1976

Generalized Working Sets for Segment Reference Strings

Peter J. Denning

Donald R. Slutz

Report Number:
76-178

TR 178

GENERALIZED WORKING SETS FOR SEGMENT REFERENCE STRINGS[1]

Peter J. Denning[2]

Donald R. Slutz[3]

March 1976
Revised January 1977
Revised September 1977
Revised December 1977

Abstract: The working set concept is extended for programs that reference segments of different sizes. The generalized working set policy (GWS) keeps as its resident set those segments whose retention costs do not exceed their retrieval costs. The GWS is a model for the entire class of demand-fetching memory policies that satisfy a resident set inclusion property. A generalized optimal policy (GOPT) is also defined; at its operating points it minimizes aggregated retention and swapping costs. Special cases of the cost structure allow GWS and GOPT to simulate any known stack algorithm, the working set, and VMIN. Efficient procedures for computing demand curves showing swapping load as a function of memory usage are developed for GWS and GOPT policies. Empirical data from an actual system are included.

Key Words and Phrases: Data base referencing, memory management, optimal memory policies, paging, program behavior, program measurement, segmentation, working sets.

CR Categories: 4.3; 8.1.

---

[1]Supported at Purdue University in part by NSF Grant GJ-41289.

[2]Computer Sciences Department, Purdue University, W. Lafayette, IN 47907.

[3]IBM Research Laboratory, 5600 Cottle Road, San Jose, CA 95193.

INTRODUCTION

Segment referencing is increasingly prevalent. It is used in many data base systems; it is supported in hardware by several manufacturers; it underlies "program restructuring," which seeks a locality-preserving distribution of small logical program blocks among large physical storage pages. There is a clear need for memory demand measures under segment referencing.

The common procedures for measuring memory demand are suited only for paged memory systems. They measure a resident set's size by counting pages, and the swapping load by counting page faults. But if storage blocks are segments of different sizes, these measures do not accurately portray a program's memory demands. The number of resident segments may bear little relation to the memory required to hold them; the number of missing-segment faults may not measure the load actually placed on the swapping system.

This paper presents the generalized working set (GWS) approach to measuring memory demand under segment referencing. It is based on defining a cost of retaining a segment in residence without being referenced, and a cost of retrieving (swapping in) a missing segment. The GWS memory policy assigns each active program a resident set containing each segment whose retention cost does not exceed its retrieval cost. A parameter Θ, the threshold, represents the cost of retrieval relative to retention. The GWS models all one-parameter memory policies whose resident sets satisfy an inclusion property under increasing values of the control parameter (Θ). The well known "stack algorithms" [MGS70, CoD73] and "time window working set" [Den68, DeS72, CoD73] are special cases of the GWS. this model extends the measurement technique to segment referencing, and it unifies previous models as well.

In a single pass over a given segment reference string, GWS procedures measure a memory demand curve $y = f(x)$. Each possible threshold value (θ) generates a particular demand point (x,y) on this curve. This curve shows the tradeoff between a "memory space investment" x and a "swapping load" y. The memory space investment x is either σ, the mean size of the resident set, or s, the mean of the retention costs of segments kept in the GWS. The swapping load represents the delay from moving segments into main memory; it is represented as $y = m + A\mu$, where m is the missing-segment fault rate, μ is the rate of information flow resulting from segment faults, and A is a parameter selected by the analyst. The familiar page fault curve is the demand curve $m = f(\sigma)$. These definitions give the analyst considerable flexibility in choosing a memory demand measure.

A special case of GWS policy is a generalized optimal policy (GOPT). No memory policy can generate a demand point below the demand curve of GOPT. Although GOPT's lookahead prevents it from operating in real time, its demand curves are easily obtained -- in fact, the GOPT and GWS demand curves can be computed from each other. It is thus cheap to learn how far from optimal a given GWS policy is. The GOPT policy reduces to VMIN for paging [PrF76].

The original procedures for measuring page fault rate curves under the time-window working set policy required storage of order O(M+p), where M is the maximum time window of interest and p is the number of pages. (See [DeS72, SlT74].) In practice, M must be very large -- 10^4 or 10^5 references -- to obtain demand points over the entire range of interest.

The GWS and GOPT measurement procedures calculate demand points for N selected threshold values, with storage of order O(N+p), where N can be as small as $\log_2 M$. For practical programs, this represents a storage reduction of two or more orders of magnitude and corresponding speedup in computing the demand curves. These procedures are generalizations of ones noted by Easton & Bennett for the time-window working set [EaB77], but they were developed independently by the authors [Den75, Slu75].

Because its cost functions measure each program singly, GWS analysis does not calculate the actual cost of running a program. It does not account for the overheads of placement or replacement policies, or the effects of queueing. GWS analysis does measure the tradeoff between memory space investment and swapping load intrinsic to each given program.

To estimate the actual cost of running a program, the memory demand curves of programs in a workload must be used to drive a simulation or analytic model of the entire system. The system model accounts for overhead and queueing. This has been done successfully many times with paging systems. For example, paging curves have been used to estimate processor utilization, throughput, and mean response time at various levels of multiprogramming [Bar73, Bar75, Cou77, DeG75]; to determine bounds on throughput [DKL76]; to construct synthetic workloads [SrK74]; and to measure program locality [DeK75, GrD77]. With demand curves from GWS analysis, these same techniques can be applied to systems with segment addressing.

DEMAND CURVES

A program's address space consists of p segments, denoted by indices 1,...,p. The size of segment i is z_i data units. A data unit is a fixed quantity such as a bit, byte, word, or page. The total of segment sizes is $Z = z_1+\ldots+z_p$. Under paging, all $z_i = 1$.

A segment reference string is a sequence $\{r(t),\ t = 1,\ldots,T\}$ in which r(t) is the index number of the segment referenced at virtual time t. The total volume of referenced information, V, is the sum, over all t, of the size of the segment referenced at time t. Since we assume that all of a segment must be loaded in main memory for referencing, the mean resident set size of any memory policy is at least V/T.

Our analysis supposes that every program starts execution with an empty resident set, and that missing segments are placed in the resident set on demand. A memory policy (MP) determines which segments are removed from resident sets. The MPs of interest here decide whether to retain or remove a segment by comparing memory usage costs against swapping loads. A fixed threshold, θ, specifies the relative cost the MP assigns to retaining and swapping segments. For a given such MP, each resident set is determined completely by the reference string and the setting of the control parameter, θ.

