Belgian Journal of Operations Research, Statistics and Computer Science Vol. 32 (3,4)

Optimal control of the degree of multiprogramming in a computer system

K. Ohno[†] S. Hirao[‡] K. Nakade[†]

Department of Systems Engineering[†]
Nagoya Institute of Technology
Showa-ku
Nagoya, 466, Japan

Faculty of Science[‡]
Konan University
Higashinada-ku
Kobe, 658, Japan

Abstract

This paper deals with an optimal control problem in a multiprogrammed computer system. First, the quasi-birth-and-death process which models the system is investigated and an optimal static admission policy is derived that maximizes the throughput among policies with fixed maximum degree of multiprogramming. Numerical results show that this optimal static admission policy gives a ten percent higher throughput than the $L = S$ and knee criteria. Next, the problem is formulated into a semi-Markov decision process and an optimal dynamic admission policy is obtained which controls the admission of jobs depending on the state of the system. It is shown numerically that this dynamic admission policy attains a five percent higher throughput than the optimal static admission policy in the system with a low-speed paging disk.

Keywords : multiprogrammed computer system, throughput, $L = S$ and knee criteria, dynamic job admission control, semi-Markov decision process

1. Introduction

Large and medium scale computer systems with virtual memory maintain multiple jobs in a main memory and share system resources. Many theoretical and empirical studies on these multiprogrammed computer systems have been carried on in order to improve their performance measures. In particular, controls in operating systems have been studied in two aspects: one is related to resource allocation and the other is an admission control [9].

The former concerns scheduling strategies to decide when and how system resources should be allocated among active multiprogrammed jobs. These resources include a CPU, a main memory, secondary storage devices and so on. Kameda [13] has dealt with CPU scheduling strategies. Memory allocation strategies include page replacement algorithms and software techniques for reducing page fault occurrences [1,7,10]. Scheduling strategies in secondary storage devices, especially, disk storage devices, have also been investigated [5].

An admission control decides how many and which types of jobs should be admitted into the active set. Admission scheduling strategies for an arrival job stream have been discussed in a multiprogrammed computer system with a predetermined maximum degree of multiprogramming [14]. The most fundamental and important scheduling problem in a multiprogrammed computer system is to control the degree of multiprogramming. This is because when the degree is too high, performance of the system is heavily declining by thrashing [6].

To avoid thrashing, several strategies have been proposed

and empirically analyzed. Well-known strategies are 50 %, L=S and knee criteria. Denning et al. [8] have discussed the characteristics of these criteria and compared them empirically under several kinds of job mix environment. These policies, however, have little theoretical background.

In recent years, several theoretical approximations are suggested. Blake [2] has applied dynamic programming to the roughly approximate model of database systems. Chanson et al. [4] have formulated a multiprogrammed computer system with two types of jobs into an undiscounted semi-Markov decision process. They have utilized a decomposition method for reducing the number of states and derived a good admission policy in this approximate process.

In this paper, we deal with an optimal admission control problem in a simple multiprogrammed computer system. We are concerned with two admission policies: static and dynamic admission policies. Under the dynamic admission policy, a job is admitted into the inner system utilizing available information about the system. That is, the degree of multiprogramming is dynamically controlled. The static admission policy is specified by the maximum degree of multiprogramming. Under this policy, a job is admitted unless the degree of multiprogramming exceeds the maximum degree. The conventional L=S and knee criteria are considered as static admission policies. We derive optimal static and dynamic admission policies that maximize the throughput and compare them numerically.

In section 2, we explain the multiprogrammed computer system to be considered. In section 3, we analyze the computer system with a fixed maximum degree of multiprogramming,

which is described by a quasi-birth-and-death process. We use the stability condition in [16] and derive a simple expression of the throughput. An optimal static admission policy is determined by the maximum degree of the multiprogramming that maximizes the throughput. In section 4, we deal with an optimal dynamic admission policy for the multiprogrammed computer system. The problem is formulated into an undiscounted semi-Markov decision process. In section 5, we compute throughputs and compare them in two steps. At first, the optimal admission policy is compared with the knee and L=S criteria. Secondly, an optimal dynamic admission policy computed by a policy iteration method is compared with the optimal static admission policy. Numerical examples show that the optimal dynamic policy attains a five percent higher throughput than the optimal static policy in the system with a low-speed paging disk.

