March 1981

Report No. STAN-CS-81-849

Also numbered:
CSL TR-205

Experiments on the Knee Criterion in a Multiprogrammed Computer System

by

Tohru Nishigaki

Department of Computer Science

Stanford University
Stanford, CA 94305

Experiments on the Knee Criterion in a Multiprogrammed Computer System

Tohru NISHIGAKI

Systems Development Laboratory, Hitachi, Ltd.

(presently a visiting scholar at

Computer Systems Laboratory, Stanford University)

ABSTRACT

Although the effectiveness of the Knee Criterion [7] as a virtual memory management strategy is widely accepted, it has been impossible to take advantage of it in a practical system, because little information is available about the program behavior of executing jobs.

A new memory management technique to achieve the Knee Criterion in a multiprogrammed virtual memory system is developed. The technique, termed the Optimum Working-set Estimator (OWE), abstracts the programs' behavior from their past histories by exponential smoothing, and modifies their working set window sizes in order to attain the Knee Criterion.

The OWE method was implemented and investigated. Measurements demonstrate its ability to control a variety of jobs. Furthermore the results also reveal that the throughput improvement is possible in a space-squeezing environment. This technique is expected to increase the efficiency of multiprogrammed virtual memory systems.

INDEX TERMS

virtual memory; storage allocation; memory management; multiprogramming; working set;

program behavior; operating system; resource scheduling

CONTENTS

1. Introduction

Considerable work has been done related to the performance of virtual memory management as summarized by Smith [24]. A large part of it, however, assumes a uniprogramming environment as a basis of its argument, either explicitly or implicitly. The major reason for the uniprogramming assumption is the simplification of the analysis. In a multiprogramming system, the optimality in memory allocation to a specific program cannot be discussed solely in terms of its own page fault rate. Various performance factors such as CPU/channel service rate, CPU/channel scheduling policy, the behavior of other programs in a multiprogramming set, etc., must also be taken into consideration. These factors make it virtually impossible to obtain a simple and straightforward solution. Despite that, it is obvious that multiprogramming is one of the most common features of actual virtual memory systems. There is therefore an urgent need for the study of this area.

Several papers have already addressed this problem of multiprogrammed memory management. In addition to the comparative analyses of various memory management strategies by Denning [6] and Masuda [16], some noteworthy rules intended for use in a multiprogramming system are given by Belady [1], Denning [7],[8] and Leroudier [14]. Above all, the "Knee Criterion" and the "$L=S$ Criterion" proposed by Denning [7],[8] are among the most important. The Knee Criterion is a memory allocation strategy which achieves the maximum ratio of the lifetime to the memory allotment for each program in a multiprogramming set. It can be easily shown that this corresponds to the minimization of each program's *space time product* due to paging [7]. On the other hand, the $L=S$ Criterion keeps the averaged lifetime at least as great as the page transfer time for a page fault. Obviously, the $L=S$ Criterion considers the *averaged* behavior of the programs rather than the behavior of *each* program in a multiprogramming set. Since program behavior is intrinsic to each program, a memory allocation mechanism will be more stable if it is based on each program's own behavior. The Knee Criterion considers individual programs and some simulation reports and theoretical arguments show its excellence [7],[8],[13]. However, the problem of Knee Criterion is its difficulty in implementation. This is because it essentially requires information of program behavior in advance of program execution, and

such information is not generally available. The lifetime curves of the jobs to be processed are hardly known in a practical environment. Consequently, to our knowledge, no experimental reports on the Knee Criterion have yet appeared.

This paper presents some results of our experiments with the Knee Criterion. A new virtual memory management technique is developed, which is termed the *Optimum Working-set Estimator* (*OWE*). The OWE enables the approximate implementation of the Knee Criterion in the following way. First the OWE *abstracts* the characteristics of each program's memory reference behavior from its past history by exponential smoothing. Secondly it modifies the window size of each program's working set [4] in such a way as attains the Knee Criterion. The OWE, therefore, constitutes a *multi*-window-size working set scheduler. In this respect the OWE is quite different from conventional implementations of *uni*-window-size working set schedulers in which there is only one window size which is shared by all programs in a multiprogramming set [2],[22]. It should be noted that the OWE, since it employs an abstraction (learning) algorithm, addresses primarily large jobs with a long CPU execution time, which will have a significant influence on the system throughput.

First, in section 2., we discuss the relation between the Knee Criterion and other control rules from the view point of throughput maximization. Although the Knee Criterion has so far been associated solely with program's *lifetime*, an attempt is made to extend the notion and associate it with the concept of program's *processor efficiency* defined by Belady [1]. It is shown that the Knee Criterion applied to processor efficiency curve is the minimization of the space time product due to processing as well as to paging. In section 3. the OWE is presented, showing how the Knee Criterion can be implemented in a practical environment. The experimental results are shown in section 4. and 5. In section 4., the ability of the OWE to achieve system optimality is investigated by examining the space time product. For this purpose, a synthetic workload is employed of which the minimum of the space time product is theoretically calculable. The overhead due to the OWE is also considered. Finally, measurements on the throughput of the benchmark workload are carried out in section 5., to reveal the improvements when the OWE is applied.

2. **Optimality** in Multiprogrammed Memory Mnagement

The lifetime of a program is defined to be the mean virtual time between page faults. The lifetime function $e(z)$ of a given program is illustrated in Fig.la, where the lifetime is specified in terms of page references when the program's resident set size averages z pages. Although the lifetime curve might take various shapes, it is generally known to have concave region following convex region as z increases [8]. The critical point corresponding to d^2e/dz^2 of 0 (max de/dz) is called the *parachor,* which identifies the core requirement z_1 to achieve reasonable processing [1]. The *knee,* on the other hand, is defined geometrically as the highest point of tangency between a ray from the origin and the curve [7]. The Knee Criterion is the rule to keep the system operating at the knee [7],[8]. In other words, the Knee Cri'tcrion corresponds to the memory allotment z_2 which maximizes the cost performance ratio e/z, where the Jifetimc e and the memory size z each arc considered to be *gain* and *investment* respectively.