Our MPs associate a reference cost and a retention cost with each segment at each time t. The reference cost accounts for the unavoidable cost of using memory while a segment is being referenced. We let q_i denote the reference cost for segment i. The total reference cost, Q, is the sum, over all t, of the reference cost of the segment referenced at time t. The cost Q is incurred by every MP in processing the given reference string. For paging, all $q_i = 1$.

The (accumulated) retention cost accounts for the memory used to maintain a segment in residence beyond its prior reference. For each segment at time t, this cost is represented by a function $R(i,t) \geq 0$ satisfying three properties:

A. Prior to the first reference of segment i, R(i,t) is infinite: no finite expenditure can cause a segment to be resident before its first reference under a demand MP. (However, nonempty initial resident sets can be represented with R(i,0) being a suitable finite value.)

B. If $r(t) \neq i$, $R(i,t+1) \geq R(i,t)$: retention cost accumulates with time since prior reference.

C. If $r(t) = i$, $R(i,t^+) = 0$: retention cost is reset just after a reference. (The cost of the reference itself is accounted for by q_i.)

In general, retention cost depends on some total internal state of (a model of) the program -- thus R(i,t) is not independent of R(j,t). To keep the notation simple, we have not shown such a state explicitly as a parameter.

It is convenient to extend these definitions to continuous time, in which segments are referenced at integer times. In this case, segment i is regarded as being resident during $[t, t+1)$ whenever $r(t) = i$, and the cost of this reference is represented by q_i. (Note that R(i,t) need not be continuous.)

The demand curve of an MP for a given reference string is a function $y = f(x)$ specifying the "swapping load" y that results from making a "memory space investment" x. A point (x,y) of this function is called a demand point. The swapping load is represented as a linear form

$$y = m + A\mu,$$

where m is the miss rate, the number of segment faults per unit virtual time, and

μ is the information flow rate, the number of data units per unit virtual time being moved to satisfy segment faults.

The analyst would normally choose the parameter A so that y is proportional to the average time required to service a single swapping operation (queueing for swapping service is excluded). This can be done by setting

$$A = \frac{\text{mean time to transfer one data unit}}{\text{mean access time of secondary store}} .$$

Under this interpretation of A, the total time to complete all the swapping is yT access times, and the mean swapping delay for one fault of segment i is $1+Az_i$ access times.

There are two possible representations of the memory space investment x: either

- σ, the mean resident set size generated by the MP; or
- s, the mean memory usage cost (per reference) actually expended by the MP.

Notice that σT is the total (virtual) space-time accumulated among all resident segments; it could be computed by summing resident set sizes for $t = 1,\ldots,T$. Likewise, sT is the total memory usage cost; it could be computed by adding the total reference cost, Q, to the total of all retention-cost increments, $R(i,t) - R(i,t-1)$, for all resident i and $t = 1,\ldots,T$. However, there are more efficient computational methods than these.

The most efficient methods for measuring the totals σT, sT, and yT are based on calculating contributions for each interval between successive references to a segment [CoD73, DeS72, SlT74, PrF76]. These contributions are summarized in Table 1. There are three cases, according as $r(t) = i$ is a first, an intermediate, or a final reference. A first reference contributes a swap, and memory usage only during [t,t].

A subsequent reference ends an interval $[t'+1, t]$ that spans a pair of successive references; segment i is resident during a prefix $[t'+1,t'']$, and a swap occurs only if $t'' < t$. After a final reference there may be an additional period of residence $[t'+1, t'']$; in a one-pass measurement, its contribution must be computed after time T+1 (the procedure cannot discover prior to this time that there are no more references).

In practice, an analyst wishes to evaluate the demand curve of an MP on a given reference string only for a given set of threshold values $\{\theta_n, n = 1,...,N\}$. The measurements will yield a corresponding set of demand points (x_n,y_n). These points are usually displayed as graphs by connecting adjacent points with straight line segments. (Fitted interpolation can also be used [Smi76].) This method of display, intended primarily for visual convenience, has been used for years with page-fault rate functions -- e.g., [Bar73, Bar75, Bel66, ChO72, CoD73, EaB77, MGS70, PrF76, SlT74]. Mathematically, these graphs approximate a value $y = f(x)$ by linear interpolation between the nearest pair of measured (x,y) demand points. If the approximation is too crude, the analyst must calculate demand points for further values of θ.

It is important to remember that the reference and retention costs are abstract quantities usd to define memory policies, and that the swapping load does not account for system delays such as queueing for swapping or overhead in placement and replacement. Therefore, costs displayed by demand curves need not correspond to the actual costs of running programs in the system. To assess actual costs, an analyst must use the demand curves to drive simulations or analytic models of a system.

	References to segment i		Residence Interval	Contributions to		
	prior	present		σT	sT	yT
First	-	$r(t)$	$[t, t]$	z_i	q_i	$1+Az_i$
Subsequent	$r(t')$	$r(t)$	$[t'+1, t'']$	$z_i(t''-t')$	$q_i+R(i,t'')$	$\begin{cases} 0, & t'' = t \\ 1+Az_i, & t'' < t \end{cases}$
Post-final	$r(t')$	-	$[t'+1, t'']$	$z_i(t''-t'-1)$	$R(i,t'')$	0

TABLE 1. Contributions associated with intervals between references.

GENERALIZED MEMORY POLICIES

Generalized Working Sets

The familiar time-window working set for paging, $W(t,\tau)$, comprises all pages which have been referenced in the virtual time interval $(t-\tau, t]$ (See [Den68].) If $r(t-u)$ is the latest reference to page i prior to time t, then $u \geq 1$ and page i is in the working set whenever $u-1 \leq \tau$.

The parameter τ can be regarded as a proportionality constant that relates the value of retaining a page in memory to the cost of retrieving it on a page fault. The working set behaves as if τ page-seconds of nonreference are as expensive as one page fault; it removes a page as soon as the cost of retaining it begins to exceed the cost of retrieving it.

The generalized working set MP extends this cost balancing principle. The cost of retaining segment i in residence from its prior reference until time t is $R(i,t)$. The cost of swapping (retrieving) it is $1 + Az_i$. The threshold θ is the constant of proportionality that relates one unit of swapping to one unit of retention cost. The generalized working set (GWS), $W(t,\theta)$ for $t = 1,\ldots,T$ and $\theta \geq 0$ comprises $r(t)$ plus all segments for which

$$R(i,t) \leq \theta(1+Az_i) .$$

This definition implies that a segment can be removed from the GWS at a noninteger time; however, the program is always charged a retention cost of exactly $\theta(1+Az_i)$ for a segment so removed from the GWS. It is easy to see that the GWS satisfies the inclusion property

$$W(t,\theta) \subseteq W(t,\theta+\varepsilon) \quad \varepsilon > 0$$

for all t. This observation shows that the GWS policies are contained

in the class of all demand-fetching MPs that have a control parameter $\theta \geq 0$ and satisfy the inclusion property. The converse is also true -- every demand-fetching MP that has a control parameter $\theta \geq 0$ and satisfies the inclusion property is equivalent to some GWS.