2. A Multiprogrammed Computer System

We consider a multiprogrammed computer system as shown in Fig. 1. The system consists of two service stations, which we call "inner system," and an input queue called queue 0. Station 1 is a CPU and has two scheduling queues called queues 1 and 2. We assume that the CPU processes jobs under the service discipline where jobs in queue 2 have higher priority than in queue 1. Station 2 is a paging disk and has a queue called queue 3. Let T be the total number of pages in the main memory, which equals the maximum possible degree. We assume first in and first out discipline in queue 0 and 3. Jobs arrive at the input queue according to a Poisson process with rate λ . The

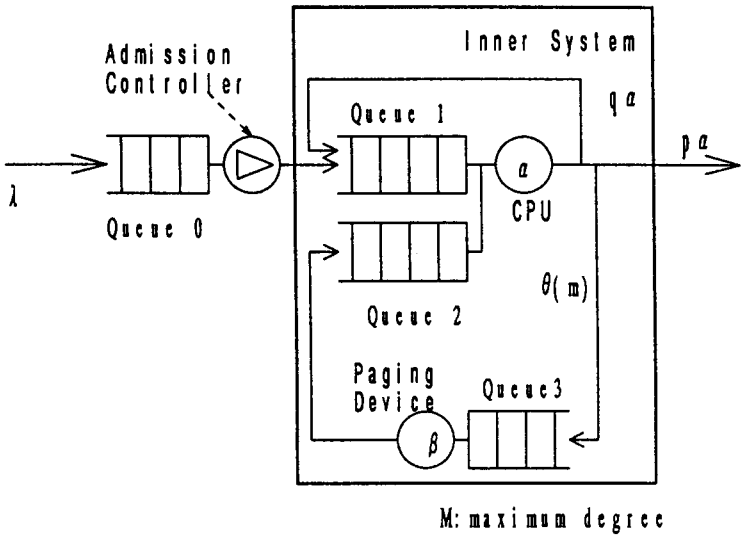

Fig.1 Multiprogrammed computer system

arrival jobs in the input queue are admitted into the inner system under the static admission policy, whenever the number of jobs in the inner system is less than the maximum degree of multiprogramming, M ($\leq$ T). On the other hand, under the dynamic admission policy, whenever the state of the system changes, the admission controller shown in Fig. 1 decides whether a job waiting in queue 0 is admitted into the inner system or not, utilizing available information about the system. Stations 1 and 2 have exponential service times with mean α^{-1} and β^{-1}, respectively. Each job in service at the CPU requests again the service of the CPU with rate $q\alpha$, departs from the inner system with rate $p\alpha$ or proceeds to the disk with rate θ (m), where p+q=1, p,q>0, and m is the number of jobs

in the inner system, i.e., the degree of multiprogramming. The rate θ (m) is called the page fault rate and is given shortly.

System lifetime function L(m) is defined as the average amount of CPU service between two consecutive page faults when the degree of multiprogramming is m, and given by

$$L(m)=e(T/m), \tag{1}$$

where e(n) denotes a program lifetime function defined as the average amount of CPU service received by a job between two consecutive page faults when n pages are allocated to the job [8]. The following program lifetime function is used frequently [3]:

$$e(n)=2\cdot b/(1+(c/n)^2), \tag{2}$$

where b and c are parameters. As L(m) is the system lifetime function, the page fault rate θ (m) is related to it by

$$\theta(m)=1/L(m).$$

The system is described by a quasi-birth-and-death process and an expression of its throughput is derived in the next section.

3. Optimal Static Admission Policy

Several queueing network models for evaluating the performance of multiprogrammed computer systems have been proposed and analyzed. For example, Kobayashi [15] has calculated various characteristics of a multiprogrammed computer system with M=3. In this section, we derive an explicit expression for the throughput of the system with a fixed maximum degree M.

The jobs in the system are classified into the following

three classes:

class 0; jobs in queue 0 (input queue),

class 1; jobs in station 1 (CPU),

class 2; jobs in station 2 (disk).