It is possible to think of another *gain,* which is shown in Fig.1b. Belady [1] proposed as a measure the *processor efficiency* of a program $e/(e+S)$, and discussed that memory allotment should be at least the point z_3 yielding $d^2\{e/(e+S)\}/dz^2$ of 0 (max $d\{e/(e+S)\}/dz$)[9]. Here S denotes the mean processing delay for a page fault. By substituting the processor efficiency for the lifetime, a new type of Knee Criterion can be obtained. Apparently this memory allotment z_4 yields the maximum performance ratio of $\{e/(e+S)\}/z$. To avoid confusion, we denote the new one as the Processor Efficiency Knee (PE-Knee) Criterion and the traditional one as the LifeTime Knee (LT-Knee) Criterion hereafter.

It is noteworthy that the satisfaction of either of these Knee Criterions achieves the minimization of the *space time product* of the program. Letting f be page fault rate, the space time product due to program execution of V references is given by ($Vz + SVfz$) [1],[3],[11],[21],[23]. The space time product per reference X, and that due to only paging Y, arc obtained as follows:

$$X = z + Y. \tag{2.1}$$

$$Y = Sfz. \tag{2.2}$$

Sincefequals to $1/e$, the next relations hold.

$$1/X = \{e/(e+S)\}/z. \tag{2.3}$$

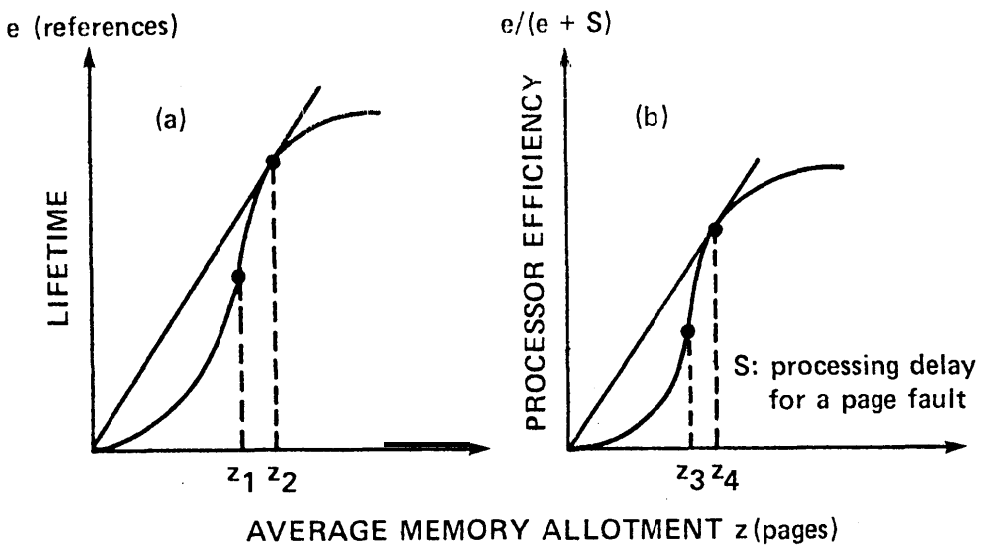

Fig.1 The memory allotments under the LifeTime Knee (LT-Knee) Criterion and the Processor Efficiency Knee (PE-Knee) Criterion.

$1/Y = (1/S)\, e/z.$ *(2.4)*

The above formulae show that the PE-Knee Criterion and the LT-Knee Criterion each represent the rule to minimize Xand Y respectively. In most cases the LT-Knee Criterion is expected to yield larger memory allotment than that of the PE-Knee Criterion [20].

The appropriateness of memory management, however, does not ensure the sufficient condition to attain the throughput maximization in a multiprogramming system. Apparently it is the management of a critical (highly utilized) resource that has significant influence on the throughput. For instance, in a system with plenty of memory and a very slow CPU, the throughput depends more on the management of CPU than that of memory. That is, high dispatching priority assignment to I/O bound jobs can improve the throughput regardless of memory management [17]-[19].

In spite of that, the following argument shows that the Knee Criterions contribute to throughput improvement. Denoting the lifetime of program i in a multiprogramming environment as e_i, the upper limit of CPU utilization by i is given by $e_i/(e_i+S)$ [1]. Consequently, the total CPU efficiency u_{cpu} satisfies formula (2.5), where the summation is taken over all programs in a multiprogramming set.

$$u_{cpu} \leq \min\{\ 1,\ \sum_i e_i/(e_i+S)\ \}. \qquad (2.5)$$

This shows that the increase in $\sum_i e_i/(e_i+S)$ is necessary to achieve high CPU efficiency.

The u_{cpu} has another upper limit. Let L be the *system lifetime* — the average CPU execution time between two successive page faults. Note that L is the mean time between page faults averaged over *all* programs rather than *a particular* program in a multiprogramming set. By considering a simple queueing model in an equilibrium state, the following relation can be obtained. (Refer to [7], [20] for the proof.)

$$u_{cpu} \leq \min\{\ 1,\ L/S\ \}. \qquad (2.6)$$

This formula gives the theoretical reasoning to the "$L=S$ Criterion" proposed by Denning [7],[8], which is to keep L just above S. Thus we obtain,

$$u_{cpu} \leq \min\{\ 1,\ \sum_i e_i/(e_i+S),\ L/S\}. \qquad (2.7)$$

The system lifetime L is a weighted *mean* of the program's lifetime e_i, where the weight is determined by its page fault generating probability. This probability depends upon not only intrinsic program behavior but also other factors such as CPU allocation policy, I/O activities of all programs, etc. Therefore it is not generally known which of the two limiting factors —— $\sum_i e_i/(e_i+S)$ or L/S —— actually constraints the system performance. Since the PE-Knee Criterion and the LT-Knee Criterion each require knowledge about $e_i/(e_i+S)$ and e_i, this argument suggests that a simple and general remark about the comparison between the two Knee Criterions can hardly be obtained.

3. Optimum Working-set Estimator (OWE)

The strict implementation of the Knee Criterion is impossible without knowing the lifetime curve of a program in advance of its execution. The fact that this information is generally unavailable has prevented the Knee Criterion from being implemented in current operating systems. However, if a *long* job with a significant CPU execution amount shows *stationary* behavior (after having passed through its initial unstable phases), it is possible to estimate the lifetime curve based on its past history. This stability in program behavior is observed in many jobs as exemplified by Hatfield [12], although there is no proof of it. These long jobs need especially to be properly controlled, since they will have a great influence on the system throughput.