To see this, let $M(t, \theta)$ denote the resident set of such an MP at time t, given that its control parameter is fixed at θ. Suppose that the inclusion property holds -- i.e., $M(t, \theta) \subseteq M(t, \theta+\epsilon)$. We will define a retention cost function R(i,t) so that the GWS $W(t, \theta)$ is identical to $M(t, \theta)$. Since the inclusion property holds, we can imagine that the p segments are placed on the interval $[0,\infty]$ so that, for every θ, exactly the segments of $M(t, \theta)$ are contained in the interval $[0, \theta]$. At t=0, all segments are at infinity, since $M(0, \theta)$ is empty. (This is a continuous form of the "stack" [MGS70, CoD73].) Let R(i,t) denote the distance of segment i from the origin; remember that R(i,t) may depend on some total internal state of (a model of) the program. This distance function satisfies the three properties of retention cost:

A. Prior to the first reference to segment i, R(i,t) is infinite, else i would be in $M(t, \theta)$ for some finite θ.

B. If $r(t) \neq i$, it is impossible for $R(i,t) < R(i,t-1)$: for if so, segment i would enter $M(t, R(i,t))$ at time t, contradicting the assumption that, for every θ, MP fetches missing segments only when they are referenced.

C. If $r(t) = i$, $R(i,t^+)$ must be 0, else segment i could not be guaranteed to be in $M(t, \theta)$ for every $\theta \geq 0$ just after a reference to it.

The foregoing arguments define the sense in which the GWS is a model for the entire class of demand-fetching MPs that have a single control parameter $\theta \geq 0$ and satisfy the inclusion property. No MP in this class displays "anomalous behavior", which would be a decrease in x or an increase in y when θ is decreased [FGG78].

For each value of θ, the GWS produces values of mean resident set size $\sigma(\theta)$, mean memory usage cost $s(\theta)$, miss rate $m(\theta)$, and information flow rate $\mu(\theta)$. The inclusion property implies that $s(\theta)$ and $\sigma(\theta)$ are nondecreasing in θ. Moreover, the segment faults at threshold $\theta+\varepsilon$ are a subsequence of those at threshold θ; this implies that $m(\theta)$ and $\mu(\theta)$ are nonincreasing in θ. Figure 1 summarizes these facts for a demand curve $y(\theta) = f(s(\theta))$. When $\theta = 0$, it will be true that

$$s(0) \geq Q/T \qquad \text{and} \qquad y(0) \leq 1 + AV/T,$$

with equality only if there are no repeated references, and only if retention cost is never 0 except for the infinitesimal interval immediately following a reference. When $\theta = \infty$, segment faults occur only at first references; there are p such faults and they move $Z = z_1+\ldots+z_p$ data units, whence $y(\infty) = (p+AZ)/T$.

Examples

The time-window working set removes a segment which has been unreferenced for θ time units. This effect occurs when $R(i,t) = (u-1)(1+Az_i)$, where $r(t-u)$ is the most recent reference to segment i prior to time t. For paging, this GWS resembles the original paged working set with $\tau = \theta+1$.

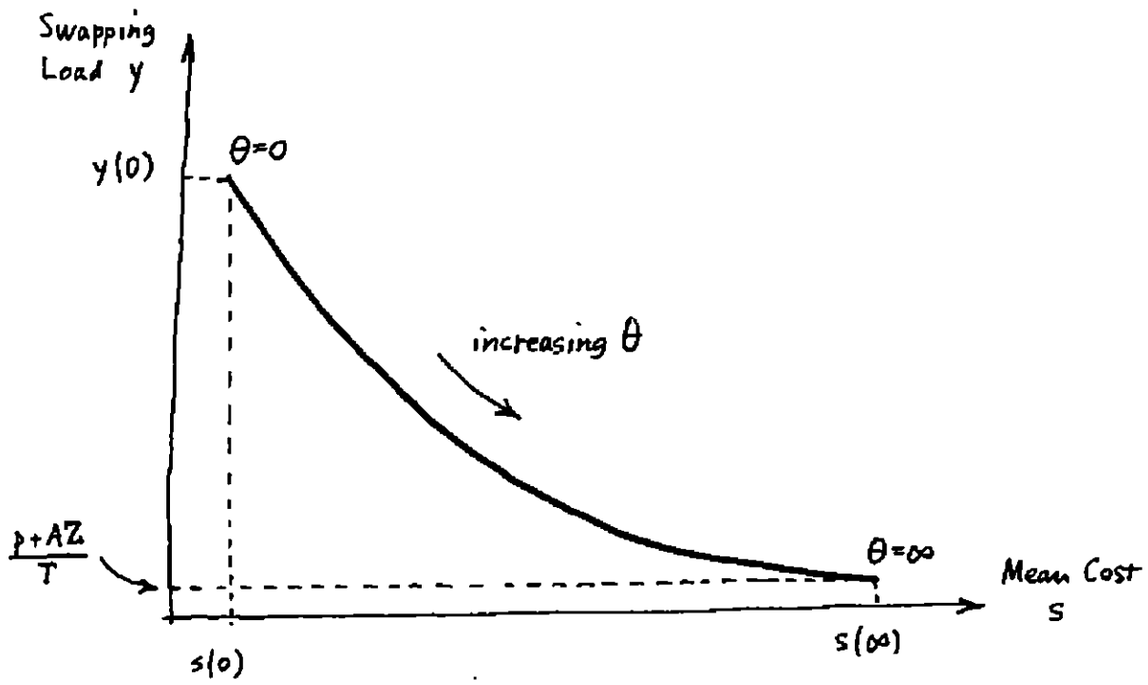

FIGURE 1. Properties of a demand curve.

The space-time working set uses $q_i = z_i$ and sets retention cost to measure the space-time accumulated by a segment after its prior reference. Thus $R(i,t) = (u-1)z_i$, where $r(t-u)$ is the most recent reference to segment i prior to time t. Putting all $z_i=1$ reduces this GWS to the time-window working set for paging. It is important to note that $s = \sigma$ -- i.e., the mean cost is the same as the mean resident set size when memory usage is measured by space-time.