Let S be the state space of the system. Then S is as follows:

$$S=\{(i,j,k);0\le j\le M,0\le k\le M,j+k\le M \quad \text{for } i=0,$$
$$0\le j\le M,0\le k\le M,j+k=M \quad \text{for } i>0\}$$

where i, j and k denote the number of jobs of classes 0, 1 and 2, respectively. States in the subsets $\underline{i}=\{(i,j,k)\}$ of S are arranged as follows:

$$\underline{0}=\{(0,0,m),(0,1,m-1),\ldots,(0,m,0),\ m=0,1,\ldots,M\},$$
$$\underline{i}=\{(i,0,M),(i,1,M-1),\ldots,(i,M,0)\},\quad \text{for } i\ge 1.$$

In addition, the subsets of S are arranged as $S=\{\underline{0},\underline{1},\underline{2},\ldots\}$. Then the behavior of the system can be described by a quasi-birth-and-death process with the infinitesimal generator Q given by

$$Q=\begin{vmatrix} B_{00} & B_{01} & & & \\ B_{10} & A_1 & A_0 & & \\ & A_2 & A_1 & A_0 & \\ & & \ddots & \ddots & \ddots \end{vmatrix},$$

where A_0 and A_1 are the (M+1)x(M+1) transition matrices from $\underline{i}$ to $\underline{i+1}$ and $\underline{i}$, respectively, for $i\ge 1$. A_2 is the (M+1)x(M+1) transition matrix from $\underline{i}$ to $\underline{i-1}$ for $i\ge 2$. B_{00}, B_{01} and B_{10} are the transition matrices from $\underline{0}$ to $\underline{0}$, from $\underline{0}$ to $\underline{1}$ and from $\underline{1}$ to $\underline{0}$, respectively, where the numbers of the states in $\underline{0}$ and $\underline{1}$ are (M+1)(M+2)/2 and (M+1), respectively. Let (x,y) element of matrix A_a be denoted by $(A_a)_{xy}$, where a=0,1,2 and $0\le x, y\le M$. Then,

A_0, A_1 and A_2 are given as follows:

$$A_0=\lambda I, \tag{3}$$

$$(A_1)_{xy}=\theta(M) \quad \text{for } 1\leq x\leq M \text{ and } y=x-1,$$

$$=\beta \quad \text{for } 0\leq x\leq M-1 \text{ and } y=x+1,$$

$$=-\lambda-\min(1,x)\,\theta(M)-\min(1,M-x)\beta-\min(1,x)p\alpha \quad \text{for } 0\leq x\leq M \text{ and } y=x,$$

$$=0, \quad \text{otherwise,} \tag{4}$$

$$(A_2)_{xy}=p\alpha \quad \text{for } 1\leq x\leq M \text{ and } y=x,$$

$$=0, \quad \text{otherwise,} \tag{5}$$

where I denotes the (M+1)x(M+1) identity matrix.

The multiprogrammed computer system with the maximum degree M is called stable if the process Q is positive recurrent. Let matrix A be given as

$$A=A_0+A_1+A_2. \tag{6}$$

Then A is the infinitesimal generator of a finite and continuous time Markov chain. Since by (4), $(A_1)_{x(x-1)}$ and $(A_1)_{x(x+1)}$ are positive, A is irreducible. We shall derive a necessary and sufficient condition for the stability. Denote by $\Pi(M)=(\pi_0,\pi_1,\ldots,\pi_M)$ the stationary probability vector of the Markov chain A. Then it holds that

$$\Pi(M)A=\mathbf{0},\ \Pi(M)\mathbf{e}=1 \text{ and } \Pi(M)\geq\mathbf{0}, \tag{7}$$

where $\mathbf{0}$ denotes the zero vector and $\mathbf{e}$ denotes the vector with all elements 1. Equations (3) through (6) lead directly to the unique solution of (7), given by

$$\pi_k(M)=\{\beta/(\theta(M))\}^k\,\pi_0(M) \quad \text{for} \quad 1\leq k\leq M,$$

where $\pi_0(M)=\{\sum_{i=0}^{M}(\beta/(\theta(M)))^i\}^{-1}$. Then a necessary and sufficient condition for the stability is given by

$\Pi(M)A_0e < \Pi(M)A_2e$ (see Theorem 1.3.2, p.19 in [16]). This reduces to

$$\lambda < p\alpha\{1-\pi_0(M)\} = p\alpha\{1-\frac{1-(\beta/(\theta(M)))}{1-(\beta/(\theta(M)))^{M+1}}\}$$

$$= p\alpha\{\frac{\beta/(\theta(M))-(\beta/(\theta(M)))^{M+1}}{1-(\beta/(\theta(M)))^{M+1}}\} = \lambda^*(M), \quad (8)$$

where $\lambda^*(M)$ gives the upper bound of the arrival rate for the system to be stable at the maximum degree M. Consequently, $\lambda^*(M)$ is identical with "throughput" at the degree. Since a static admission policy is specified with M, an optimal admission policy maximizing the throughput is given by $M^* = \underset{M=1,\ldots,T}{\arg\max}\{\lambda^*(M)\}$, where arg max denotes the value of argument M maximizing $\lambda^*(M)$.

