACTIVE JOBS AND J BATCH JOBS. THE TUPLE (X,Y) REPRESENTS THE CONTROL DECISION (I.E., ADMIT X INTERACTIVE AND Y BATCH JOBS INTO THE MEMORY LOOP.).

All the tables have the following parameter values:

ØT = 0.05; ØB = 0.1, Y = 60, N = 6, m2 = 5, bT=0.018, bB = 0.012, cB = 10, cT = 20, TIO = 20, BIO = 10, Fs = 30, Ps = 80.

```
Mean interactive response time without control        = 66.97
Mean batch turn around  time without control          = 33.32
Throughput of interactive job without control         =  0.0690
Throughput of batch job without control               =  0.0669
```

WEIGHT = 1.0

Optimal policy is

				J			
		0	1	2	3	4	5
	0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,4)
	1	(1,0)	(1,1)	(0,2)	(0,3)	(0,4)	(0,4)
	2	(2,0)	(2,1)	(0,2)	(0,3)	(0,4)	(0,4)
I	3	(3,0)	(2,1)	(0,2)	(0,3)	(0,4)	(0,4)
	4	(3,0)	(2,1)	(0,2)	(0,3)	(0,4)	(0,4)
	5	(3,0)	(2,1)	(0,2)	(0,3)	(0,4)	(0,4)
	6	(3,0)	(2,1)	(0,2)	(0,3)	(0,4)	(0,4)

```
Mean interactive response time                        = 29.35
Mean batch turn around  time                          = 10.72
Throughput rate of interactive job                    =  0.1216
Throughput rate of batch job                          =  0.0990
```

TABLE 1. OPTIMAL ADMISSION POLICY WITH CPUT=0.3, CPUB=0.3

```
WEIGHT = 2.0

    Optimal policy is
    ******************** J ****************
      0      1      2      3      4      5
  0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
  1 (1,0)  (1,1)  (1,2)  (0,3)  (0,4)  (1,3)
  2 (2,0)  (2,1)  (1,2)  (0,3)  (0,4)  (1,3)
I 3 (3,0)  (2,1)  (1,2)  (0,3)  (0,4)  (1,3)
  4 (4,0)  (2,1)  (1,2)  (0,3)  (0,4)  (1,3)
  5 (3,0)  (2,1)  (1,2)  (0,3)  (0,4)  (0,4)
  6 (3,0)  (2,1)  (1,2)  (0,3)  (0,4)  (1,3)
    ****************************************

Mean interactive response time                        = 27.50
Mean batch turn around  time                          = 11.91
Throughput rate of interactive job                    =  0.1263
Throughput rate of batch job                          =  0.0986
```

TABLE 2. OPTIMAL ADMISSION POLICY WITH CPUT=0.3, CPUB=0.3

```
WEIGHT = 3.0

    Optimal policy is
    ******************** J ****************
      0      1      2      3      4      5
  0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
  1 (1,0)  (1,1)  (1,2)  (1,3)  (1,3)  (1,2)
  2 (2,0)  (2,1)  (2,2)  (1,3)  (2,1)  (2,0)
I 3 (3,0)  (2,1)  (2,2)  (2,2)  (3,0)  (3,0)
  4 (3,0)  (2,1)  (2,2)  (3,0)  (3,0)  (3,0)
  5 (3,0)  (2,1)  (2,2)  (3,0)  (3,0)  (3,0)
  6 (3,0)  (2,1)  (2,2)  (3,0)  (3,0)  (3,0)
    ****************************************

Mean interactive response time                        = 16.96
Mean batch turn around  time                          = 30.07
Throughput rate of interactive job                    =  0.1623
Throughput rate of batch job                          =  0.0609
```

TABLE 3. OPTIMAL ADMISSION POLICY WITH CPUT=0.3, CPUB=0.3

```
WEIGHT = 2.0

     Optimal policy is
     ********************* J ****************
       0      1      2      3      4      5
   0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
   1 (1,0)  (1,1)  (1,2)  (1,2)  (1,2)  (1,1)
   2 (2,0)  (2,1)  (2,1)  (2,1)  (2,0)  (2,0)
 I 3 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
   4 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
   5 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
   6 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
     *****************************************

Mean interactive response time                =  7.55
Mean batch turn around  time                  = 34.92
Throughput rate of interactive job            =  0.2178
Throughput rate of batch job                  =  0.0598
```