The function $R(i,t) = D(i,t)-1$, where $D(i,t)$ is the stack distance of page i at time t under a given stack algorithm, is also a valid retention cost function [MGS70, CoD73]. Therefore $W(t,\theta)$ is precisely the resident set of size $\theta+1$ of the given stack algorithm. In this case $s(\theta)$ is the mean stack distance over the referenced pages.

It is possible to specify a GWS reflecting program behavior [TQT75]. Denote by $P(i,t)$ the probability of referencing page i at time t; assume that $r(t) \neq i$ implies $P(i,t+1) \leq P(i,t)$ and that $r(t) = i$ implies $P(i,t^+) = 1$. Then $R(i,t) = 1-P(i,t)$ is a valid retention cost function and, for $0 \leq \theta \leq 1$, $W(t,\theta)$ is r(t) plus all pages whose reference probability is at least $1-\theta$. A similar idea was suggested by Coffman and Ryan [CoR72].

The page fault frequency policy (PFF) [ChO72] is not a GWS. The PFF retention costs increase with time, but are reset to zero on any page fault; PFF thus violates retention cost Property B. PFF violates the inclusion property and exhibits anomalous behavior [FGG78].

Generalized Optimal Policies

The purpose of this subsection is to define a demand-fetching MP whose convex demand curve divides the (x,y) plane into a feasible upper part and an infeasible lower part. This policy will be called the generalized optimal policy (GOPT) because no memory policy can generate a demand point below its demand curve. Because the GOPT has lookahead, it is useless for optimal memory management in real time. However, its demand durve, which is easily computed as a byproduct of the GWS's demand curve, can be valuable in showing the analyst how well a program or memory policy behaves.

Recall that memory space investment (x) is either the mean size of the resident set (σ) or the mean of memory usage costs (s), and that in the space-time memory usage cost $s = \sigma$. This means that, to find the minimum possible swapping for a given mean resident set size, the analyst needs to study only the demand curve of space-time GOPT. It also means that we may study GOPT only in the (s,y) plane without loss of generality.

Underlying the GWS is the concept that the threshold θ is the value of one unit of swapping relative to one unit of memory usage. This means that $s+\theta y$ can be interpreted as the "net cost" of demand point (s,y). The concept underlying GOPT is to make replacement decisions to minimize the "net cost" relative to the given measures of memory usage and swapping.

It now follows that GOPT must remove a segment from the resident set just after its final reference, for any delay would increase memory usage (s) without affecting swapping (y). Indeed, if GOPT opts to remove any segment from the resident set, it must do so immediately

after a reference to that segment; any delay would increase s without affecting y.

It follows from these properties that, for each reference r(t) = i, GOPT makes just one of two decisions: retain i until its next reference r(t+u), or remove i just after time t. For a given value of threshold $\theta \geq 0$, the retain decision is taken if and only if

$$R(i,t+u) \leq \theta(1+Az_i) .$$

Because θ specifies the value of retaining relative to swapping, this criterion causes GOPT to select the cheaper decision for each reference. It follows that GOPT minimizes the total cost $sT + \theta yT$.[4,5] Notice that an equivalent statement of the GOPT decision rule is: take the "retain" decision for r(t)=i just if the cost/swap ratio $R(i,t+u)/(1+Az_i)$ does not exceed the threshold. If r(t) is a final reference, setting R(i,t+u) to be infinite forces the "remove" decision.

---

[4]Another way to see this is to consider the effect, on the sum $s+\theta y$, of changing a "remove" to a "retain" decision, and vice versa. Changing the reference r(t)=i from a "remove" to a "retain" changes the total memory usage cost to $sT + R(i,t+u)$, and the swapping cost to $yT - (1+Az_i)$. This changes the total net cost to $sT + \theta yT + [R(i,t+u) - \theta(1+Az_i)]$. Since r(t) is retained under GOPT, the bracketted term is positive -- such a change cannot lower the net cost. A similar argument shows that changing a GOPT "retain" to a "remove" cannot lower the net cost.

[5] GOPT need not be optimal among nondemand optimal MPs. Let F(i,w) denote the swapping cost when segment i is fetched w time units prior to a reference r(t). Note that $F(i,0) = \theta(1+Az_i)$. Prefetching would be advantageous if $R(i,t) - R(i,t-w) + F(i,w) < F(i,0)$ for some $w > 0$.

It is easy to see that GOPT satisfies the inclusion peoperty on its resident sets -- at threshold θ it takes a subset of the retain decisions it takes at threshold $\theta+\varepsilon$. Therefore, there exists a GWS that simulates GOPT. (One possible GWS uses the retention cost function R'(i,t) defined as follows. Whenever r(t) and r(t+u) are successive references to segment i, set $R'(i,t') = R(i,t+u)$ for all t' in the interval $[t+1, t+u]$.)

We can show now that GOPT's demand curve is convex and divides the (s,y) plane into a feasible upper part and an infeasible lower part. Let $\rho_0 = 0$, and let ρ_k denote the k^{th} largest of the cost/swap ratios occurring in the reference string. Let $K \leq T-p$ be the number of distinct finite values of these ratios (the p final references have infinite ratios). Then $\rho_0 \leq \rho_1 < \ldots < \rho_K$. When $\theta = \rho_k$, GOPT generates a demand point (s_k,y_k) for which s_k is the mean reference cost Q/T plus the mean of retention costs over all references whose cost/swap ratios do not exceed ρ_k, and y_k is the mean of swapping loads over all references whose ratios exceed ρ_k. As shown in Figure 2, the slope of the line connecting adjacent demand points for $\theta = \rho_{k-1}$ and $\theta = \rho_k$ is

$$\frac{\Delta y}{\Delta s} = \frac{y_k - y_{k-1}}{s_k - s_{k-1}} = -1/\rho_k .$$