The throughput cannot exceed the maximum throughput $\lambda^{**} = \lambda^*(M^*)$ under static admission policies. It is expected, however, that the throughput of the system can be improved if jobs waiting in queue 0 are dynamically controlled to admit into the inner system. In the next section, we deal with an optimal dynamic admission control.

4. Optimal Dynamic Admission Policy

We analyze a dynamic policy which controls the admission of jobs waiting in queue 0 into the inner system, when the state of the system changes. Now this problem is formulated into an undiscounted semi-Markov decision process in the following [11]. As a reward structure, we suppose that reward one is obtained from a completed job.

Let X_i (i=0,...,3) be as follows:

X_0: the total number of jobs in queue 0;

X_1: the total number of jobs in station 1;

X_2: the total number of jobs in queue 2;

X_3: the total number of jobs in station 2;

Then the state of the system is expressed by vector s given by

$s = (X_0, X_1, X_2, X_3)$.

Let S be the set of possible states $\{ s;\ 0 \le X_0 \le N_0,\ 0 \le X_1, X_3,\ 0 \le X_1 + X_3 \le T,\ 0 \le X_2 \le X_1 \}$, where N_0 denotes the buffer size of the input queue 0. It is shown in section 5 that the maximum throughput among dynamic policies can be attained for a large arrival rate λ ($> p\alpha$) even though N_0 is finite.

Denote by 1 an action admitting a job waiting in queue 0 into the inner system and by 0 an action not admitting the job. When the state of the system changes into state $s \in S$, an action is chosen from the set of possible actions A(s), where given by

$$A(s) = \{0\}, \quad \text{if } X_1 + X_3 = T \text{ or } X_0 = 0,$$
$$= \{0,1\}, \quad \text{otherwise}.$$

When action $a \in A(s)$ is chosen in state $s=(X_0, X_1, X_2, X_3)$, the state of the system changes immediately into the intermediate state $s'=(X_0', X_1', X_2', X_3')$, given by:

for a=0, $X_0'=X_0$, $X_1'=X_1$, $X_2'=X_2$ and $X_3'=X_3$, and

for a=1, $X_0'=X_0-1$, $X_1'=X_1+1$, $X_2'=X_2$, and $X_3'=X_3$.

Denote by $Q_{s's''}(x|a)$ the probability that the next transition occurs and the state of the system becomes $s''=(X_0'', X_1'', X_2'', X_3'')$ within period x given that action a is chosen in state $s=(X_0, X_1, X_2, X_3)$.

Let

$$d(X_0', X_1', X_3') = \delta(X_1')(\alpha + \theta(X_1' + X_3')) + \delta(X_3')\beta + \gamma(X_0')\lambda$$

and $q(x|X_0',X_1',X_3') = \frac{1-\exp(-d(X_0',X_1',X_3')x)}{d(X_0',X_1',X_3')}$,

where

$\delta(X) = \min(1,X)$

and $\gamma(X_0') = 0$, if $X_0' = N_0$,

$= 1$, if $X_0' < N_0$.

Then the transition probabilities are as follows:

1) When $d(X_0',X_1',X_3') \neq 0$,

$Q_{ss''}(x|a) = q(x|X_0',X_1',X_3')\gamma(X_0')\lambda$,

if $s'' = (X_0'+1, X_1', X_2', X_3')$,

$= q(x|X_0',X_1',X_3')\delta(X_3')\beta$,

if $s'' = (X_0', X_1'+1, X_2'+1, X_3'-1)$,

$= q(x|X_0',X_1',X_3')\delta(X_1')q\alpha$,

if {$s'' = (X_0', X_1', X_2', X_3')$ and $X_2' = 0$}

or {$s'' = (X_0', X_1', X_2'-1, X_3')$ and $X_2' > 0$},

$= q(x|X_0',X_1',X_3')\delta(X_1')\theta(X_1'+X_3')$,

if {$s'' = (X_0', X_1'-1, X_2', X_3'+1)$ and $X_2' = 0$}

or {$s'' = (X_0', X_1'-1, X_2'-1, X_3'+1)$ and $X_2' > 0$},

$= q(x|X_0',X_1',X_3')\delta(X_1')p\alpha$,

if {$s'' = (X_0', X_1'-1, X_2', X_3')$ and $X_2' = 0$}

or {$s'' = (X_0', X_1'-1, X_2'-1, X_3')$ and $X_2' > 0$},

$= 0$, otherwise.

2) When $d(X_0',X_1',X_3') = 0$,

$Q_{ss''}(x|a) = 1-\exp(-\lambda x)$, if $s'' = (X_0', X_1', X_2', X_3')$,

$= 0$, otherwise.

Define $P_{ss''}(a)$ by

$$P_{ss''}(a) = \lim_{x \to \infty} Q_{ss''}(x|a).$$

The mean time $\nu(s,a)$ until the next transition given

that action a is chosen in state s and the expected reward r(s,a) during that period are given by:

1) when $d(X_0', X_1', X_3') \neq 0$, $\nu(s,a) = \frac{1}{d(X_0', X_1', X_3')}$

and $r(s,a) = \frac{\delta(X_1')p\alpha}{d(X_0', X_1', X_3')}$,

2) when $d(X_0', X_1', X_3') = 0$, $\nu(s,a) = \lambda^{-1}$

and $r(s,a) = 0$.

The optimal dynamic admission problem is formulated into the semi-Markov decision process with the following optimality equations [17]:

$$g_s^* = \max_{a \in A(s)} \{ \sum_{s'' \in S} P_{ss''}(a)\, g_{s''}^* \}, \qquad s \in S ,$$

and $$v_s^* = \max_{a \in B(s)} \{ r(s,a) + \sum_{s'' \in S} P_{ss''}(a)\, v_{s''}^* - \sum_{s'' \in S} P_{ss''}(a) \nu_{ss''}(a)\, g_{s''}^* \},$$

where g_s^* denotes the maximum throughput obtained by starting from the initial state s, v_s^* the relative value of state s, and $\nu_{ss''}(a)$ the expected transition time under the condition on the present state s, the action a and the transition state s", and

$$B(s) = \{ a ; g_s^* = \sum_{s'' \in S} P_{ss''}(a)\, g_{s''}^* \}.$$

Since sets S and A(s) are finite, there exists an optimal stationary Markov deterministic policy. Moreover, since for each $s = (X_0, X_1, X_2, X_3)$ and $\bar{s} = (\bar{X}_0, \bar{X}_1, \bar{X}_2, \bar{X}_3)$, s is reachable from $\bar{s}$ under some stationary policy, $g_s^* = g^*$ is satisfied [18]. Hence the optimality equations are reduced to

$$v_s^* = \max_{a \in A(s)} \{ r(s,a) + \sum_{s'' \in S} P_{ss''}(a)\, v_{s''}^* - \nu(s,a)\, g^* \}. \qquad (9)$$

Then a stationary deterministic policy f^* maximizing the right hand side of (9) is optimal.

5 Comparisons of Admission Policies

We compare the maximum throughput λ^{**} attained by the optimal static admission policy with ones given by the L=S and knee criteria. Now derive the formulas to calculate the degrees determined by the L=S and knee criteria, which are denoted by $M_{L=S}$ and M_{knee}, respectively. The L=S criterion operates by adjusting the maximum degree of multiprogramming so that the system lifetime (L) nearly equals the mean service time of paging disk (S), i.e., L=wS, where $1 \leq w < 2$. Therefore (1) and (2) imply that

$$M_{L=S} = \text{int}[T/c(\sqrt{(2b\beta/w) - 1})], \tag{10}$$

where int[x] denotes the rounded value of x, which is modified to one when $x<0.5$ or $((2b\beta/w) - 1)<0$, and to T when $x>T$. According to the knee criterion, the maximum degree M is chosen such that M corresponds to the knee point of (2). Hence, the degree M_{knee} is given by

$$M_{knee} = \text{int}[T/c]. \tag{11}$$

Then the throughputs $\lambda^{*}(M_{L=S})$ and $\lambda^{*}(M_{knee})$ attained by the L=S and knee criteria are computed by (8).

