1975

The Computation and Use of Optimal Paging Curves

Peter J. Denning

Report Number:
75-154

THE COMPUTATION AND USE OF OPTIMAL PAGING CURVES

Peter J. Denning
Department of Computer Science
Purdue University
Lafayette, Indiana 47907

CSD TR 154

Submitted to Communications of ACM.

THE COMPUTATION AND USE OF OPTIMAL PAGING CURVES*

Peter J. Denning+

June 1975

ABSTRACT: A program's paging curve for a given memory management policy defines, for various resident set sizes, the mean rate at which the program references missing pages. By exploiting duality with the working set policy (WS), this paper develops the properties of the optimal variable-space memory policy (VMIN) of Prieve and Fabry. The paging curve of VMIN can be computed efficiently in a single pass over a reference string without simulating VMIN. New procedures are given for computing both the VMIN and WS paging curves with minimal storage. Examples illustrate how a design engineer can use these results to appraise existing policies relative to the optimum and to determine the best return possible under policy improvements.

Key Words and Phrases: optimal memory management, working set policies, optimal multiprogramming, storage management, storage hierarchy evaluation, paging curves, lifetime curves.

CR Categories: 4.32; 8.1.

1. INTRODUCTION

A program's paging curve for a given memory management policy defines, for various resident sizes, the mean rate at which the program references missing pages. With paging curves, a design engineer can use an analytic or simulation model to estimate important measures such as processor utilization, throughput, or mean response time at various system loads [Bar73, Bar75, DeG75]; he can construct representative synthetic workloads [SrK74]; or he can estimate instrinsic locality properties of programs [DeK75, MaB75]. For these reasons, procedures that efficiently compute paging curves of given programs are important tools for system testing and evaluation.

Design engineers are frequently interested in the behavior of systems under optimal memory policies, even if the optimal policy is unrealizable because

---

* Work reported herein supported in part by NSF Grant GJ-41289.

+ Author's address: Computer Science Department, Purdue University, West Lafayette, Indiana 47907.

Submitted to Communications of ACM.

its decisions depend on future actions of programs. Their interest may arise from a desire to understand the error inherent in the locality estimation procedure embedded in a memory policy [DeK75]. It may also arise from a desire to understand how far from optimal a system may be operating. In the latter case the difference between a system's work capacity with the current and the optimal memory policy permits the design engineer to judge the maximum tolerable investment in memory policy improvement: a proposed improvement is certainly unacceptable if its implementation cost exceeds the best possible return. For this reason, knowledge of optimal paging curves can be an important aid in evaluating changes to a system.

Prieve and Fabry recently reported on a memory policy VMIN, showing its optimality over all demand-fetch policies with respect a cost function comprising a linear combination of the virtual memory space-time and the number of page faults generated by a given program [PrF75]. The initials VMIN denote "variable space MIN", MIN being the optimal fixed-space algorithm of Belady [Bel66, BeP74]. Like its counterpart, VMIN is unrealizable. In contrast to its counterpart, VMIN's decisions depend on the future reference pattern of each page individually and not on the present contents of memory or the relative order of future references; because of this, the proof of optimality and the computation of the paging curve are considerably simpler for VMIN than for MIN.

This paper has three principal aims: to develop the properties of the VMIN policy and show their simple relation to those of the WS (working set) policy; to show how to compute the paging curves of both WS and VMIN efficiently (i.e., without simulating either) in a single scan of a reference string, and with very small storage requirements; and to demonstrate how the VMIN paging curves can be used to assess both the effectiveness of an existing memory policy and the best return possible under a policy improvement.

2. PAGING CURVES

Let $r(t)$ denote the page containing the address a given program references at the t^{th} reference, for $t=1,2,\ldots,K$. Let $Z(t)$ denote the resident set assigned by a given memory policy P to the program just after its reference $r(t)$; for convenience, $Z(0)$ is assumed empty. Let $x(t)$ denote the number of pages in $Z(t)$. The average resident set size is

$$X = \frac{1}{K} \sum_{t=1}^{K} x(t), \tag{2-1}$$

so that the virtual space-time of the program is KX. Define a binary variable $\Delta(t)$ to be 1 if $r(t)$ is missing from $Z(t-1)$, and 0 otherwise. The paging rate of the

program is

$$F = \frac{1}{K} \sum_{t=1}^{K} \Delta(t), \tag{2-2}$$

so that the total number of page faults is KF.

The pair (X,F) defined by policy P will be called an operating point of P. Typical memory policies have one or more parameters that can be adjusted to generate a set of operating points representing a range of X and F values. The paging curve of policy P is a function defined by

$$f(P,X) = F; \tag{2-3}$$

its graph is the piecewise linear curve through the set of possible operating points of P.

As suggested in Figure 1, $f(P,0)=1$ holds for all P. Also $f(P,X_m) = Q/K$ for a reference string of length K over an Q-page program, X_m being the maximal mean resident size generable by P. Most memory policies are well enough behaved that they produce nonincreasing paging curves in X; however, only for the MIN and VMIN policies is the paging curve known to be convex.

If P is a "fixed-space" policy, the resident set sizes satisfy $x(t) \leq x_0$ for some given memory space x_0. In this case, $X \leq x_0$, with equality tending for long reference strings. Typically x_0 is the parameter to P and f(P,X) is essentially the familiar page fault rate function (for fixed-space policies, the paging curve is usually rendered as $f(P,x_0)$ rather than f(P,X)). If, for example, P is the least-recent-used policy (LRU), then f(LRU,X) is a decreasing function of X and can be computed efficiently by a procedure based on counting "LRU stack distances" [MST70, CoD73].

If P is a "variable-space" policy, some parameter other than resident set size may be used to establish an operating point of P. In the case of the moving window working set policy (WS), for example, the parameter is the window size T. The working set W(t,T) is the set of pages referenced among r(t-T+1),...,r(t). For a particular value of T, the operating point is (X,F) = (s(T),m(T)), where s(T) is the mean working set size and m(T) the corresponding missing page rate [DeS72, CoD73]. In this case, f(WS,X) can be computed efficiently by a procedure based on counting "interreference intervals" of pages [CoD73, DeS72, SlT74]. Another example is the page fault frequency (PFF) policy, in which the parameter is the target page fault rate [CHO72]. The example of prime interest in this paper is