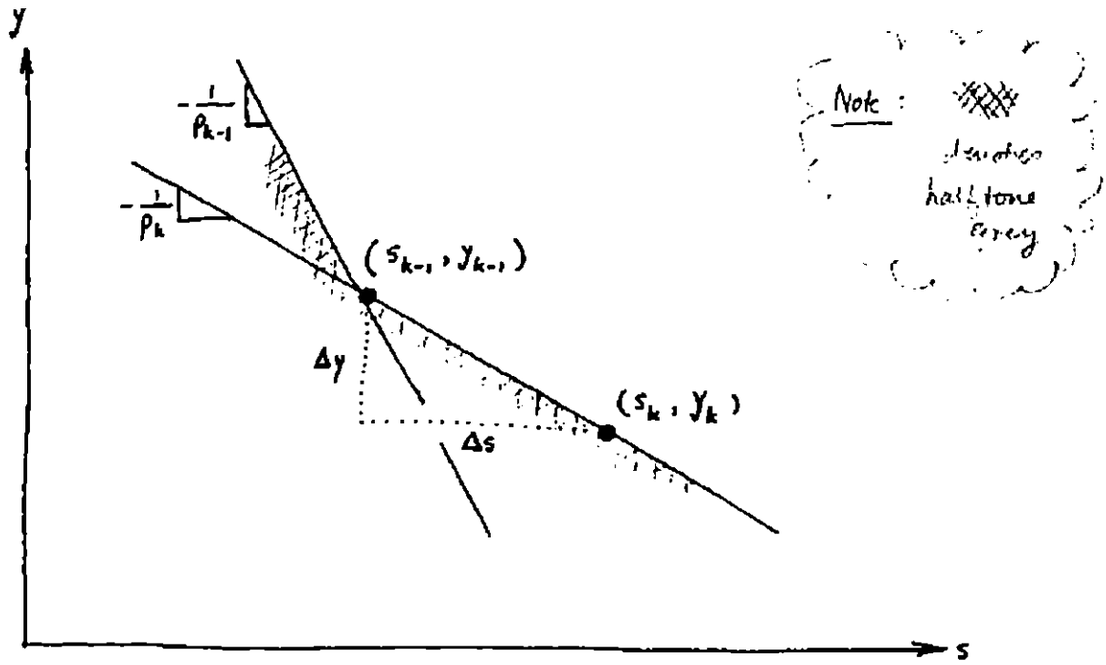

FIGURE 2. Adjacent demand points of GOPT.

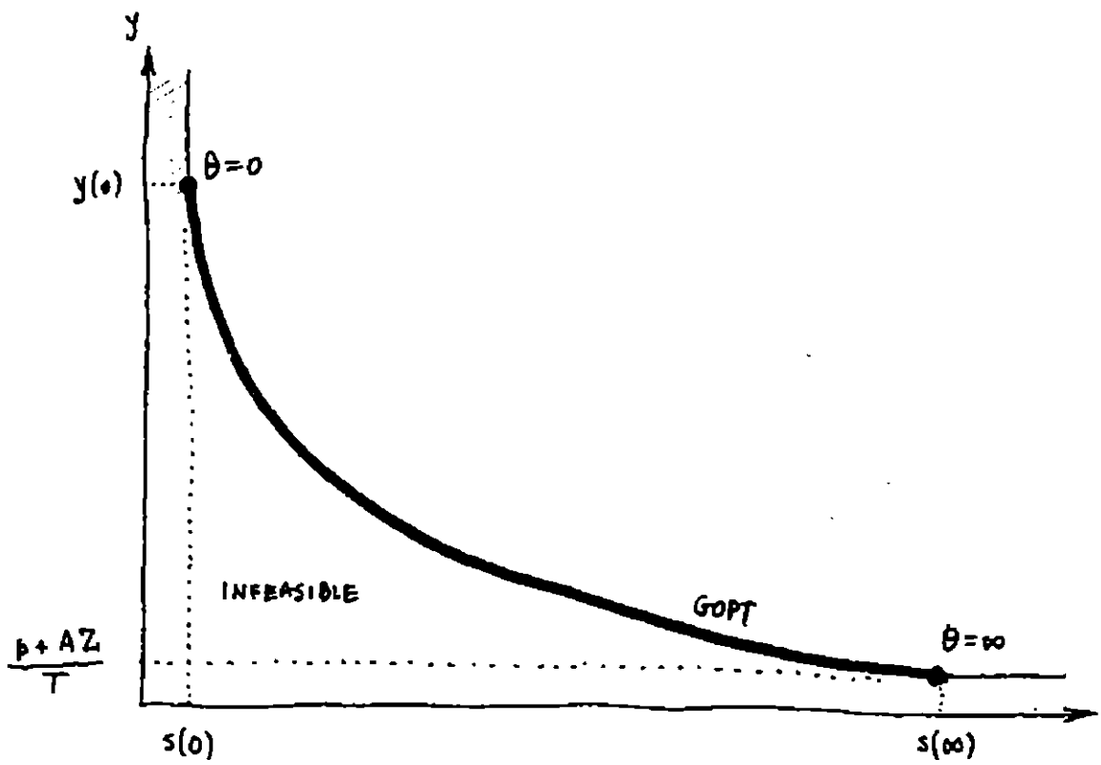

FIGURE 3. Infeasible region of demand points.

This is because all the references which GOPT changes from "remove" to "retain" decisions when θ changes from ρ_{k-1} to ρ_k have the same value of cost/swap ratio (ρ_k). Note also that for $\rho_{k-1} \leq \theta < \rho_k$, GOPT generates the one demand point (s_{k-1}, y_{k-1}). Note finally that no MP can generate a demand point below the line connecting the two demand points for $\theta = \rho_{k-1}$ and $\theta = \rho_k$; for if (s,y) were such a demand point, its "net cost" would be $s+\rho_k y < s_k+\rho_k y_k$ in contradiction to the optimality of GOPT.

Repeating the argument for k = 1,2,...,K and accounting for the boundary conditions at $\theta = 0$ and $\theta = \infty$, we find that the GOPT demand curve is convex and divides the (s,y) plane into a feasible and infeasible region, as shown in Figure 3.

Suppose that GOPT demand points are computed for a finite set of arbitrary thresholds $\theta_1, \ldots, \theta_N$. The resulting N demand points will be a subset of the K possible ones, and the piecewise linear curve connecting adjacent points will be convex. However, if $N < K$, this demand curve will not partition the plane into a feasible and feasible region.

It is possible to define other optimal MPs based on criteria such as "minimize s for each given y" or "minimize y for each given s." Because such MPs may select arbitrary subsets of references to be "retain" and "remove" decisions, they may generate as many as 2^T distinct demand points. However, the demand curve of such an MP must lie on or above the demand curve of GOPT, and need not be convex.

Examples

Let $[t, t+u]$ denote an interreference interval of segment i. When retention cost is measured by space-time, $R(i,t+u) = (u-1)z_i$; this GOPT retains i only if $u \leq 1 + \theta(A+1/z_i)$. For paging with all $z_i = 1$ and with $A = 0$, this GOPT reduces to the VMIN policy [PrF76].