Suppose that the parameters are set as follows:

α =1.0, β =0.6, T=6, p=0.125, q=0.875, b=1.0, c=2 and w=1.5.

Then the throughput $\lambda^{*}(M)$ can be calculated by (8) for each M=1,2,...,6, and the optimal degree M^{*} is equal to 2. $M_{L=S}$ and M_{knee} can also be calculated from (10) and (11), and $M_{L=S}$ =1 and M_{knee} = 3. At these degrees,

$\lambda^*(1)=\lambda^*(M_{L=S})=0.06490$ (L=S criterion),

$\lambda^*(2)=\lambda^{**}=0.07542$ (optimal degree)

and $\lambda^*(3)=\lambda^*(M_{knee})=0.06756$ (knee criterion).

β	optimal degree		L=S criterion		knee criterion	
0.2	0.03309	(1)	0.03309	(1)	0.02484	(3)
0.4	0.05782	(2)	0.05233	(1)	0.04803	(3)
0.6	0.07542	(2)	0.06490	(1)	0.06756	(3)
0.8	0.08752	(2)	0.07377	(1)	0.08266	(3)
1.0	0.09594	(2)	0.09594	(2)	0.09375	(3)
1.2	0.10195	(2)	0.10195	(2)	0.10171	(3)
1.4	0.10740	(3)	0.10740	(3)	0.10740	(3)
1.6	0.11150	(3)	0.11150	(3)	0.11150	(3)
1.8	0.11447	(3)	0.11107	(4)	0.11447	(3)
2.0	0.11667	(3)	0.11441	(4)	0.11667	(3)
2.2	0.11831	(3)	0.11686	(4)	0.11831	(3)
2.4	0.11956	(3)	0.11868	(4)	0.11956	(3)
2.6	0.12053	(3)	0.11689	(5)	0.12053	(3)
2.8	0.12128	(3)	0.11873	(5)	0.12128	(3)
3.0	0.12188	(3)	0.12011	(5)	0.12188	(3)
3.2	0.12245	(4)	0.12117	(5)	0.12235	(3)
3.4	0.12292	(4)	0.11908	(6)	0.12274	(3)
	:		:		:	
5.4	0.12459	(4)	0.12434	(6)	0.12435	(3)
5.6	0.12464	(4)	0.12445	(6)	0.12441	(3)
5.8	0.12469	(5)	0.12454	(6)	0.12447	(3)
6.0	0.12473	(5)	0.12462	(6)	0.12452	(3)
	:		:		:	
9.0	0.12496	(5)	0.12496	(6)	0.12485	(3)
9.2	0.12496	(5)	0.12496	(6)	0.12486	(3)
9.4	0.12497	(6)	0.12497	(6)	0.12487	(3)
	:		:		:	
12.0	0.12499	(6)	0.12499	(6)	0.12493	(3)

(M):degree of multiprogramming

Table 1 Throughputs under the optimal degree M^* and two conventional criteria $M_{L=S}$ and M_{knee} for disk speed β ranging from 0.2 to 12.0

This shows that M^* gives a 10 percent higher throughput than $M_{L=s}$ and M_{knee}.

Table 1 shows the values of λ^{**} as paging disk service rate β changes from 0.2 to 12.0 by 0.2. The table also contains the values of corresponding $\lambda^*(M_{L=s})$ and $\lambda^*(M_{knee})$. It shows that as the disk speed becomes higher, the optimal degree comes close to the maximum possible degree T. On the contrary, when the disk speed becomes lower, the degree becomes 1 to avoid thrashing.

We numerically compare λ^{**} with the maximum throughput g^*. The maximum throughput g^* and an optimal policy f^* can be determined by solving (9) with fixed λ and N_o using the policy iteration method (PIM) [12]. The maximum throughputs remain the same to the accuracy of 10^{-10} when λ =10.0 ($>p\alpha$ =0.125) and N_o is larger than or equal to 8. Therefore, we set λ = 10.0 and N_o=8, and solve the problem to obtain g^* = 0.07601. The difference between g^*=0.07601 and λ^{**}=0.07542 shows that for multiprogrammed computer systems, it is effective to dynamically control the admittance of jobs into the inner system.