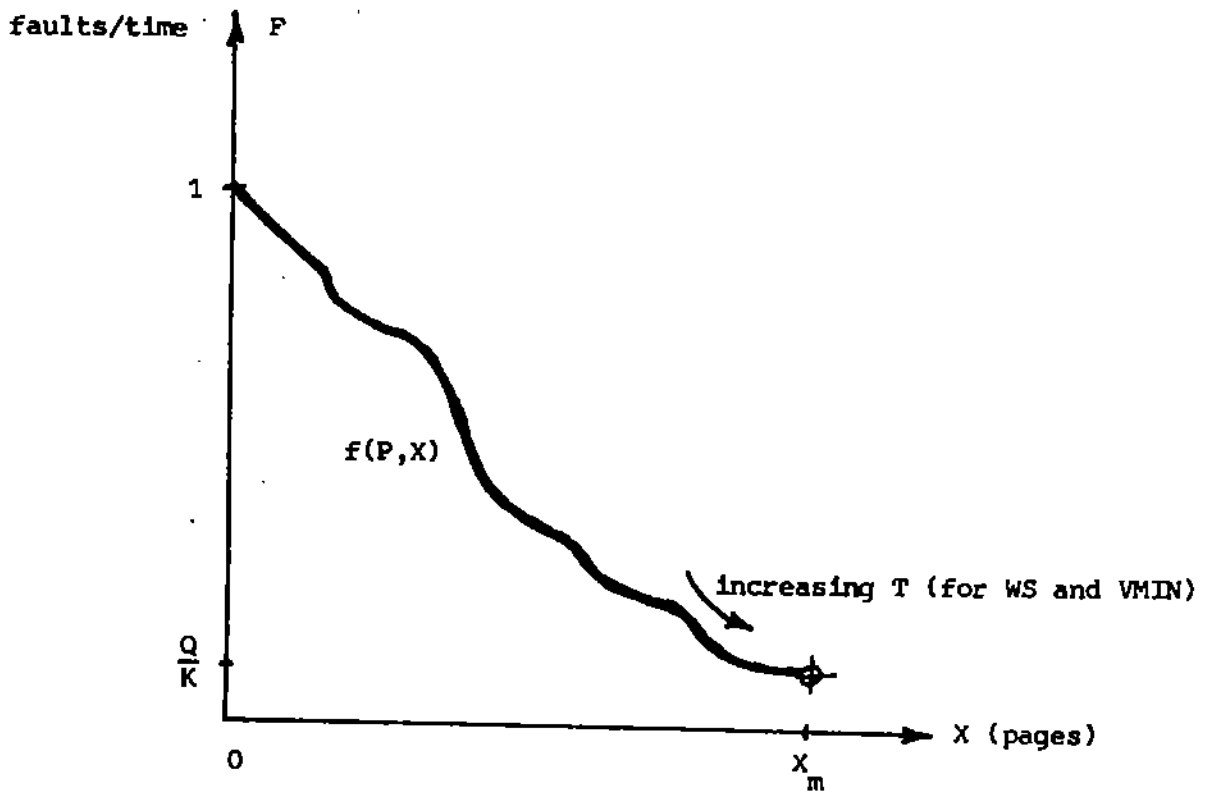

Figure 1. Paging curve.

the VMIN procedure, in which (as will be shown below) the parameter is the ratio of memory to page fault to memory costs.

Suppose that $y(t)$ denotes the forward reference interval at time t; that is, $r(t)=i$ and $y(t)=y$ imply that the earliest next reference to i is at time $t+y$. If $r(t)$ is not referenced again, $y(t)$ is infinite. For future use, we note that the distribution of $y(t)$ is precisely the interreference distribution of the reference string [DeS72].

3. THE VMIN POLICY

Prieve and Fabry define the "cost" of policy P to be a mixture of virtual space-time and page fault costs; it is of the form $a(XK)+b(FK)$, where a is the cost of one page of memory for one unit of time and b is the cost of a page fault [PrF75]. Any policy that minimizes the above cost function will minimize the cost per reference,

$$C = aX + bF. \tag{3-1}$$

The policy VMIN fetches pages on demand. Its retention rule is simple: page $r(t)$ remains in the resident set through time $t+y(t)$ if

$$y(t) \leq \frac{b}{a}, \tag{3-2}$$

and is otherwise removed immediately, where $y(t)$ is the forward reference interval at time t. The optimality of (3-2) rests on two observations. First, each page can be considered for retention or removal independently of the content of the present resident set, since VMIN is not required to observe a memory constraint. Second, since the cost function is linear, the contribution to total cost (CK) of page $r(t)$ is $ay(t)$ if that page is retained until the next reference; otherwise, the page is retained for a time $u<y(t)$ beyond time t, whereupon it contributes

$$\begin{cases} au, & \text{if } y(t) \text{ infinite,} \\ au+b, & \text{otherwise} \end{cases}$$

to total cost. Since (3-2) holds only when $ay(t) \leq b \leq au+b$, VMIN obviously makes the least cost decision at each page reference. The foregoing is a summary of a more careful argument given by Prieve and Fabry.

Whereas WS retains in the resident set precisely the pages referenced in the

backward T-window from time t, VMIN retains those referenced in the forward T-window from time t, where T = b/a. In this sense VMIN is the dual of the WS policy. The computational procedure for computing the paging curve of VMIN exploits this property.*

It is easy to see that an attempt to fetch a missing page v > 0 time units prior to its next reference will only increase the total cost by av. Therefore, VMIN is less costly than any prepaging policy.

For given ratio b/a, let (X_0,F_0) denote a VMIN operating point; and let (X,F) denote an operating point of another policy. Since $aX_0 + bF_0 \leq aX + bF$, there is associated with each VMIN operating point (X_0,F_0) an "excluded region" in which no memory policy can generate an operating point [PrF75]; see Figure 2. This implies that the paging curve f(VMIN,X) must be convex: for as shown in Figure 3, if an operating point (X_0,F_0) lay above a chord connecting two other operating points, at least one of the two others would lie in the excluded region of (X_0,F_0). Moreover, f(VMIN,X) must be strictly decreasing in X, since an operating point (X,F) with $X<X_0$ and $F\leq F_0$ would lie in the excluded region of (X_0,F_0).

Because the VMIN paging curve defines the convex hull of feasible operating points of all memory policies, it follows that

$$f(VMIN,X) \leq f(P,X) \tag{3-3}$$

for all X and any policy P. The cost function C=aX+bF consequentially minimizes the paging curve.

It should be noted that the convex shape of the optimal paging curve is a property of any policy minimizing the given cost function C = aX+bF. VMIN is one such policy. Since VMIN depends only on one parameter T = b/a, it generates the same paging curve as a large class of optimal policies all having the same b/a ratio. In the following we assume T is an integer (T = 0,1,2,...). If b/a is not an interger, VMIN will produce the same decisions as if T is the largest integer not exceeding b/a.

---

* There is a similar duality between LRU, which replaces pages with the least recent reference times, and MIN, which replaces pages with the most distant next reference times. However, since MIN replacement decisions are not independent of the present resident set or the relative future reference order of pages, a much more complicated argument is needed to prove optimality [ADU71].