We noted that GWS simulates a stack paging algorithm if the retention cost function is the stack algorithm's distance function. However, GOPT is not the MIN policy in this case [Bel66, BeP74]. MIN optimizes (σ,m) demand points over the entire class of fixed-space stack algorithms and, hence, over the entire class of possible stack distance functions; in contrast, GOPT optimizes relative to a single, given stack distance function. Moreover, VMIN may produce a demand curve below that of MIN [PrF76].

An argument similar to the one used to prove the GOPT can be used to prove that the time-window working set may be optimal among nonlook-ahead policies, when the program has sufficient locality of reference [Den78]. The required conditions seem to hold in practice [GrD77].

A Relation Between GWS and GOPT

Let $[t, t+u]$ denote an interreference interval of segment i. When $R(i,t+u) \leq \theta(1+Az_i)$ both policies retain i during the interval $[t, t+u]$. Otherwise, GOPT removes i at the beginning of the interval while GWS retains it until its retention cost attains $\theta(1+Az_i)$. Therefore, for given θ, GWS and GOPT have identical fault sequences; they produce the same swapping load.

The memory usage cost difference between GWS and GOPT is estimated easily (Figure 4). After each nonfinal reference at which GOPT removes segment i, GWS generates the additional retention cost $\theta(1+Az_i)$.

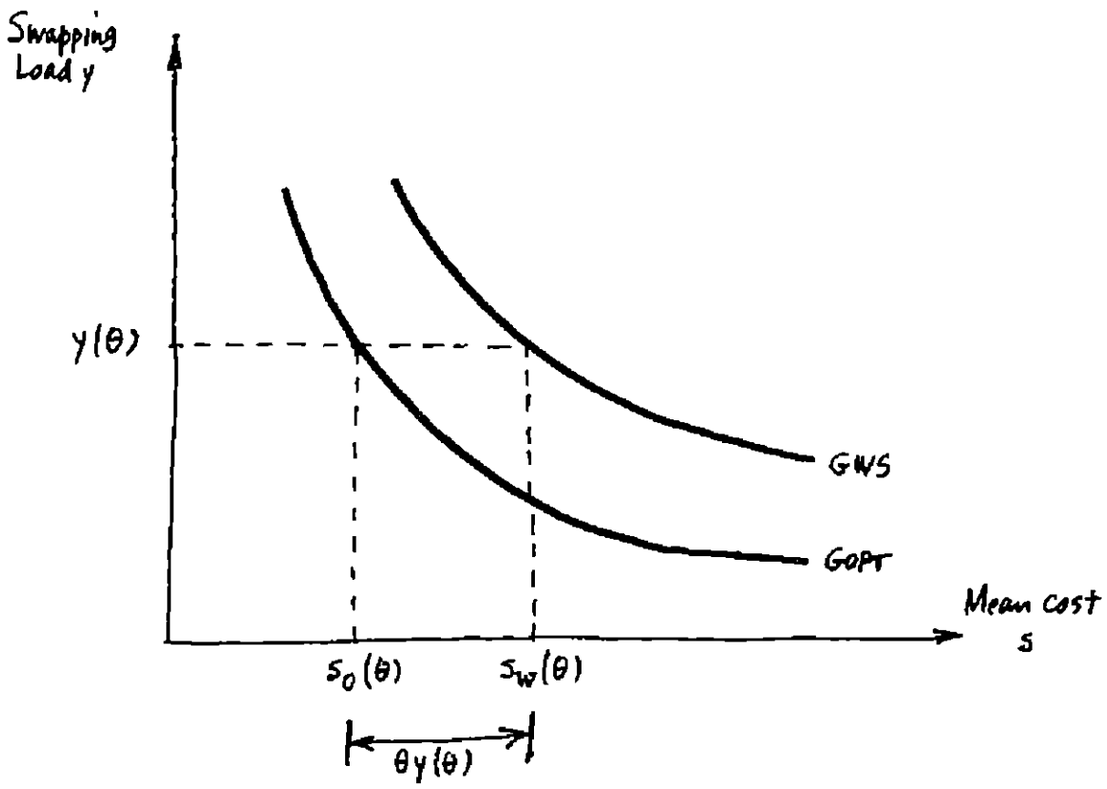

FIGURE 4. Comparison of GOPT and GWS.

After the final reference, GWS may be forced to remove i at time T+1, so that the final cost difference is at least 0 but at most $\theta(1+Az_i)$. By associating the final GWS cost contribution for segment i with the initial fault for segment i, we see that a) the total of all cost differences cannot exceed the sum of $\theta(1+Az_i)$ contributions at faults -- i.e., $T\theta y(\theta)$; and b) because the cost differences after final references cannot total more than $\theta(p+AZ)$ among the p segments of total volume Z, the total cost difference is at least $T\theta y(\theta) - \theta(p+AZ)$. Thus

$$\theta\left(y(\theta) - \frac{p+AZ}{T}\right) \leq s_W(\theta) - s_0(\theta) \leq \theta y(\theta) .$$

For small θ or large T, $\theta y(\theta)$ is a good approximation to the cost difference.

EFFICIENT COMPUTATION OF DEMAND CURVES

Let $\theta_0 = 0$ and suppose that $\theta_1,\ldots,\theta_N$ is a sequence of increasing threshold values for which GWS and GOPT demand points are to be computed. Often $\theta_{n+1} = 2\theta_n$ for $n>0$ gives clear resolution of a demand curve, whence N approximately $\log_2 T$ for a reference string of length T [Smi76]. In the following, [t-u, t] will denote an interreference interval of segment i; if r(t) is a first reference, u will be a large value. The length u can be computed simply, if each segment's time of most recent reference is kept in a table [CoD73, EaB77, SlT74].

The four measures (m, μ, s, σ) will be specified from information obtained in one pass over the reference string and stored in four sets of N+2 counters. For n = 1,...,N, the counters are defined so:

a(n), Total swapping load from segment faults that would be saved by increasing θ from θ_{n-1} to θ_n;

b(n), Total reference volume for faults tallied in a(n);

c(n), Total of retention costs that would be added by increasing θ from θ_{n-1} to θ_n; and

d(n), Total of resident segment space-times that would be added by increasing θ from θ_{n-1} to θ_n.

For $n = 0$, the counters record events for $\theta = 0$. For $n = N+1$, they record all events for $\theta > \theta_N$. Two additional counters, V and Q, tally the total reference volume and the total of (unavoidable) reference costs; each reference to segment i contributes z_i to V and q_i to Q. Initially all the counters contain zeroes.