The Optimum Working-set Estimator (OWE) is an approximate technique to realize the Knee Criterion. Conceptually the OWE constitutes of two algorithms; one is the abstraction (learning) algorithm to estimate the program's lifetime/processor efficiency curve, and the other is the optimization algorithm to make memory allotment be done at the knee point. For this purpose, the OWE works on the basis of a working set scheduler [4], which is one of the most widely-known program driven memory management schedulers. In other words, the OWE determines each program's window size to realize the memory allotment at the *knee* point.

It should be noted that working set schedulers, despite of their capability to estimate program's locality, have so far been implemented solely as *uni*-window-size schedulers. All programs in a

multiprogramming set share the window size, which is either fixed [22] or dynamical19 adjusted according to paging load [2]. They therefore have failed to be optimal in terms of the space time product. Below is described the OWE, an attempt to achieve minimization of each program's space time product.

Let the sample points on program's virtual time be t_n (n= 1,2,. . .), which are chosen at every time quantum of A. Although page replacement in a strict working set scheduler might be done at every reference instance, it is done in our implementation only at sample points to avoid excessive overhead. That is, the pages having been unreferenced since (t_n-T-1) are replaced at t_n, where the window size T is assumed multiple of A. Denoting the working set at t_n with window size Tas $W(t_n, T)$, this approximation is expressed as follows:

$$W(t_n+\varepsilon,T) \simeq W(t_n+\varepsilon,T+e). \qquad (3.1)$$

$$(0 \leq \varepsilon < \Delta, n=1,2,\ldots)$$

Let the working set size at t_n be $w(t_n, T)$, and the average working set size be $z(T)$. The estimator of $z(T)$ at t_n is defined by eq.(3.2) and denoted as $\zeta(t_n, T)$.

$$\begin{cases} \zeta(t_n,T) = (1-\alpha)w(t_n,T) + \alpha\zeta(t_{n-1},T). \\ \zeta(t_0,T) = w(t_1,T). \end{cases} \qquad (3.2)$$

$$(0<\alpha<1,\ n=1,2,\ldots)$$

This formula can bc also written as

$$\zeta(t_n,T) = (1-\alpha)\sum_{j=0}^{n-1} w(t_{n-j},T)\alpha^j + \alpha^n w(t_1,T). \qquad (3.3)$$

Namely ζ is an unbiased estimator of z, which is given as a weighted sum of the working set sizes measured in the past at every time quantum of A. The weights are chosen so as to, more or less, reflect the changes in program behavior.

The following relation holds bctwccn the page fault ratcfand average working set size z [5].

$$f = dz/dT. \qquad (3.4)$$

This leads us to the following definition of $\varphi(t_n, T)$, which is the estimator of f at t_n.

$$\varphi(t_n,T) = \{\zeta(t_n,T+\Delta) - \zeta(t_{n-1},T)\}/\Delta . \qquad (3.5)$$

$$(n = 1,2,\ldots)$$

Let the number of the page faults gcncratcd from t_{n-1} to t_n with the window size of T bc $g(t_n, T)$. Under

the assumption of eq.(3.1), $g(t_n, T)$ is calculated as

$$g(t_n, T) = w(t_n, T+\Delta) - w(t_{n-1}, T). \tag{3.6}$$

By using eqs.(3.3),(3.5),(3.6), we obtain

$$\begin{aligned}
\varphi(t_n, T) &= [\,(1-\alpha)\sum_{j=0}^{n-1} w(t_{n-j}, T+\Delta)\alpha^j + \alpha^n w(t_1, T+\mathrm{A}) \\
&\quad -(1-\alpha)\sum_{j=0}^{n-2} w(t_{n-j-1}, T)\alpha^j - \alpha^{n-1} w(t_1, T)\,]/\Delta \\
&= [\,(1-\alpha)\sum_{j=0}^{n-2} \{w(t_{n-j}, T+\mathrm{A}) - w(t_{n-j-1}, T)\}\alpha^j + (1-\alpha) w(t_1, T+\Delta)\alpha^{n-1} \\
&\quad + \alpha^n w(t_1, T+\mathrm{A}) - \alpha^{n-1} w(t_1, T)\,]/\Delta \\
&= (1-\alpha)\sum_{j=0}^{n-2} \{\, g(t_{n-j}, T)/\Delta \,\}\alpha^j \\
&\quad + \alpha^{n-1}\{\, w(t_1, T+\Delta) - w(t_1, T)\,\}/\Delta
\end{aligned} \tag{3.7}$$

This formula shows that $\varphi(t_n, T)$ is approximately the weighted sum of the page fault rates measured at every time period of A when the window size is set at T.

The estimates of the space time products X(7) and Y(T) at t_n are denoted each as $\chi(t_n, T)$ and $\psi(t_n, T)$, and are defined as follows:

$$\chi(t_n, T) = \zeta(t_n, T) + S\varphi(t_n, T)\zeta(t_n, T). \quad (n = 1,2, \ldots) \tag{3.8}$$

$$\psi(t_n, T) = S\varphi(t_n, T)\zeta(t_n, T). \quad (\mathrm{n} = 1,2, \ldots) \tag{3.9}$$

The approximate implementations of the PE-Knee Criterion and the LT-Knee Criterion each can be realized by the minimization of $\chi(t_n, T)$ and $\psi(t_n, T)$ respectively. It is indicated in eqs.(3.2),(3.5),(3.8),(3.9) that $\chi(t_n, T)$ and $\psi(t_n, T)$ can be evaluated at every t_n by measuring $w(t_n, T)$ as a function of T.

In order for the measurement, a table is set up whose entries are called unreferenced interval counters. Each page is associated with one of those counters, which indicates the virtual time elapsed since the last reference to the page. At every t_n ($n = 1,2, \ldots$), each page is checked whether or not it was referenced during the interval $\{ t_{n-1}, t_n \}$. This information is always available for a computer with reference bits. If it was referenced, the corresponding unreferenced interval counter is zero-cleared; if not, the counter is added by one. The evaluation of $w(t_n, T)$ is done at every window size interval of A.

That is, the measurement points Tm are given as follows:

$$Tm = m\Delta . \qquad (m = 1,2,\ldots) \tag{3.10}$$

It is easily seen that $w(t_n, T_m)(m=1,2,\ldots)$ is obtained as the number of discrete pages whose unreferenced interval counters are less than or equal to m.