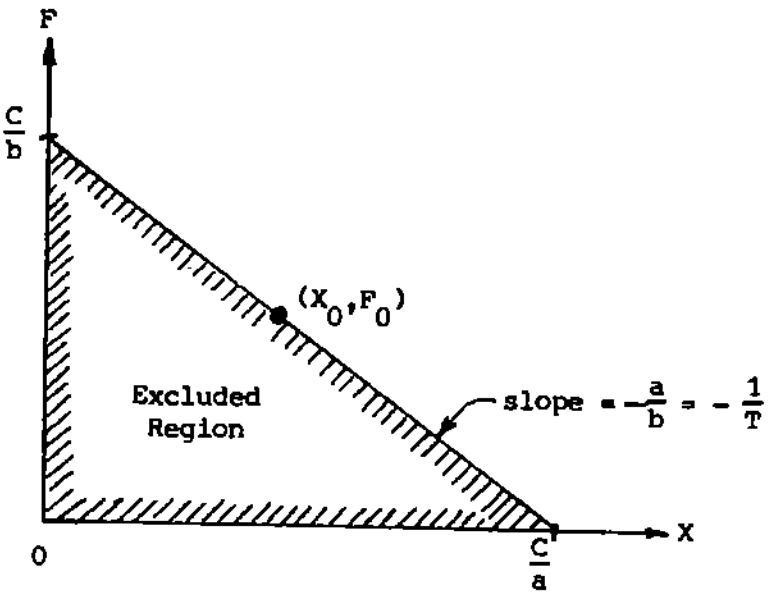

Figure 2. Excluded region of an operating point.

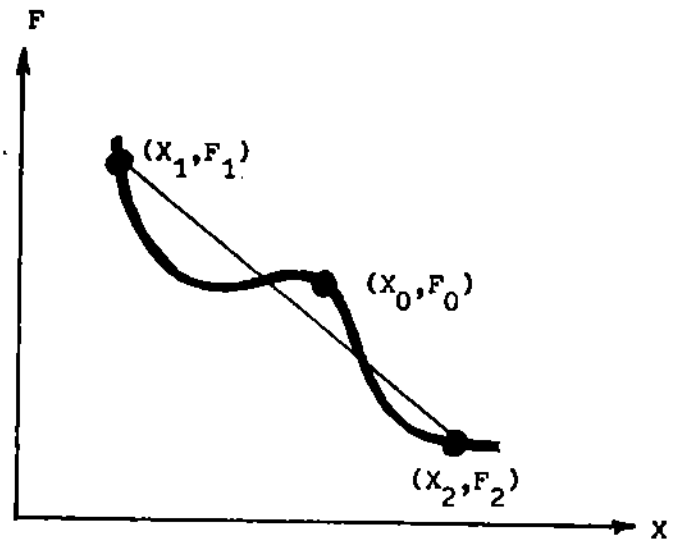

Figure 3. Illustrating why f(VMIN,X) is convex.

4. EFFICIENT COMPUTATION OF WS AND VMIN PAGING CURVES

4.1 Relation of VMIN and WS

Because of their duality, the VMIN and WS paging curves are related by a simple transformation; this permits f(VMIN,X) to be computed without simulating VMIN.

The important observation is that VMIN and WS have the same paging rate m(T). Figure 4 suggest a series of successive reference times to some give page i, with the first reference at time u_1. Suppose that u_n (n>1) is the first reference at which VMIN removes i from the resident set. It follows that

$$(4\text{-}1) \qquad u_2-u_1 \leq T, \ \ldots\ , u_n-u_{n-1} \leq T, u_{n+1}-u_n > T.$$

However, these relations also imply that page i will be a member of the working set W(t,T) for $u_1 \leq t < u_n+T$. The pattern of Figure 4 applies at each time VMIN generates a page fault, and it is obvious that VMIN and WS generate the same fault sequence.

Now let $t_1 t_2, \ldots, t_R$ denote the page fault times for policy P in the reference string r(1)r(2)...r(K). Support s_i is the removal time of the page made resident at time t_i; then $r(t_i)$ is resident during $t_i \leq t \leq s_i$. Define the presence function corresponding to the i^{th} page fault to be

$$(4\text{-}2) \qquad p_i(t) = \begin{cases} 1, & t_i \leq t \leq s_i \\ 0, & \text{otherwise} \end{cases}$$

Observe that

$$(4\text{-}3) \qquad \sum_{t=1}^{K} p_i(t) = s_i - t_i = h_i$$

is the holding time of the i^{th} faulting page. The mean holding time per page fault is

$$(4\text{-}4) \qquad H = \frac{1}{R} \sum_{i=1}^{R} h_i .$$

Observe also that the resident set size at time t is

$$(4\text{-}5) \qquad x(t) = \sum_{i=1}^{R} p_i(t) \ .$$

These relations can be combined into a simple formula X=FH as follows:

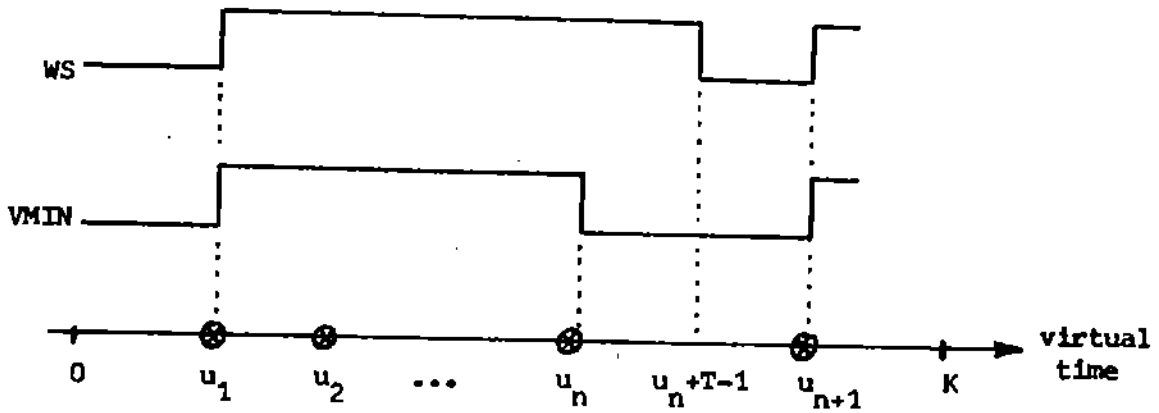

Figure 4. Presence functions for VMIN and WS.