The values in the counters are updated for $t = 1,\ldots,T$ as follows. If $R(i,t) = 0$, set n to 0, otherwise find the largest n $(1 \leq n \leq N+1)$ such that $\theta_{n-1} < R(i,t)/(1+Az_i)$. Then add

$1+Az_i$ to a(n)

z_i to b(n)

$R(i,t)$ to c(n)

$z_i(u-1)$ to d(n).

For initial references, R(i,t) is chosen to be larger than $\theta_N(1+Az_i)$ for all i. When R(i,t) depends only on u, this can be satisfied by choosing a sufficiently large initial value for u.

The counter updating actions will fail to record contributions occurring after the final reference to a segment. This does not affect GOPT, which removes every segment after its final reference; but it does affect GWS. Since the GWS behavior following the final reference to segment i depends on whether $R(i,T+1) \leq \theta(1+Az_i)$, the corrections for these "end effects" are computed by performing the counter updating actions as if $i = r(T+1)$ for each segment $i = 1,\ldots,p$. (See Table 1.) We let $a^*(n)$ denote the total corrections generated for counter $a(n)$; similarly for $b^*(n)$, $c^*(n)$, and $d^*(n)$.

Miss Rate, Flow Rate, and Swapping Load

The miss rate, information flow rate, and swapping load are the same for both GWS and GOPT. Reference r(t) = i produces a fault, whose retrieval demand is $1+Az_i$, if and only if $\Theta < R(i,t)/(1+Az_i)$; it follows that

$$y(\Theta_n) = \frac{a(n+1)+\ldots+a(N+1)}{T} .$$

The flow rate is, similarly,

$$\mu(\Theta_n) = \frac{b(n+1)+\ldots+b(N+1)}{T} .$$

The miss rate can be calculated as

$$m(\Theta_n) = y(\Theta_n) - A\mu(\Theta_n) .$$

GOPT Mean Cost and Mean Resident Set Size

The GOPT mean cost is denoted $s_0(\Theta)$ and mean resident set size $\sigma_0(\Theta)$. The total GOPT cost, $Ts_0(\Theta)$, is Q plus all retention costs generated on inter-reference intervals [t-u,t] for which $R(i,t)/(1+Az_i) \leq \Theta$:

$$s_0(\Theta_n) = \frac{Q+c(0)+\ldots+c(n)}{T} .$$

The total space-time of resident segments, $T\sigma_0(\Theta)$, is reference volume V plus all additional space-time from retained segments:

$$\sigma_0(\Theta_n) = \frac{V+d(0)+\ldots+d(n)}{T} .$$

The large retention cost assumed for an initial reference causes counters c(N+1) and d(N+1) to receive meaningless values on first references; however, these counters are not used.

GWS Mean Cost

The GWS mean cost is denoted $s_W(\Theta)$. We noted earlier that $s_W(\Theta)$ is approximated well by $s_0(\Theta) + \Theta y(\Theta)$, whenever $e = \Theta(p+AZ)/T$ is small compared to $\Theta y(\Theta)$. The exact GWS mean cost is calculated by correcting the lower bound on the cost difference, $\Theta y(\Theta)-e$, with the additional retention cost contributions following the final references. Let $s_W^*(\Theta)$ denote the correction:

$$s_W(\Theta) = s_0(\Theta) + \Theta\left(y(\Theta) - \frac{p+AZ}{T}\right) + s_W^*(\Theta) .$$

At time T+1, segment i contributes R(i,T+1) to the correction if $R(i,T+1)/(1+Az_i) \leq \Theta$; otherwise it contributes $\Theta(1+Az_i)$. Summing these contributions,

$$s_W^*(\Theta_n) = \frac{c^*(0)+\ldots c^*(n)}{T} + \Theta_n \frac{a^*(n+1)+\ldots+a^*(N+1)}{T} .$$

Since the sum of all the corrections cannot exceed $\Theta(p+AZ)$,

$$s_W^*(\Theta_n) \leq \Theta_n(p+AZ)/T .$$

There is another way of computing the exact $s_W(\Theta)$. After the counters have been used to compute the GOPT demand measures, the corrections are added to them. Then the mean cost is computed directly, using corrected a- and c-counters, from

$$s_W(\Theta_n) = \frac{\Theta+c(0)+\ldots+c(n)}{T} + \Theta_n \frac{a(n+1)+\ldots+a(N+1)-(p+AZ)}{T} .$$

The quantity p+AZ is deducted from the corrected a-counters because there are no retention costs prior to the first references.

Consider a page reference string (all $z_i = 1$) with A=0 and suppose that the thresholds are chosen to be the first N integers (i.e., $\Theta_n = n$). If retention costs are integers, such as at integer times for the time and space-time retention measures, then R(i,t) = n implies 1 is added to counter a(n) and n to counter b(n) during updating. This implies that b(n) = na(n). The mean GWS costs can be expressed as

$$s_W(n) = s_W(n-1) + m(n) + \frac{a^*(n+1)+\ldots+a^*(N+1)-p}{T} .$$

This generalizes the working set relation obtained for T infinite [CoD73, DeS72]. Since m(n) and $a^*(n+1)+\ldots+a^*(N+1)$ are nonincreasing in n, $s_W(n)$ is increasing and concave downward. A subset of the points on the $s_W(n)$ curve will define a piecewise linear curve which is also concave downward.

GWS Mean Resident Set Size

In the space-time retention measure, $s_W(\Theta)$ is the mean resident set size of GWS. But the mean resident set size $\sigma_W(\Theta)$ for an arbitrary retention measure may be more difficult to compute. This is because the additional space-time accumulated, among segments retained by GWS beyond their GOPT removal times, is not related simply to any of our previous measures. To illustrate, let v_i denote the time required beyond time t for segment i = r(t) to accumulate retention cost $\Theta(1+Az_i)$ -- i.e., $R(i,t+v_i) = \Theta(1+Az_i)$. Let $y(i,\Theta)$ denote the swapping load due to segment i. Ignoring end effects, arguments similar to our previous ones show that

$$\sigma_W(\Theta) \approx \sigma_0(\Theta) + \frac{1}{T}\sum_{i=1}^{p} v_i z_i y(i,\Theta) .$$

This could not be computed unless the a-counters were partitioned into p sets, one for each segment.