The actual measurement mechanism in a multiprogramming system is based on both the real time and the virtual time of each program in a multiprogramming set. The OWE becomes active at regular real time intervals and calculates the CPU execution amount since its last activation for each program by using the program's accumulated CPU execution time. If a program has executed more than A, the unreferenced interval counters for the pages in its working set are checked and $w(t_n, T_m)$ ($m = 1,2,\ldots$) are measured. (For simplicity, those pages shared by two or more programs are not considered in this paper.) Note that the errors in the sampling of every A do not exceed the OWE's activation interval.

The above description shows that it is possible to chose the optimal window size T for each program so as to attain $\min_m \chi(t_n, T_m)$ and/or $\min_m \psi(t_n, T_m)$ at every t_n. However, it should be noted that the obtained window size might be a *local* optimal value, since it is to be chosen from a limited range. Denoting the window size set at t_{n-1} as T_M, the pages whose unreferenced interval counters are $(M+1)$ or more are all replaced at t_n- I. Therefore we can measure solely $w(t_n, T_m)(m=1,2,\cdots,M+1)$, and $\chi(t_n, T_m)$ and/or $\psi(t_n, Tm)$ can be evaluated only in the limited range of $0 \leq T \leq T_M$.

In order to cope with the case where the minimal point is out of the above range, we introduce a heuristic technique : that is, page replacement is prohibited and the window size is increased from T_M to T_{M+} I, whenever both of the following conditions are met.

$$\min_{1 \leq m \leq M} \chi(t_n, T_m) = \chi(t_n, T_M). \tag{3.11}$$

$$\chi(t_n, T_{M-1}) >> \chi(t_n, T_M). \tag{3.12}$$

(Obviously ψ is substituted for χ in eqs.(3.11) and (3.12) if the LT-Knee Criterion is employed instead of the PI'-Knee Criterion.) Since in most cases χ and ψ are expected to vary with Tin a relatively simple manner [3],[10],[11], there is a great possibility that this heuristic leads us to a point close to the optimal one.

4. Minimization of the Space Time Product

The OWE was realized experimentally on the Virtual-storage Operating System 3 (VOS3), which is a large scale operating system for HITAC M-180/M-200H. Since the OWE is an approximate implementation of the Knee Criterion, it is necessary to examine the degree of approximation. This section describes the experimental results which, as a preliminary step of measurements, investigate the space time product of a program whose minimization is achieved by the Knee Criterion. The space time product under the OWE was compared with its theoretical minimum under the Knee Criterion. For this purpose, a set of synthetic programs was chosen as a workload, whose program behaviors are readily known.

The page references in these synthetic programs are governed by the Simple Least Recently Used stack Model (SLRUM)[25]. That is, the page whose stack distance is k in a program of N pages is, referenced with the probability of $q(k)$ $(k=1,2,\ldots,N)$. The $q(k)$'s are time-invariant and the sum of them $\sum_{k=1}^{N} q(k)$ equals to 1. We can generate programs of various stationary behaviors by assigning appropriate values to $q(k)$'s. When $q(k)$'s are given, the average working set size $z(T)$ can be calculated in the following way, as proposed by Turner [26].

Let the probability that a working set size is h be $P(h,T)$, which is the probability that h discrete pages are referenced during T. By using the following formulae [26], $P(h,T)$ $(h=1,2,\ldots,N)$ can be calculated.

$$P(h,T) = P(h-1,T-1)\sum_{k=h}^{N} q(k) + P(h,T-1)\sum_{k=1}^{h} q(k). \tag{4.1}$$

$$P(h,1) = \begin{cases} 1 : h=1 \\ 0 : h=2,3,\ldots,N. \end{cases} \tag{4.2}$$

And the average working set size $z(T)$ is given by

$$z(T) = \sum_{h=1}^{N} hP(h,T). \tag{4.3}$$

We can obtain the page fault rate $f(T)$ from $z(T)$ by using eq.(3.4), and finally calculate the space time product of eq.(2.1) and/or eq.(2.2).

Three different synthetic programs Sl-S3 were created and investigated. The characteristics of them calculated by the above procedure are shown in Fig.2. The space time product X(7') of S2 and S3 are respectively monotonously decreasing and increasing, and only S1 has its minimum at the inflextion point. They are considered representatives of three types of programs. The program sizes are all 50 pages.

The synthetic programs Sl-S3 were executed under the control of the OWE in a multiprogramming environment, and their working set sizes were observed. The OWE was invoked and executed at about 100,000 instructions of real time interval, and the virtual time sampling interval A was chosen at 200,000 instructions. Other parameters were set as follows: the smoothing parameter **a** in eq.(3.2) was 0.5, the paging delay S in eqs.(3.8),(3.9) was 80,000 instructions.

The measured data are compared with theoretical optimal values in Table 1. The data were collected and averaged at every virtual time sampling interval of 200,000 instructions over the measurement period of about 8 minutes. The degree of coincidence between the measured and optimal values is satisfactory, except for the case of S3 under the LT-Knee Criterion. The relative differences in the average working set size and the space time product are each 0.4-5.0 % and 0.8-10.2 % respectively. Even the differences in the window size, which do not look very small, are still considered to be in an acceptable range since the variance in Taround the optimum is known to make an insignificant difference in X(7) and/or Y(T) [3],[10],[11].