The values of g^* and λ^{**} are computed as paging disk service rate β changes from 0.2 to 12.0. Improvement of g^* over λ^{**} can be represented by

$$100\ (g^* - \lambda^{**}) / \lambda^{**} \text{ percent,}$$

which is called efficiency. These numerical results are given in Table 2. This table shows that when the paging disk serves jobs at very low or high disk speed, the efficiency is low, and it is maximized at β =1.2. This is because when the disk speed is very slow, the optimal policy is to restrict the number of jobs in the inner system to 1 to avoid thrashing, and as shown in Table 1, M^*

is equal to 1. On the contrary, when the speed is very high, thrashing does not occur, even if the jobs are processed at the maximum possible degree 6. The optimal policy accepts jobs up to the maximum possible degree.

β	g *	λ **	efficiency
0.2	0.03309	0.03309	0.00(%)
0.4	0.05830	0.05782	0.85
0.6	0.07601	0.07542	0.79
0.8	0.08812	0.08752	0.69
1.0	0.09560	0.09594	0.59
1.2	0.10383	0.10195	1.85
1.4	0.10933	0.10740	1.79
1.6	0.11320	0.11150	1.53
1.8	0.11596	0.11447	1.30
2.0	0.11796	0.11667	1.11
2.2	0.11955	0.11831	1.05
2.4	0.12097	0.11956	1.18
2.6	0.12198	0.12053	1.21
2.8	0.12271	0.12128	1.18
3.0	0.12324	0.12187	1.12
3.2	0.12363	0.12245	0.96
3.4	0.12392	0.12292	0.81
3.6	0.12414	0.12328	0.69
3.8	0.12431	0.12358	0.59
4.0	0.12444	0.12381	0.51
	:	:	:
5.0	0.12478	0.12446	0.25
6.0	0.12490	0.12473	0.13
7.0	0.12495	0.12478	0.06
8.0	0.12497	0.12493	0.03
9.0	0.12498	0.12496	0.02
10.0	0.12499	0.12498	0.01
11.0	0.12499	0.12499	0.00
12.0	0.12499	0.12499	0.00

Table 2 Efficiency of optimal dynamic admission policy

b	β	maximum efficiency
6.0	0.2	5.20 (%)
4.0	0.3	4.43
3.0	0.4	3.85
2.0	0.6	3.06
1.0	1.2	1.85
0.6	2.1	1.35
0.5	2.6	1.13
0.4	3.1	0.97
0.2	6.2	0.52
0.1	12.3	0.27

Table 3 Maximum efficiency and the value of β attaining it under fixed values of parameter b

Table 3 shows the maximum efficiency with respect to disk speed β under the fixed parameter b. In Fig. 2, the efficiency is plotted against disk speed β under the fixed values of b=2, 4 and 6, respectively. It is shown that when b is large, that is, the page fault occurs hardly, the slow disk speed attains the maximum efficiency.