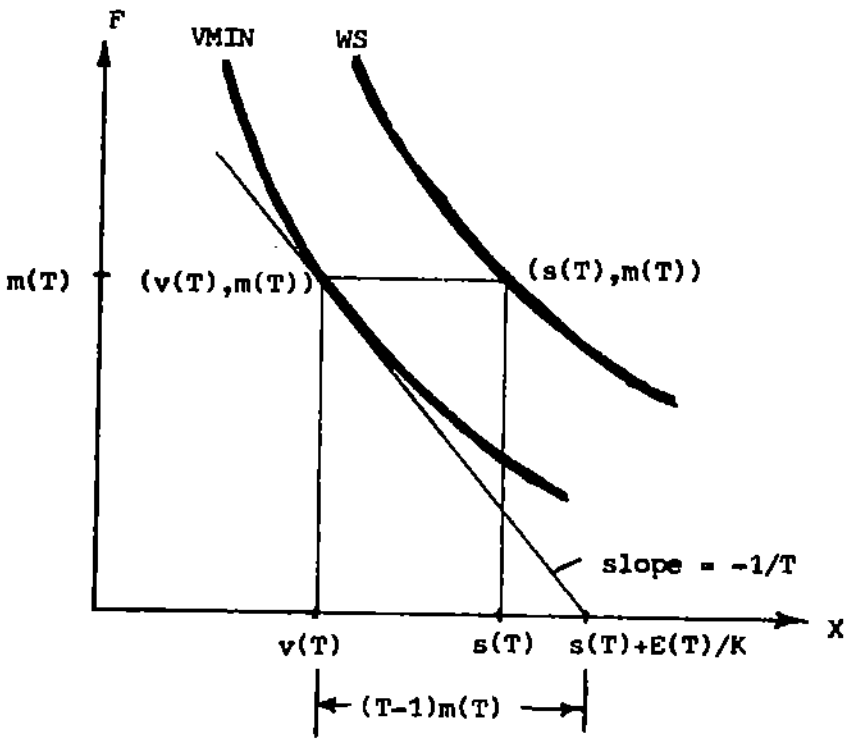

Figure 5. Geometric interpretation of results.

$$(4\text{-}6) \qquad X = \frac{1}{K}\sum_{t=1}^{K} x(t) = \frac{1}{K}\sum_{i=1}^{R}\sum_{t=1}^{K} p_i(t) = \frac{R}{K}\,\frac{1}{R}\sum_{i=1}^{R} h_i = FH,$$

where $F = R/K$ is the fault rate. (Some readers will recognize $X=FH$ as resembling Little's formula; see Maxwell [Max70].)

The diagram of Figure 4 shows that the holding times of VMIN and WS are related by $h_i^V+T-1=h_i^W$ as long as $T\geq 1$ and the last reference to the page does not fall within $T-1$ of the end of the reference string. (The superscripts V and W refer to VMIN and WS, respectively.) The relation between the means H^V and H^W can be expressed as

$$(4\text{-}7) \qquad H^V + (T-1) - \frac{E(T)}{R} = H^W, \qquad T>0,$$

where $E(T)/R$ is a correction for the end effect resulting when the last reference to a page is at time t, $K-T+1 < t \leq K$. Clearly $E(1)=0$. When T is increased to $T+1$, the total error $E(T)$ increases by one for each page whose last reference is in the interval $[K-T+1,K]$, i.e., for each member of the terminal working set $W(K,T)$. Letting $w(K,T)$ denote the size of this working set, the total error is defined for $T=1,2,3,\ldots$ by

$$(4\text{-}8) \qquad \begin{aligned} E(1) &= 0 \\ E(T+1) &= E(T) + w(K,T), \quad T>1. \end{aligned}$$

Now, let $v(T)$ denote the mean resident set size of VMIN for parameter T. From the formula $X=FH$ and $F=m(T)$ for both VMIN and WS, we have $v(T) = m(T)H^V$; then from (4-7),

$$(4\text{-}9) \qquad v(T) = m(T)\left[H^W - (T-1) + \frac{E(T)}{R}\right].$$

Employing the formula $X=FH$ to deduce that $m(T)H^W = s(T)$, and then using the definition of $m(T)$ as R/K, we can reduce this to

$$(4\text{-}10) \qquad v(T) = s(T) - (T-1)m(T) + \frac{E(T)}{K}, \quad T>0.$$

Since the holding time relation $h_i^W+(T-1)=h_i^W$ is meaningless for $T=0$, formula (4-10) does not apply for $T=0$. By arguing that holding times are both VMIN and WS are zero when $T=0$, the relation $X=FH$ implies $v(0)=0$ (a necessary choice since $f(\text{VMIN},0)\leq f(\text{WS},0)$).

As long as K is small compared to T, $E(T)$ will not be significant; in this case $v(T)$ could be computed from the simpler approximation $v(T) = s(T) - (T-1)m(T)$. Figure 5 suggests a geometric interpretation of the relation between WS and VMIN operating points, showing the line defining the VMIN excluded region for its operating point $(v(T),m(T))$.

It is interesting to note that formula (4-10) predicts $v(T+1)=v(T)$ whenever $m(T)=m(T+1)$, even though $s(T)<s(T+1)$; the proof is simple algebra using the identity $s(T+1) = s(T)+m(T)-w(K,T)/K$ [CoD73, SlT74] (see also eq. (4-12) in the next section.) If this were not the case, the convex shape of $f(VMIN,X)$ would be violated.

4.2. Computational Procedures

Slutz and Traiger [SlT74] present an algorithm for computing operating points $(s(T),m(T))$ for $T=1,2,\ldots,K$ on a reference string $r(1)\ldots r(K)$. A set of initially-zero counters $e(i)$ for $i=1,\ldots,K-1$ and $i = \infty$ are maintained, so that $e(i)$ counts the number of times t at which the backward reference interval is i. An array TIME can be used to generate the counts as follows: TIME[j] contains the most recent reference time to page j. If $r(t)=j$ then $i = t-TIME[j]$; then $e(i)$ is increased by 1 and TIME[j] set to t. (If $r(t)$ is the first reference to page j, $e(\infty)$ is increased by 1.) The missing page rate is calculated iteratively by

$$m(K) = e(\infty)/K$$
$$m(T-1) = m(T) + e(T)/K, \quad T = K,K-1,\ldots,1. \tag{4-11}$$

The mean working set size is calculated iteratively by

$$s(0) = 0$$
$$s(T+1) = s(T) + m(T) - w(K,T)/K, \quad T = 0,1,\ldots,K-1. \tag{4-12}$$

The formula for $m(T)$ follows from the observation that a page is missing if and only if its backward reference interval exceeds T. The formula for $s(T+1)$ follows by applying the formulas (2-1) and (2-2) to the observation that the working set sizes satisfy $w(t+1,T+1) = w(t,T) + \Delta(t)$. [See DeS72, CoD73, SlT74, DeG75.]

The quantity $w(K,T)$ can be determined by sorting the array TIME after the last reference, in order of decreasing reference time. Then $w(K,T)$ is the largest integer i such that $K-TIME[i] \leq T$.

There are two difficulties with the above procedure when applied to practical programs. One is the amount of storage required. Even if the number of counters

is limited to T_0, at least $2T_0$ array elements are needed to store the operating points $(s(T),m(T))$ for $T=1,2,\ldots,T_0$. Fairly large values of T_0 often have to be employed for useful answers [Bar73, Bar75, ChO72, DeK75]. A second difficulty is the computational overhead in producing the list of operating points, much of which is useless since operating points for large T values tend to be very close together [Bar73, Bar75, DeK75].