As fo. 33 under the LT-Knee Criterion, the degree of concordance is less satisfactory, as far as the *ratio* (*relative difference*) of the space time product $Y(T)$ is concerned. However, it should be noted that the *absolute difference* between the two is not significant, since Y(7) itself is small in this case. This was caused by the fact that the optimization algorithm of the OWE is based on *absolute* rather than *relative* value of X(T) and/or $Y(T)$. That is, the OWE judges a value of $\boldsymbol{T}$ to be almost optimal only when its modification causes small absolute differences in $X(T)$ and/or $Y(T)$. Since the system throughput is influenced not by relative but by absolute values of the space time product, this algorithm and its results are considered reasonable. In summary, Table 1 reveals experimental verification of the OWE as an

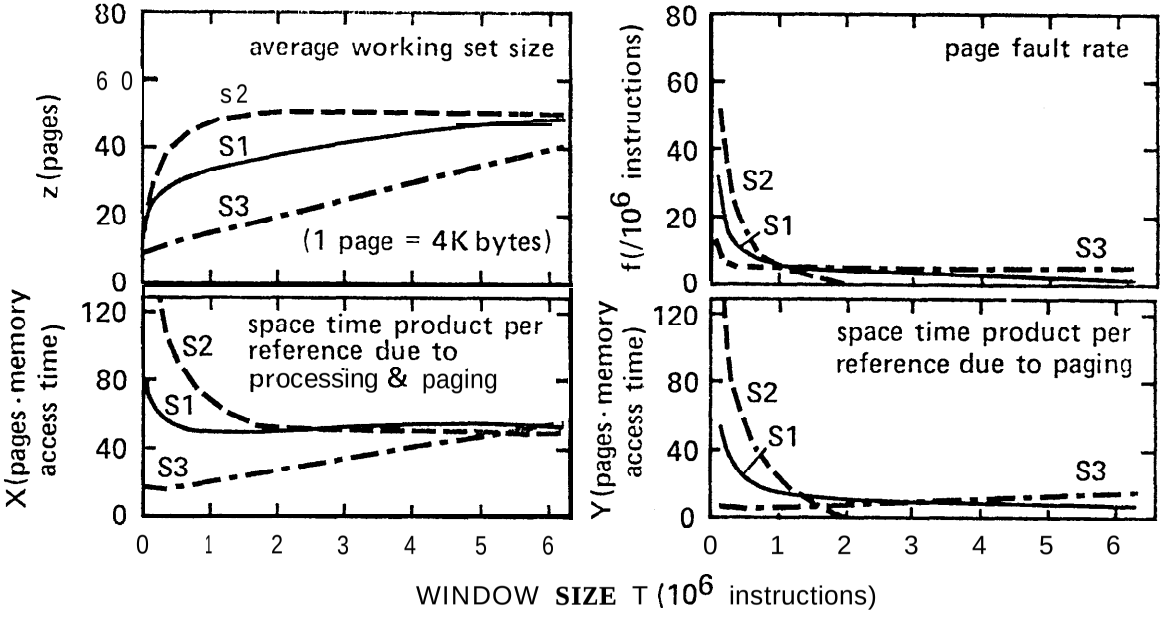

Fig.2 The space time products and other characteristics of the synthetic programs Sl-S3.

Table 1 The validation of the OWE's ability to achieve the Knee Criterion.

(upper row : measured values, lower row : optimal values)

CRITERION	PE-KNEE			LT-KNEE		
PROGRAM	S1	S2	s3	S1	s2	s3
AVERAGE WINDOW SIZE ($1\ 0^6$ instructions)	0.82 1.20	1.76 2.60	0.56 0.30	2.74 4.00	2.18 2.60	1.20 0.40
AVERAGE WORKINGSET SIZE (pages)	32.5 34.2	49.0 50.0	11.1 10.7	42.5 44.1	49.8 50.0	15.8 11.3
SPACE TIMEPRODUCT PER REFERENCE (pages- memory access time)	48.5 48.1	55.1 50.0	15.8 15.2	9.83 9.38	0.001 0.0	6.30 4.55

(1 page = 4K bytes)

approximate implementation of the Knee Criterion.

The overhead of the OWE was also examined in the measurements. It is possible to set the theoretically optimal window size throughout the measurement period not by dynamic abstraction (learning) but as a pre-fixed value, since it is already known for the synthetic workload as indicated in Fig.2. The measurements of this idealized case were carried out, and the CPU overheads of the operating system were compared with the OWE's case where the optimal window sizes were found dynamically. The increase in the CPU overhead is considered to comprise the overhead of the OWE. The overhead of the OWE is in general an increasing function of the working set size, since a large part of the OWE's processing is done for each page. We therefore carried out measurements for S2 and S3, whose working set sizes are each relatively large and small respectively. A sufficient number of S2 type programs were copied and executed for about 8 minutes in a multiprogramming environment. The overhead of the OWE in this case was 1.2% for both the PE-Knee Criterion and the LT-Knee Criterion. The same kind of measurement for S3 programs showed the overhead to be 0.35% for the PE-Knee Criterion and 0.59% for the LT-Knee Criterion respectively. The overhead for S1 is expected to lie between those for S2 and S3. These figures are considered to be quite acceptable.

The OWE is a rather stable controller. This is because, as was already mentioned, the $X(T)$ and/or $Y(T)$ are generally insensitive to the variation in T around the optimal value [3],[10],[11]. This fact also implies that, once an approximately optimal T has been found, it is not necessary to frequently modify T any more, as long as the program shows stationary behavior. That is to say, a further reduction of the overhead of the OWE is fairly promising, which will be our future work.

5. Measurements on Throughput

The decrease in the space time product is considered to have generally preferable effect on the system throughput. This section shows the results of measurements on the throughput of benchmark workload under the OWE. The object of these measurements is to investigate the degree and the required conditions of throughput improvements.

The benchmark workload used in the cxpcrimcnt comprises FORTRAN programs with compile, link, and execution phases (except for program B4 which constitutes of only an execution phase), all of which have considerable amount of CPU execution time of more than 25 sec. Their average working set sizes $z(T)$'s are approximately shown in Fig.3., which includes a peculiar linear curve along with those for benchmark programs B1-B4. This is a curve of a King Size Program added to the benchmark programs to make the analysis easier.

The King Size Program is a synthetic program which references pages only consecutively at a certain rate, and keeps this behavior by returning to the first page when it reaches the last. It is obviously preferable in terms of the space tirne product to allocate as small amount of memory as possible to this kind of program. This is true because its page fault rate is virtually independent of its memory allotment (unless the whole program is included in memory). The conventional uni-window-size working set scheduler cannot achieve this goal, since it determines a common window size shared by all programs in a multiprogramming set. The OWE, on the other hand, tends to decrease each program's space time product by adjusting each window size, so that wc can expect a higher throughput.

However, the dcgrcc of throughput improvement depends greatly upon the extent to which the memory usage is critical. The system performance is influenced by memory scheduling only in a space-squeezing environment [17]-[19], which often happens on an arrival of a job with heavy memory usage. This kind of job is not uncommon —— an array manipulation job is a typical cxamplc. The King Size Program was created as an abstract model of these jobs, and was used in the cxpcrimcnt to realize, more or less, a space-squeezing ctivironmcnt.