TABLE 4. OPTIMAL ADMISSION POLICY WITH CPUT=0.5, CPUB=0.2

```
WEIGHT = 2.0

     Optimal policy is
     ********************* J ****************
       0      1      2      3      4      5
   0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
   1 (1,0)  (1,1)  (1,2)  (1,3)  (0,4)  (1,2)
   2 (2,0)  (2,1)  (2,2)  (1,3)  (1,3)  (2,0)
 I 3 (3,0)  (3,1)  (2,2)  (1,3)  (1,3)  (3,0)
   4 (3,0)  (3,1)  (2,2)  (1,3)  (2,2)  (3,0)
   5 (3,0)  (2,1)  (2,2)  (1,3)  (2,2)  (3,0)
   6 (3,0)  (2,1)  (2,2)  (1,3)  (3,0)  (3,0)
     *****************************************

Mean interactive response time                =  9.65
Mean batch turn around  time                  = 10.66
Throughput rate of interactive job            =  0.2023
Throughput rate of batch job                  =  0.0965
```

TABLE 5. OPTIMAL ADMISSION POLICY WITH CPUT=0.5, CPUB=0.4

```
WEIGHT = 2.0

    Optimal policy is
    ******************** J ****************
        0      1      2      3      4      5
   0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
   1 (1,0)  (1,1)  (1,2)  (1,3)  (0,4)  (1,3)
   2 (2,0)  (2,1)  (2,2)  (1,3)  (0,4)  (0,4)
 I 3 (3,0)  (2,1)  (1,2)  (0,3)  (0,4)  (0,4)
   4 (3,0)  (2,1)  (1,2)  (1,3)  (0,4)  (0,4)
   5 (3,0)  (2,1)  (1,2)  (1,3)  (0,4)  (1,3)
   6 (3,0)  (2,1)  (1,2)  (0,3)  (0,4)  (0,4)
    *****************************************

Mean interactive response time                      =  8.55
Mean batch turn around  time                        =  5.36
Throughput rate of interactive job                  =  0.2101
Throughput rate of batch job                        =  0.0999
```

TABLE 6. OPTIMAL ADMISSION POLICY WITH CPUT=0.5, CPUB=0.6

```
WEIGHT = 3.5

    Optimal policy is
    ******************** J ****************
        0      1      2      3      4      5
   0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
   1 (1,0)  (1,1)  (1,2)  (1,2)  (1,2)  (1,0)
   2 (2,0)  (2,1)  (2,1)  (2,0)  (2,0)  (2,0)
 I 3 (3,0)  (3,1)  (3,0)  (3,0)  (3,0)  (3,0)
   4 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
   5 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
   6 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
    *****************************************

Mean interactive response time                      = 23.84
Mean batch turn around  time                        = 48.66
Throughput rate of interactive job                  =  0.1369
Throughput rate of batch job                        =  0.0068
Lower bound of interactive response time            = 23.68
```

TABLE 7. OPTIMAL ADMISSION POLICY WITH CPUT=0.2, CPUB=0.2

```
WEIGHT = 9.0

     Optimal policy is
     ******************** J ****************
       0       1       2       3       4       5
   0 (0,0)   (0,1)   (0,2)   (0,3)   (0,4)   (0,4)
   1 (1,0)   (1,1)   (1,2)   (1,2)   (1,2)   (1,1)
   2 (2,0)   (2,1)   (2,1)   (2,1)   (2,1)   (2,0)
 I 3 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   4 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   5 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   6 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
     ****************************************

Mean interactive response time                 = 24.53
Mean batch turn around  time                   = 41.58
Throughput rate of interactive job             =  0.1347
Throughput rate of batch job                   =  0.0303
Lower bound of interactive response time       = 23.68
```