The obvious solution is to reformulate the procedure so that it computes operating points only for some specified set of parameter values $T_0,T_1,\ldots,T_{N-1}$, in which $T_0=0$ and $T_{n-1}<T_n$ for $0<n<N$. In this case we can define a set of N counters $c(1),\ldots,c(N)$ so that

$$c(n) = \begin{cases} \displaystyle\sum_{T_{n-1}<i\le T_n} e(i), & n<N \\ \displaystyle\sum_{T_{N-1}<i} e(i), & n=N. \end{cases} \tag{4-13}$$

These counters can be maintained directly. The array TIME is initialized to $-(T_{N-1}+1)$. If $r(t)=j$, then we find the largest n so that $t-\text{TIME}[j] > T_{n-1}$ and add 1 to $c(n)$. (TIME[j] is then set to t as before.) The search can be speeded up by choosing the T_n so that $g(T_n)=n$ for some easily computed function g. The $m(T_n)$ are computed iteratively by

$$\boxed{\begin{aligned} m(T_{N-1}) &= c(N)/K \\ m(T_{n-1}) &= m(T_n) + c(n)/K, \quad n=N-1,\ldots,1 \end{aligned}} \tag{4-14}$$

Since the values of $m(T)$ for $T_{n-1}<T<T_n$ are not available, formula (4-2) cannot be used directly to compute $s(T_n)$. From (4-2),

$$s(T) = \sum_{j=1}^{T-1} m(j) - \frac{1}{K}\sum_{j=0}^{T-1} w(K,j) = \sum_{j=0}^{T-1} m(j) - \frac{E(T)}{K}, \tag{4-15}$$

where $E(.)$ is the error function defined in eq. (4-8). From the tableau

$$\begin{aligned} m(0) &= \tfrac{1}{K}(e(1) + e(2) + \ldots + e(T) + \ldots) \\ + m(1) &= \tfrac{1}{K}(\quad e(2) + \ldots + e(T) + \ldots) \\ &\ldots \\ + m(T-1) &= \tfrac{1}{K}(\quad e(T) + \ldots) \end{aligned}$$

it is not difficult to deduce that

$$(4\text{-}16) \qquad \sum_{j=0}^{T-1} m(j) = \frac{1}{K} \sum_{j=1}^{T} je(j) + Tm(T) \ .$$

Comparing (4-15) with the definition of $v(T) = s(T)+E(T)/K-(T-1)m(T)$ from eq. (4-10), and using (4-16),

$$(4\text{-}17) \qquad v(T) = \frac{1}{K} \sum_{j=1}^{T} je(j) + m(T) \ .$$

Also,

$$(4\text{-}18) \qquad v(T_n) - v(T_{n-1}) = \frac{1}{K} \sum_{T_{n-1}<j\leq T_n} je(j) - (m(T_{n-1})-m(T_n)) \ .$$

Define counters $d(1),\ldots,d(N)$ so that

$$(4\text{-}19) \qquad d(n) = \begin{cases} \displaystyle\sum_{T_{n-1}<j\leq T_n} je(j), & n<N \\ \displaystyle\sum_{T_{N-1}<j} je(j), & n=N. \end{cases}$$

Using these counters and $m(T_{n-1})-m(T_n) = c(n)/K$ from (4-14), we have an iterative formula for VMIN mean resident set size:

$$(4\text{-}20) \qquad \boxed{\begin{aligned} v(T_0) &= 0 \\ v(T_n) &= v(T_{n-1}) + \frac{d(n)-c(n)}{K}, \quad n=1,\ldots,N-1. \end{aligned}}$$

As with the c-counters, the d-counters can be maintained directly. Whenever $e(i)$ increases by one, where $T_{n-1}<i\leq T_n$, $d(n)$ increases by i. Suppose $r(t) = j$ and $i = t-TIME[j]$. Find the smallest n so that $i>T_{n-1}$ and add i to $d(n)$.

The formula (4-10), when combined with (4-20), yields for the mean working set size

$$\begin{aligned} s(T_n) - s(T_{n-1}) = &\left[v(T_n) - v(T_{n-1})\right] \\ &+ \left[(T_n-1)m(T_n) - (T_{n-1}-1)m(T_{n-1})\right] \\ &- \left[\frac{E(T_n) - E(T_{n-1})}{K}\right] \end{aligned}$$

According to (4-20), the first term reduces to $(d(n)-c(n))/K$. According to (4-14), the second term reduces to $(T_n-T_{n-1})m(T_n) - (T_{n-1}-1)c(n)/K$.

With these reductions, we obtain this iterative formula for mean working set size:

(4-21)
$$\begin{aligned} s(T_0) &= 0 \\ s(T_n) &= s(T_{n-1}) + d(n)/K - T_{n-1}c(n) \\ &\quad + (T_n - T_{n-1})m(T_n) - \frac{1}{K}\left(E(T_n) - E(T_{n-1})\right) \\ &\qquad\qquad n=1,\ldots,N-1. \end{aligned}$$

Given that the TIME array is sorted by decreasing time of last reference after $r(K)$ is processed, $E(T_n)$ is not difficult to compute. Let u_i=TIME[i] after the sorting, and define u_0=0. Find the smallest k such that $T_n \leq u_{k+1}$; then

(4-22)
$$E(T_n) = \sum_{j=1}^{T_n - 1} w(K,j) = \sum_{i=1}^{k-1} i(u_i - u_{i-1}) + k(T - u_k) .$$

A simple iteration can be constructed to scan TIME and generate the values $E(T_n)$ using the principle suggested in (4-22). As noted earlier, if T_{N-1} is small compared to K, the error terms $E(T_n)$ will be negligible and can be omitted from the computations.

To summarize: The computation of $m(T_n)$, $s(T_n)$ and $v(T_n)$ can be performed for given parameter choices $T_1,\ldots,T_{N-1}$. Two sets of counters, $d(n)$ and $c(n)$ for $1 \leq n \leq N$ are needed. Whenever a backward reference interval of length i is encountered, the largest n so that $i > T_{n-1}$ is chosen; then $c(n)$ is increased by 1 and $d(n)$ by i. An array TIME, which lists the most recent reference time of each page, can be used to determine the inter-reference interval lengths. Upon being sorted after the last reference, TIME can also be used to calculate the error function. The formulas (4-14), (4-20), and (4-22) then yield the operating points of VMIN and WS policies.