Throughout the cxpcrimcnts, the number of job-initiators (virtual spaces) was kept constant of 6, one of which was devoted to the King Size Program. Some B1-B4 programs were copied to create a high enough workload. Thus all the virtual spaces were kept active during the measurement period of around 10 minu tcs.

The throughput of the OWE is compared with that of a conventional uni-window-size working set

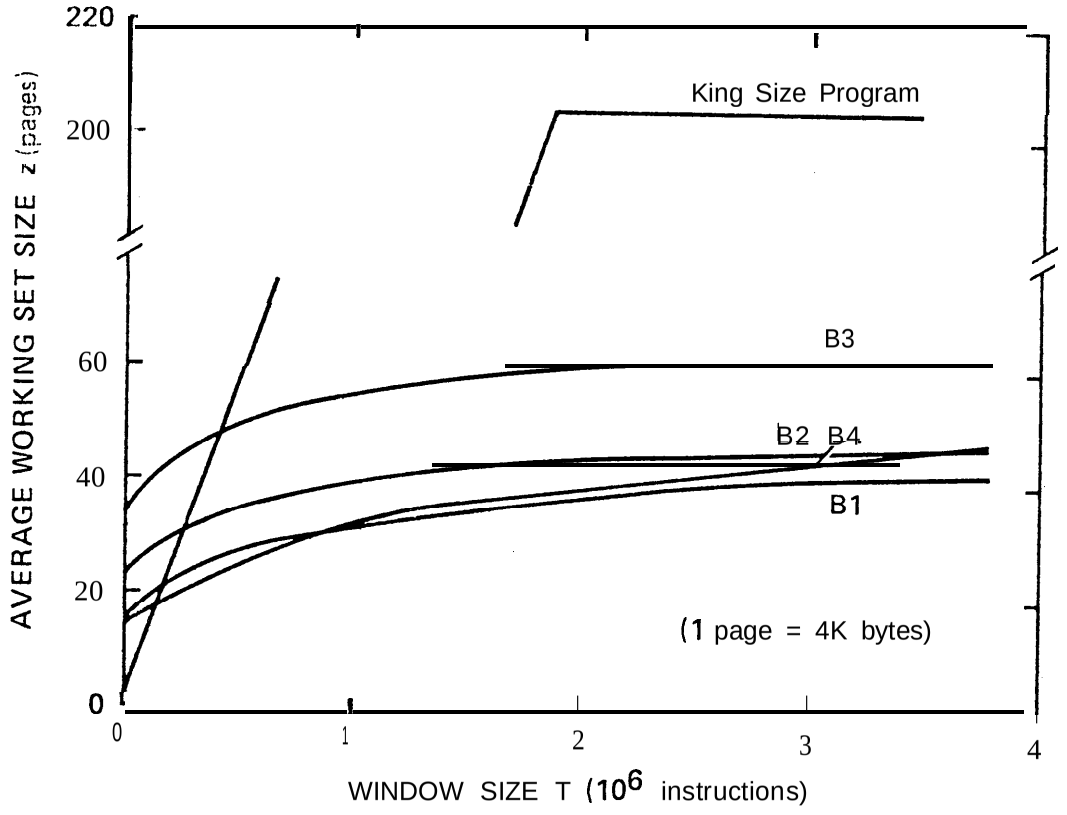

Fig.3 The approximate average working set sizes of the benchmark programs B1-B4 and the synthetic King Size Program.

scheduler in Fig.4. Although the window size in some uni-window-size working set schedulers is kept constant [22], it was modified in our experiment based on a paging load. Namely, the window size shared by all programs in a multiprogramming set was increased when the paging system was overloaded and decreased when underloaded. This adjustment is considered to increase the throughput as is suggested by the grounds of the 50% Rule [14] and the $L=S$ Criterion [7],[8].

Dcspitc that, the difference in the throughput between the OWE and a uni-window-size scheduler is clearly indicated in Fig.4. The increase in the CPU utilization by the OWE is significant for both the PE-Knee Criterion and the LT-Knee Crierion in a space-squeezing environment of the real memory capacity of 2M bytes. This improvement is not achieved for the case of the real memory capacity of 3M bytes, where memory is no more a critical resource and its allocation schedule has therefore little influence on the throughput.

The throughput diffcrcncc between uni-window-size and multi-window-size (OWE) working set schedulers is also indicated by using another measure termed the *total* service *amount*, which is the sum of the scrvicc amount of each job which was active in the measurement period. The service amount of job i, R_i, is defined as follows [15],[17]-[19]:

$$R_i = CPU_i + IO_i,$$

where CPU_i and IO_i each represent the amount of CPU scrvicc and file input/output scrvicc supplied to job i in the measurement period. An execution of 1,000 instructions and 1 file input/output. operation each constitute 1 *service unit.* The total service amount $\sum_i R_i$ is more precise throughput measure than the CPU utilization, since it includes file input/output operations and excludes the CPU overhead of an operating system. The difference between that of the OWE and the uni-window-size scheduler is 12.2-19.1 % in the space-squeezing (2M bytes) case and 0.9-1.0 % in the non-space-squeezing (3 M by tes) case.

Some additional data arc shown in Fig.5, which clarifies the reasons for these throughput improvcmcnts. The memory allotment to the King Size Program under the OWE is much smaller than that under the uni-window-size working set schccluler. In a space-squeezing environment, this causes considerable increase in a multiprogramming degree (= the number of jobs resident in a real memory).