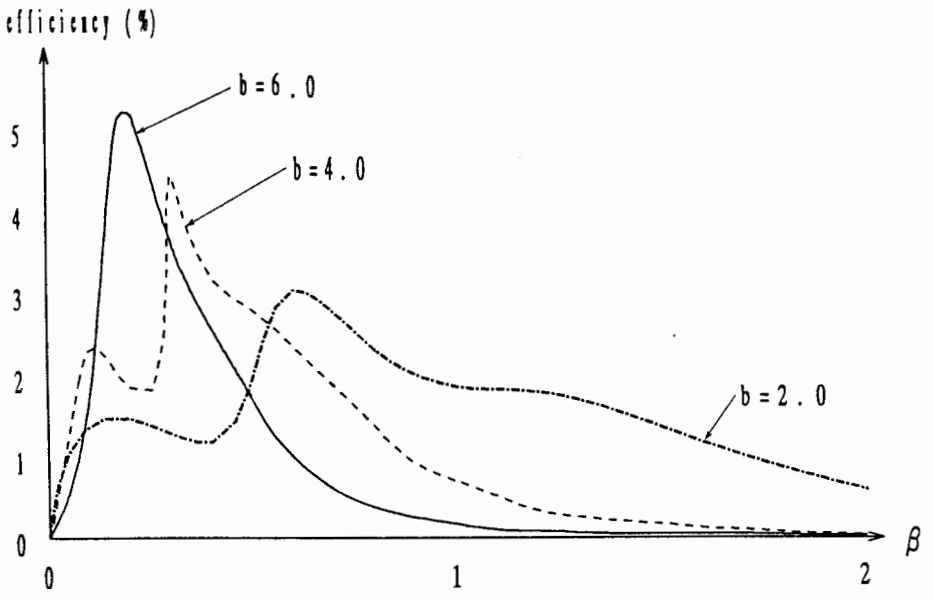

Fig.2 Efficiency against disk speed β under fixed values of parameter b

6. Conclusion

We deal with the optimal static and dynamic admission policies that maximize the throughput of the multiprogrammed computer system, and show how the throughput can be improved by

the optimal dynamic admission policy.

In fact, it is shown in Table 1 that the optimal static admission policy is superior to the L=S and knee criteria by a ten percent in a few cases. Table 3 shows that the optimal dynamic admission policy attains a five percent higher throughput than the optimal static admission policy in the system with low speed paging disk.

The above results are derived for the simple system in Fig. 1. The superiority of the dynamic admission policy will hold for practical computer systems. It is hoped that the dynamic admission policy is adopted in actual computer systems to improve their performance.

Acknowledgment

The authors would like to express their appreciation to Professor H. Kameda of The University of Electro-Communications for his helpful advice to their paper. They also are much indebted to an anonymous referee for his helpful and instructive comments.

References

1. Belady, L. A. (1966) "A Study of Replacement Algorithms for Virtual Storage Computers," IBM Syst. J., vol. 5, 78-101.
2. Blake, R. (1982) "Optimal Control of Thrashing," Performance Evaluation Review, vol. 11, 1-10.
3. Chamberlin, D., Fuller, S. and Liu, L. (1973) "An Analysis of Page Allocation Strategies for Multiprogramming Systems

with Virtual Memory," IBM J. Res. Dev., vol. 17, 404-412.

4. Chanson, S. T., Puterman, M. L. and Wong, W. C. M. (1989) "A Markov Decision Process Model for Computer System Load Control," INFOR, vol. 27, 387-402.
5. Coffman, E. G. and Hofri, M. (1986) "Queueing Models of Secondary Storage Devices," Queueing Systems, vol. 2, 129-168.
6. Denning, P. J. (1968) "Thrashing: Its Causes and Prevention," Proc. AFIPS FJCC 33, 915-922.
7. Denning, P. J. (1980) "Working Sets Past and Present," IEEE Trans. on Software Engineering, vol. 6, 64-84.
8. Denning, P. J., Kahn, K. C., Leroudier, J., Potier, D. and Suri, R. (1976) "Optimal Multiprogramming," Acta Informatica, vol. 7, 197-216.
9. Gelenbe, E. and Mitrani, I. (1980) Analysis and Synthesis of Computer Systems, Academic Press, London.
10. Hatfield, D. J. and Gerald, J. (1971) "Program Restructuring for Virtual Memory," IBM Syst. J., vol. 10, 168-192.
11. Heyman, D. P. and Sobel, M. J. (1984) Stochastic Models in Operations Research, Volume II, McGraw-Hill, Inc.
12. Howard, R. A. (1960) Dynamic Programming and Markov Processes, The M.I.T. Press, Cambridge.
13. Kameda, H. (1984) "Optimality of a Central Processor Scheduling Policy for Processing a Job Stream," ACM Trans. on Computer Systems, vol. 2, 78-90.
14. Kameda, H. (1986) "Effects of Job Loading Policies for Multiprogramming Systems in Processing a Job Stream," ACM Trans. on Computer Systems, vol. 4, 71-106.
15. Kobayashi, H. (1978) Modeling and Analysis - an Introduction

to System Performance Evaluation Methodology, Addison Wesley, Reading.

16. Neuts, M. F. (1981) Matrix-Geometric Solutions in Stochastic Models - an Algorithmic Approach, The John Hopkins University Press, Baltimore.
17. Schweitzer, P. J. and Federgruen, A. (1978) "The Functional Equations of Undiscounted Markov Renewal Programming," Mathematics of Operations Research, vol. 3, 308-321.
18. Thomas, L. C. (1979) "Connectness Conditions used in Finite State Markov Decision Process," Journal of Mathematical Analysis and Applications, vol. 68, 548-556.