5. EXAMPLES AND CONCLUSIONS

Using the methods developed above and the LRU stack processing technique, I calculated the lifetime curves of the VMIN, WS, and LRU policies for two reference strings obtained from a locality model [DeK75]. The lifetime curve, which is the reciprocal of the paging curve, defines the mean time between page faults. Both reference strings had (approximately) normally distributed locality size with a mean of 30 pages and standard deviation of 10 pages. Reference string 1 had references selected randomly from each locality, while reference string 2 had its references selected cyclically; the latter string is considered as having sharper locality than the former. The corresponding lifetime curves are displayed in Figures 6 and 7. The lifetimes are normalized with respect to the lifetime that

would be observed if paging arose only from interlocality transitions. The numbers next to the VMIN and WS curves indicate the ratio of T to the mean locality holding time.

In these and other cases, the largest operating point generable by VMIN always had X_m within a few per cent of the mean locality size. A possible use of the VMIN curves may, therefore, be estimating a given program's mean locality size. In these and other examples, the WS and VMIN curves tended closer together in direct correlation with the degree of locality in the reference string; this suggests the possibility that some measure of the difference between these curves could be used as a measure of locality.

In the introduction, I noted that a practical use of optimal paging curves is the evaluation of proposed policy improvements. To illustrate this point, I compared the different policies in a queueing network model of a simple system, assuming the programs would have their lifetime as in Figure 6 and 7. The network comprised three stations: the central processor, the disk, and the paging drum. I selected its parameters using the method detailed in [DeG75], and I computed its processor utilization using Buzen's method [Buz73]. Each running program generates disk requests at 1/10 the disk service rate. In a sufficient memory space, a program's paging rate is the same as the drum service rate. Main memory has a capacity of 300 pages. Each program was limited to a maximum of 60 pages (twice its mean locality size.) The programs in a given experiment were all of the same kind. In the VMIN and WS policies, the parameter T is used to control the load: smaller T-values imply smaller resident sets, whence the load control attains higher loads.

Figures 8 and 9 show the processor utilization U(M) as a function of the (average) load M for the three memory policies. The system throughput is directly proportional to U(M). Though they depict a hypothetical system, the following will illustrate the kind of conclusions a design engineer could reach in practice by using measured paging curves in a model of his system.

Firstly, throughput can be significantly increased by the WS policy, over the LRU policy, when programs have sharply defined locality (Figure 9); but there may be little difference when locality is less well defined (Figure 8). However, its automatic load control makes WS more stable than LRU. Because its optimal level of multiprogramming tends to be higher than that of LRU, WS also encourages more efficient utilization of memory space.

Secondly, neither WS nor LRU can achieve the same peak throughput as VMIN. However, the peak WS throughput is within 6% of the peak VMIN throughput - and so the best possible return under any policy improvement is a 6% increase in through-

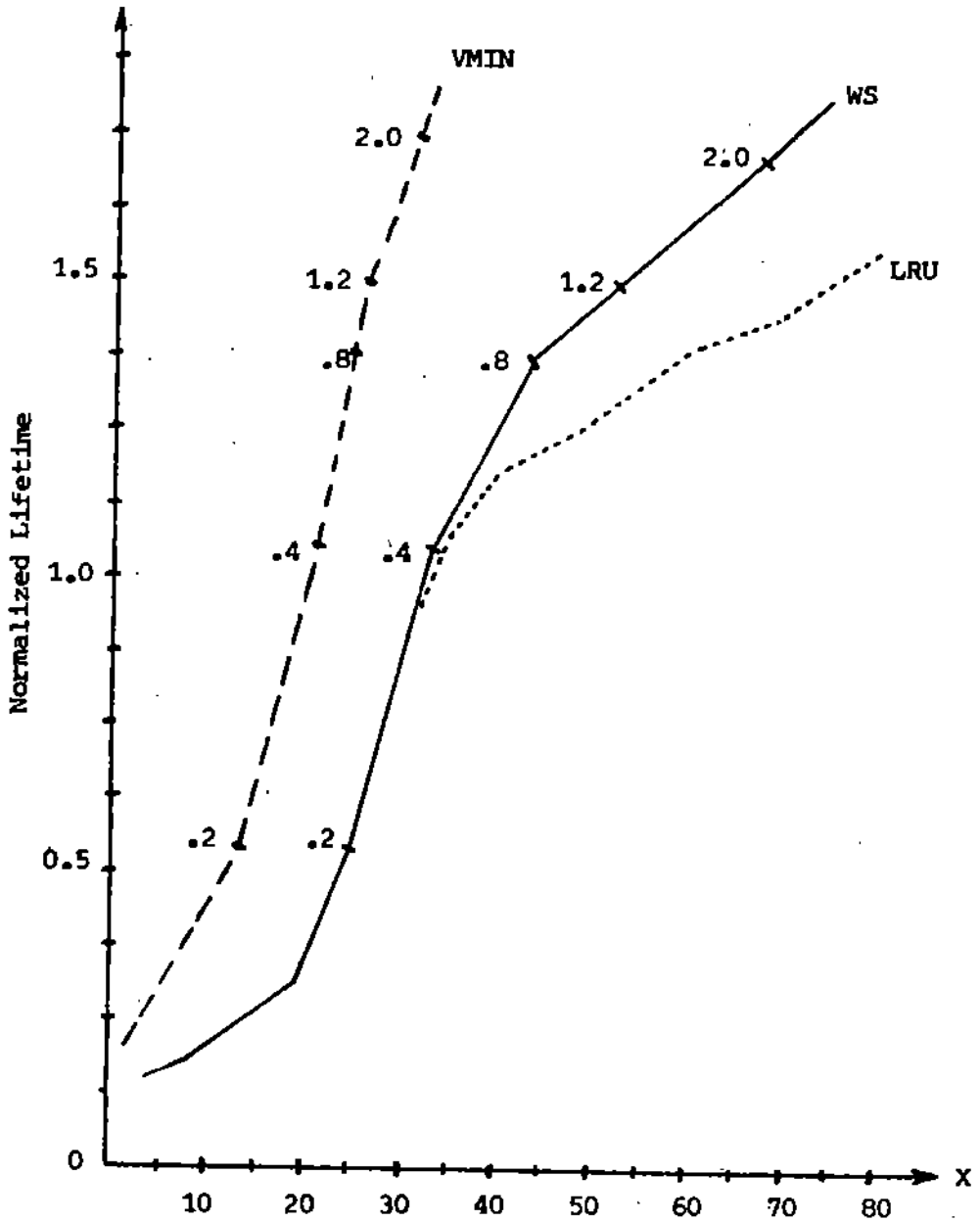

Figure 6. Lifetime curve, Reference string 1.