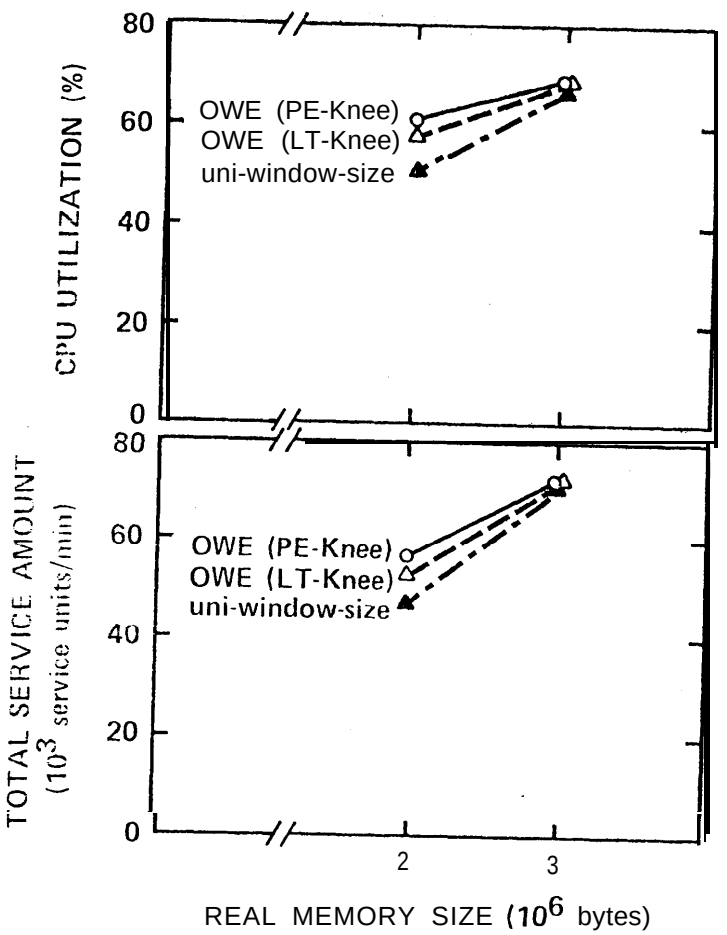

Fig.4 The throughput under the OWE (LT-Knee Criterion, PE-Knee Criterion), in comparison with that under a conventional uni-window-size working set scheduler.

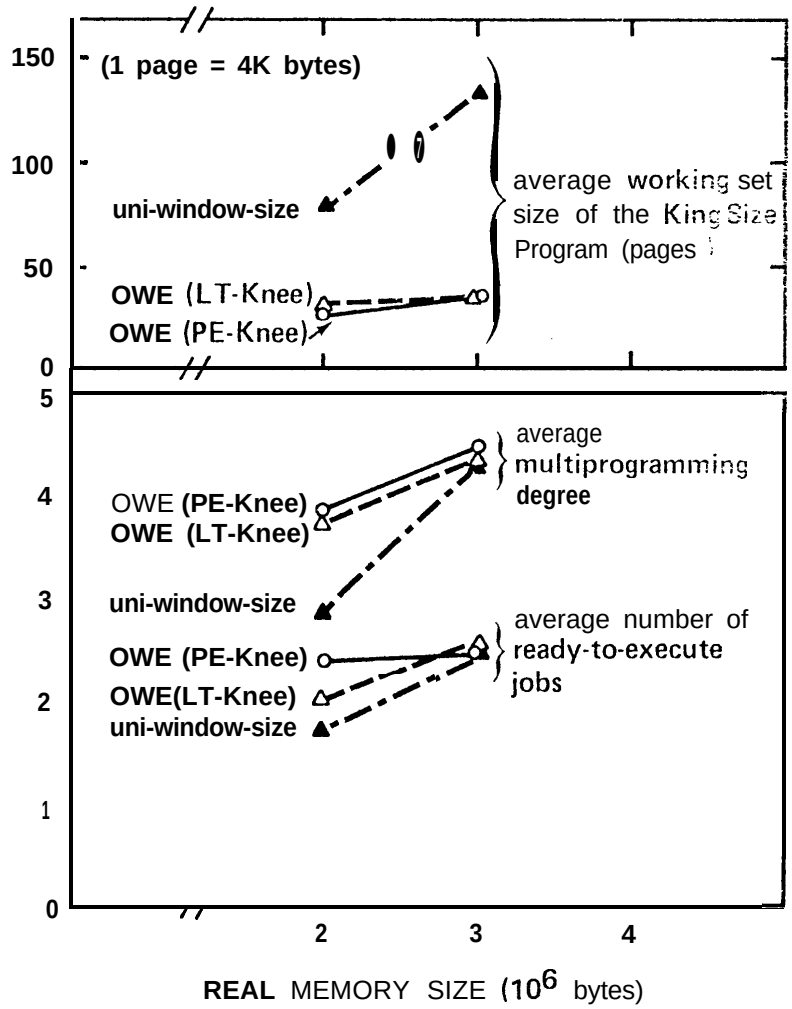

Fig.5 Some performance factors affecting the throughput; under the OWE (LT-Knee Criterion, PE-Knee Criterion) and a conventional uni-window-six working set scheduler.

Finally this leads to the increased number of the ready-to-execute jobs, resulting in the improved throughput. In a non-space-squeezing environment, however, the control of the space time product of the King Size Program scarcely improves the multiprogramming degree and the number of the ready-to-cxccute jobs, thus has no evident effect on the throughput.

It is very likely that more large scale jobs like the King Size Program emerge, as virtual memory systems become more widely used. This is a good reason that the OWE will be able to improve the system performance to a great extent. However, there some problems remain to be investigated. The comparison between the PE-Knee Criterion and the LX-Knee Criterion is among the most important. The difference is not obvious in either Fig.4 nor Fig.5, although the PE-Knee Criterion yields slightly better throughput.

6. Summary and Conclusion

An experiment on the Knee Criterion [7] was carried out, by the development of a new memory management technique termed the Optimum Working-set Estimator (OWE).

The Knee Criterion was discussed in relation to the system throughput, and the criterion was applied to the processor efficiency curve as well as the lifetime curve. Both the Processor Efficiency Knee (PE-Knee) Criterion and the LifeTime Knee (LT-Knee) Criterion correspond to the minimization of the space time product per reference, for the former due to processing and paging and for the latter only due to paging. It was shown that the decrease in the space time product has preferable effect on the system throughput.

The OWE employs an algorithm which estimates the lifetime and/or the processor efficiency of a program in execution from its past behavior. It adjusts the window sizes of individual program's working sets in such a way as attains the Knee Criterion.