TABLE 8. OPTIMAL ADMISSION POLICY WITH CPUT=0.2, CPUB=0.5

```
WEIGHT = 3.5

     Optimal policy is
     ******************** J ****************
       0       1       2       3       4       5
   0 (0,0)   (0,1)   (0,2)   (0,3)   (0,4)   (0,4)
   1 (1,0)   (1,1)   (1,2)   (1,3)   (1,2)   (1,1)
   2 (2,0)   (2,1)   (2,1)   (2,1)   (2,0)   (2,0)
 I 3 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   4 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   5 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
   6 (3,0)   (3,0)   (3,0)   (3,0)   (3,0)   (3,0)
     ****************************************

Mean interactive response time                 =  9.96
Mean batch turn around  time                   = 30.91
Throughput rate of interactive job             =  0.2002
Throughput rate of batch job                   =  0.0653
Lower bound of interactive response time       =  8.64
```

TABLE 9. OPTIMAL ADMISSION POLICY WITH CPUT=0.4, CPUB=0.3

```
WEIGHT = 3.5

    Optimal policy is
    ********************* J *****************
       0      1      2      3      4      5
  0 (0,0)  (0,1)  (0,2)  (0,3)  (0,4)  (0,4)
  1 (1,0)  (1,1)  (1,2)  (1,2)  (1,1)  (1,0)
  2 (2,0)  (2,0)  (2,0)  (2,0)  (2,0)  (2,0)
I 3 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
  4 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
  5 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
  6 (3,0)  (3,0)  (3,0)  (3,0)  (3,0)  (3,0)
    *****************************************

Mean interactive response time                      =  5.42
Mean batch turn around  time                        = 34.61
Throughput rate of interactive job                  =  0.2360
Throughput rate of batch job                        =  0.0596
Lower bound of interactive response time            =  4.93
```

TABLE 10. OPTIMAL ADMISSION POLICY WITH CPUT=0.6, CPUB=0.

N1 \ N2	0	1	2	3	4	5
0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.5352	0.7495	0.8386	0.8689	0.8640
1	0.4872	0.3379	0.2454	0.1824	0.1359	0.0993
	0.0	0.3832	0.5703	0.6624	0.6986	0.6924
2	0.6779	0.4946	0.3671	0.2711	0.1956	0.1225
	0.0	0.2917	0.4496	0.5298	0.5553	0.5048
3	0.7483	0.5544	0.4059	0.2894	0.1799	0.1243
	0.0	0.2292	0.3573	0.4175	0.4013	0.3803
4	0.7449	0.5404	0.3810	0.2356	0.1633	0.1090
	0.0	0.1008	0.2790	0.2994	0.3024	0.2752
5	0.6750	0.4711	0.2900	0.2017	0.1352	0.1218
	0.0	0.1399	0.1987	0.2257	0.2193	0.2479
6	0.5599	0.3436	0.2396	0.1611	0.1453	0.0937
	0.0	0.0990	0.1498	0.1638	0.1977	0.1716

TABLE 11. CPU UTILIZATION UT(N1,N2)*/UB(N1,N2)
N1 AND N2 ARE THE NUMBER OF INTERACTIVE AND BATCH JOBS IN THE MEMORY LOOP RESPECTIVELY.
Y=60 BT=0.018 BB=0.012 AT=20 AB=10 FS=30 PS=80

```
TW = interactive response time without control
TO =      "           "        "  with optimal control
TS =      "           "        "  with suboptimal control
PW = total throughput without control
PO =   "       "      with optimal control
PS =   "       "      with suboptimal control
BO = batch throughput with optimal control
BS =   "       "      with suboptimal control
```

cpuT=0.2

cpuB	0.1	0.2	0.3	0.4	0.5	0.6
TW	243	192	135	93	68.9	55.4
TO	23.6	23.6	23.7	23.8	24.5	26.6
TS	24.3	24.4	24.5	24.5	24.6	24.8
PW	0.043	0.0695	0.099	0.129	0.153	0.171
PO	0.141	0.143	0.146	0.150	0.165	0.185
PS	0.141	o.147	0.154	0.160	0.165	0.180
BO	0.003	0.0054	0.0087	0.0134	0.0303	0.057
BS	0.006	0.0127	0.0193	0.0254	0.0306	0.051

TABLE 12
Comparision of the performance of the standard admission policy to optimal policies under different workloads