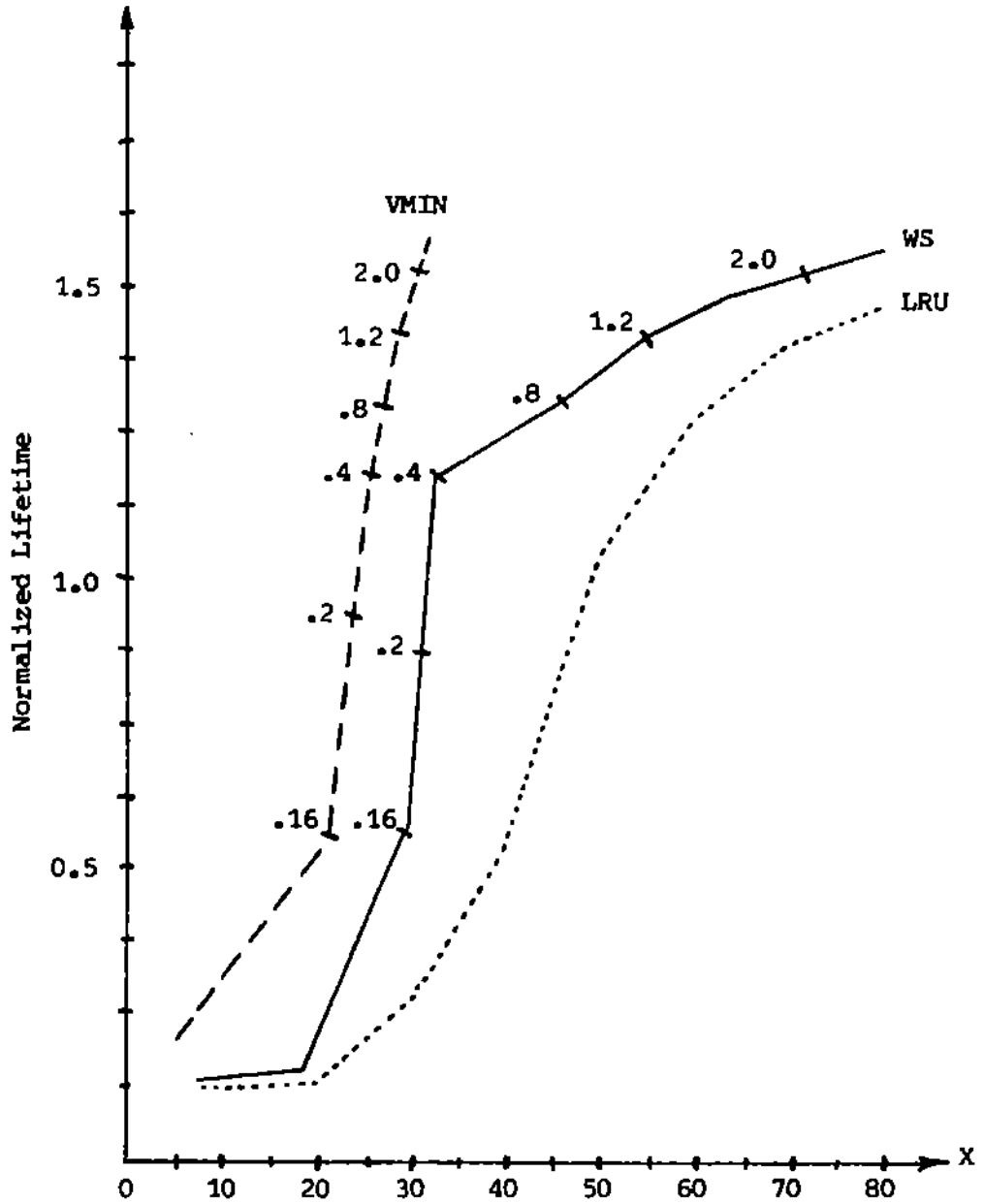

Figure 7. Lifetime curves, Reference String 2.

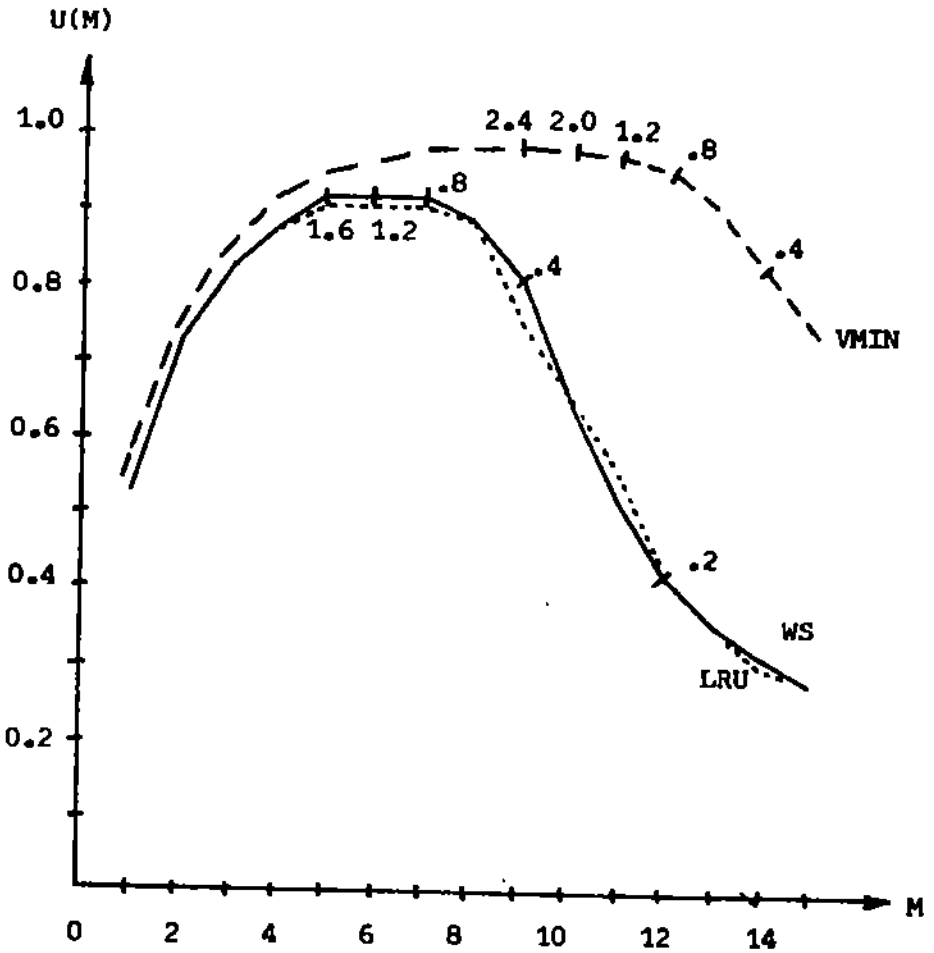

Figure 8. Processor Utilization, Reference String 1.