An cxpcrinicntal implementation of the OWE has made it possible to carry out measurements

which are divided into 2 parts. The first part of the measurements revealed the effectiveness of the OWE in the achievement of the Knee Criterion (the space time product minimization), by using the synthetic programs whose minimal space time products are theoretically calculable. The CPU overhead of the OWE was also shown to be around 1% or less. The second part of the measurements employed FORTRAN benchmark workload and a King Size Program to investigate the throughput improvement under the OWE. The results demonstrated a significant throughput improvement (12-19 %) in a space-squeezing environment and an insignificant one (1%) in a non-space-squeezing environment, for both the PE-Knee Criterion and the LT-Knee Criterion.

The OWE scheduling algorithm is considered to be promising for the improvement of the performance of multiprogrammed virtual memory systems. There remain some problems to be investigated about the OWE, at both practical and theoretical levels. The reduction of its CPU overhead and the comparison between the PE-Knee Criterion and the LT-Knee Criterion are among the most important ones.

ACKNOWLEDGEMENT

The author would like to thank Prof. S. Ohsuga of the University of Tokyo and Prof. G. Wiedcrhold of Stanford University for their helpful comments. The author is also indebted to Dr. T. Miura of Hitachi Systems Development Laboratory and Mr. Y. Hattori of Hitachi Software Works for the opportunity of study. The support and assistance of Mr. K. Ohmachi and Mr. C. Ikeda of Hitachi Systems Development Laboratory in implementing this research has been especially appreciated.

REFERENCES

[1] L.A.Belady and C.J.Kuehner, "Dynamic Space-Sharing in Computer Systems," ***Commun. ACM***, vol.12, no.5, pp.282-288, 1969.

[2] W.W.Chiu and W.-M.Chow, "A performance of MVS," *IBM Syst.J.*, vol.17,no.4, pp.444-462, 1978.

[3] W.W.Chu and H.Opderbeck, "The page fault frequency replacement algorithm," *Proc. FJCC*, pp.597-609, 1972.

[4] P.J.Denning, "The Working Set Model for Program Behavior," *Commun.* ACM, vol.11, no.5, pp.323-333, 1968.

[5] P.J.Denning and S.C.Schwartz, "Properties of the Working-Set Model," *Commun. ACM*, vol.15, no.3, pp.191-198, 1972.

[6] P.J.Denning and G.S.Graham, "Multiprogrammed Memory Management," *Proc. IEEE*, vol.63, no.6, pp.924-939, 1975.

[7] P.J.Denning, K.C.Kahn, J.Leroudier, D.Potier and R.Suri, "Optimal Multiprogramming," *Acta Inf.*, vol.7, pp.197-216, 1976.

[8] P.J.Denning and K.C.Kahn, "An $L=S$ criterion for optimal multiprogramming," *Proc. ACM SIGMETRICS Int'l Symp. on Computer Performance Modeling, Measurement and Evaluation*, pp.219-229, 1976.

[9] T.-H.Fin and H.Kameda, "An Extention of a Model for Memory Partitioning in Multiprogrammed Virtual Memory Computer Systems," *J. Information Processing*, vol.2, no.2, pp.89-96, 1979.

[10] G.S.Graham and P.J.Denning, "On the relative controllability of memory policies," *Proc. **Int'l** Symp. on Computer Performance Modeling, Measurement and Evaluation 1977*, Amsterdam: North Holland, 1977, pp.411-428.

[11] R.K.Gupta and M.A.Franklin, "Working Set and Page Fault Frequency Paging Algorithms : A Performance Comparison," *IEEE Trans. Comput.*, vol.C-27, no.8, pp.706-712, 1978.

[12] D.J.Hatfield and J.Gerald, "Program restructuring for virtual memory," *IBM Syst.J.*, vol.10, no.3, pp.168-192, 1971.

[13] A.Krzesinski and P.Teunissen, "A Multiclass Network Model of a Demand Paging Computer System," *Acta Inf.*, vol.9, pp.331-343, 1978.

[14] J.Leroudier and D.Potier, "Principles of optimality for multiprogramming," *Proc. ACM*

***SIGMETRICS** Int'l Symp. on Computer Performance Modeling, Measurement and Evaluation*, pp.211-218, 1976.

[15] H.W.Lynch and J.B.Page, "The OS/VS2 Release 2 System Resources Manager," ***IBM Syst.J.***, vol.13, no.4, pp.274-291, 1974.

[16] T.Masuda, "Analysis of Memory Management Strategies for Multiprogrammed Virtual Storage Systems," *J.Information* ***Processing,*** vol.1, no.1, pp.14-24, 1978.

[17] T.Nishigaki, "The General Resources Manager based on Feedback Concept in Multiprogrammed Computer Systems," *Joho-Shori* (in Japanese), vol.19, no.11, pp.1026-1033, 1978.

[18] T.Nishigaki, C.Ikeda, K.Ohmachi and K.Noguchi, "An Experiment on the General Resources Manager in Multiprogrammed Computer Systems," *J. Information Processing,* vol.1, no.4, pp.187-192, 1979.

[19] T.Nishigaki, K.Noguchi and K.Ohmachi, "An Approach to the GRM Performance Analysis by Asymptotic Approximation," *J. Information Processing,* vol.3, no.2, pp.59-67, 1980.

[20] T.Nishigaki and C.Ikeda, "A Study and Experiment on the Optimum Working Set in a Virtual Memory System," *Trans.Inf.Process.Soc.Japan* (in Japanese), vol.21, no.4, pp.325-331, 1980.

[21] B.G.Prieve, "VMIN — An Optimal Variable-Space Page Replacement Algorithm," *Commun.ACM,* vol. 19, no.5, pp.295-297, 1976.

[22] J.Rodriguez-Rosell and J.-P.Dupuy, "The Design, Implementation, and Evaluation of a Working Set Dispatcher," *Commun. ACM,* vol.16, no.4, pp.247-253, 1973.

[23] A.J.Smith, "A Modified Working Set Paging Algorithm," *IEEE Trans.Comput.,* vol.C-25, no.9, pp.907-914, 1976.

[24] A.J.Smith, "Bibliography on paging and related topics," *Operating Systems Review,* vol.12, no.4, pp.39-56, 1978.

[25] J.R.Spirn and P.J.Denning, "Experiments with program locality," *Proc.FJCC,* pp.611-621, 1972.

[26] R.Turner and B.Strecker, "Use of the LRU Stack Depth Distribution for Simulation of Paging Behavior," *Commun. ACM,* vol.20, no.11, pp.795-798, 1977.