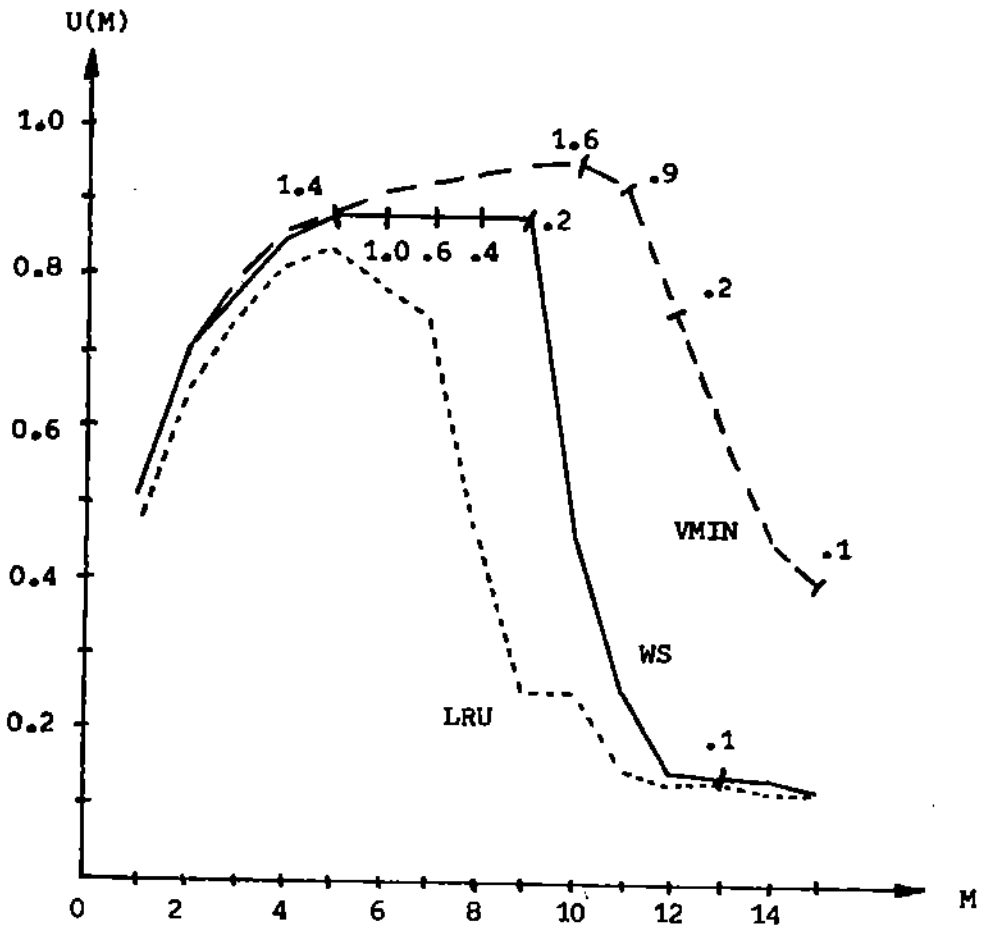

Figure 9. Processor utilization, Reference String 2.

put. A proposed improvement over the WS policy would be considered worthy of further testing only if it increased the cost per job by less than 6%.

Thirdly, the load $M=10$ corresponds to each program being permitted on average to have space available for its locality set. For highly local programs, the WS policy can be adjusted to produce its optimum level of multiprogramming close to this value. LRU cannot be so adjusted.

Fourthly, in our locality experiments [DeK75], we observed that the mean locality holding time could be estimated from the locality curves. Setting the window T to this value caused WS to operate with $M \tilde{=} 6$, which produces the peak throughput. A wide range of T-values on either side of the mean locality holding time (typically +100%, -60%) produced M values along the WS plateau, demonstrating the robustness of WS in this case. It is not too important where on the WS plateau the system operates, since its throughput is approximately the same (however, for the larger M values on the plateau, the main memory is serving as a job storage queue.) This does not necessarily argue for a smaller main memory: not only would a smaller memory interfere with the ability of the variable partition policy to freely allocate working sets; but the smaller main memory would produce a utilization curve with a narrower plateau, thereby reducing the robustness of the WS policy.

The methods of this paper can be used to find efficiently the paging curves of practical programs, both for the realizable WS policy and the unrealizable VMIN policy. When used with queueing networks, these curves can enable the design engineer to appraise the effectiveness of a WS policy with respect to the optimal, and to evaluate the best possible return to be realized in any policy improvement.

REFERENCES

[ADU71] Aho, A., Denning, P. J., and Ullman, J. D., "Principles of optimal page replacement." J. ACM 18, 1 (January 1971), 80-93.

[Bar73] Bard, Y., "Characterization of program paging in a time sharing environment." IBM J. R & D 17, 5 (September 1973), 387-393.

[Bar75] Bard, Y., "Application of the page survival index (PSI) to virtual-memory system performance." IBM J. R & D 19, 3 (May 1975), 212-220.

[Bel66] Belady, L. A., "A study of replacement algorithms for virtual storage computers." IBM Sys. J. 5, 2 (1966), 78-101.

[BeP74] Belady, L. A., and Palermo, F. P., "On-line measurement of paging behavior by the multivalued MIN algorithm." IBM J. R & D 18, 1 (January 1974), 2-19.

[Buz73] Buzen, J. P., "Computational algorithms for closed queueing networks with exponential servers." Comm, ACM 16, 9 (September 1973), 527-531.

[ChO72] Chu, W. W., and Opderbeck, H., "The page fault frequency paging algorithm." Proc. AFIPS Conf. (1972 FJCC). 597-609.

[CoD73] Coffman, E. G. Jr., and Denning, P. J., Operating Systems Theory. Prentice-Hall (1973).

[DeG75] Denning, P. J. and Graham, G. S., "Multiprogrammed memory management." Proc. IEEE on Interactive Computer Systems (June 1975).

[DeK75] Denning, P. J. and Kahn, K. C., "A study of program locality and lifetime functions," Technical Report CSD-TR-148, Computer Science Department, Purdue University (May 1975); submitted to SOSP-5.

[DeS72] Denning, P. J. and Schwartz, S. C., "Properties of the working set model." Comm. ACM 15, 3 (March 1972), 191-198.

[MaB75] Madison, W. and Batson, A., "Characteristics of program localities," Technical Report, Department of Computer Science, University of Virginia (May 1975); submitted to SOSP-5.

[Max70] Maxwell, W. L., "On the generality of the equation L=λW." Operations Research 18 (1970), 172-174.

[MST70] Mattson, R. L., Slutz, D. R., Traiger, I., and Gecsei, J., "Evaluation techniques for storage hierarchies." IBM Sys. J. 9, 2 (1970), 78-101.

[PrF75] Prieve, B. G. and Fabry, R. S., "VMIN - an optimal variable space paging algorithm," Technical Report, Bell Laboratories, Naperville, Illinois (May 1975); submitted to SOSP-5.

[SlT74] Slutz, D. R. and Traiger, I., "A note on the calculation of average working set size." Comm. ACM 17, 10 (October 1974), 563-565.

[SrK74] Sreenivasan, K. and Kleinman, A. J., "On the construction of a representative synthetic workload." Comm. ACM 17, 3(March 1974), 127-132.