The exact $\sigma_W(\theta)$ can be computed, for the time-window working set, using the available information. For an interreference interval [t-u,t] of segment i, $u > \theta+1$ implies that segment i contributes θz_i space-time before its removal from the GWS. Reasoning similar to that used before produces

$$\sigma_W(\theta_n) = \frac{V+d(0)+\ldots+d(n)}{T} + \theta_n \frac{b(n+1)+\ldots+b(N+1)-Z}{T} ,$$

using the corrected b- and d-counters. Moreover,

$$\theta\left(\mu(\theta)-Z/T\right) \leq \sigma_W(\theta) - \sigma_0(\theta) \leq \theta\mu(\theta) .$$

EXAMPLES

Tables 2 and 3 show the distributions and calculations of the various measures for a short reference string. The demand curves m = f(s) are plotted in Figure 5 (note $\Lambda = 0$ in this case). For the time-window GWS, we calculated the resident set sizes to enable a direct comparison with costs in the space-time cost structure. Some of space-time GOPT's demand points are more favorable than for time-window GOPT, since the latter does not necessarily produce the smallest miss rate for a given mean resident set size.

Figure 6 shows the demand curves from an actual segment reference string obtained from a data base system at the IBM San Jose Research Laboratory. The data base contains several hundred thousand segments, whose sizes range from tens to hundreds of bytes with an average of about 75 bytes. / The segment reference string records only references to data segments during several hours of tracing the system; it contains nearly two million references to 183,000 distinct segments, made jointly by several concurrent users. For

(See also [RaR76, Rod76].)

this system, the time-window and space-time working sets give nearly the same performance, with neither showing a consistent advantage. At high miss rates, they require 15 to 20 times as much space as the optimal policy; this difference reduces to a factor of 4 or 5 for low miss rates.

CONCLUSION

We have extended the working set concepts to general cost measures and seg-reference strings. The memory usage costs include the unavoidable cost of all references to each segment and a nondecreasing cost of retaining each segment while unreferenced. The swapping load is proportional the delay in retrieving a missing segment. Using threshold Θ as the relative value between one unit of swapping and one unit of retention, the generalized working set (GWS) defines the resident set to be the segment referenced at time t plus all others whose retention-cost to swapping-load ratio does not yet exceed Θ. Corresponding to this is a generalized optimal policy (GOPT) which removes a segment just after it is referenced, if the retention-cost to swapping-load ratio will exceed Θ by the time of next reference.

Demand curves for the GWS and GOPT policies can be computed in a single scan of the reference string without simulation. These computations can be done with little space if we are willing to determine demand points for a small number of threshold values. For the space-time GWS and time-window GWS, demand points for adjacent value of Θ tend to be very close (or identical) when Θ is large [Bar73, Bar75, ChO72]. Thus, little resolution is lost in constructing piecewise linear curves connecting computed demand points.

When all segments are of one size and the cost structure is based on space-time, these results reduce to the familiar ones for paging: GWS becomes the conventional time-window working set [Den68], GOPT becomes VMIN [PrF76].

Preliminary data showed little practical difference between time-window and space-time GWS performance.

Most of our results do not apply if θ can vary at run time. No policy, including one with θ-variation, can generate a demand point below the GOPT curve: such variation is of no interest for optimal policies. However, it is possible to vary the GWS threshold so that GWS simulates an optimal policy for part of the time; the resulting demand point may lie below the fixed-θ GWS demand curve.

We showed that the cost difference between GWS and GOPT on demand curves $y = f(s)$ is approximately $\theta y(\theta)$. For programs whose behavior comprises long phases of referencing over associated locality sets, most of the segment faults occur during transitions between phases [Den78, DeK75, GrD77, MaB76]. For such programs, the easily-computed $\theta y(\theta)$ is a possible measure of the intrinsic differences between a lookahead policy, which can anticipate a transition, and a nonlookahead policy.

ACKNOWLEDGEMENTS

We should like to thank Alan P. Batson, R. Stockton Gaines, G. Scott Graham, Kevin C. Kahn, and Alan J. Smith for comments as the many revisions of this manuscript evolved.

TABLE 2

Example of the time-window working set

r(t):	E	C	B	E	A	B	D	C	D	E	B	C	B	B
R:	∞	∞	∞	2	∞	2	∞	5	1	5	4	3	1	0
b(R):	5	3	2	5	1	2	4	3	4	5	2	3	2	2
d(R):	-	-	-	10	-	4	-	15	4	25	8	9	2	0

End Corrections

seg	size	R	b*	d*
A	1	9	1	9
B	2	0	2	0
C	3	2	3	6
D	4	5	4	20
E	5	4	5	20

		Misses	Volume		corrected		GWS		GOPT
R=Θ	a(Θ)	Tm(Θ)	b(Θ)	Tμ(Θ)	d(Θ)	b(Θ)	$T\sigma_W(\Theta)$	d(Θ)	$T\sigma_0(\Theta)$
0	1	13	2	41	0	4	43	0	43
1	2	11	6	35	6	6	82	6	49
2	2	9	7	28	20	10	115	14	63
3	1	8	3	25	9	3	138	9	72
4	1	7	2	23	28	7	158	8	80
5	2	5	8	15	60	12	171	40	120
9	0	5	0	15	9	1	175	0	120
∞	5		15			15			

R = Retention cost value [unreferenced time]
a(Θ) = Number of references of R=Θ
b(Θ) = Total of segment sizes among references of R=Θ
c(Θ) = Total of R values among references of R=Θ
d(Θ) = Total of Rz_i among references of R=Θ
V = Total of b-counters = 43

TABLE 3

Example of the space-time working set

r(t):	E	C	B	E	A	B	D	C	D	E	B	C	B	B
R:	∞	∞	∞	10	∞	4	∞	15	4	25	8	9	2	0
	5	3	2	5	1	2	4	3	4	5	2	3	2	2

End Corrections

Seg	Size	R=c*
A	1	9
B	2	0
C	3	6
D	4	20
E	5	20

			Misses		Volume	corrected		GOPT	GWS
R=Θ	a(Θ)	c(Θ)	Tm(Θ)	b(Θ)	TμΘ)	a(Θ)	c(Θ)	$Ts_0(\Theta)$	$Ts_W(\Theta)$
0	1	0	13	2	41	2	0	43	43
2	1	2	12	2	39	1	2	45	67
4	2	8	10	6	33	2	8	53	89
6	0	0	10	0	33	1	6	53	107
8	1	8	9	2	31	1	8	61	123
9	1	9	8	3	28	2	18	70	130
10	1	10	7	5	23	1	10	80	135
15	1	15	6	3	20	1	15	95	155
20	0	0	6	0	20	2	40	95	170
25	1	25	5	5	15	1	25	120	175
∞	5			15		5			

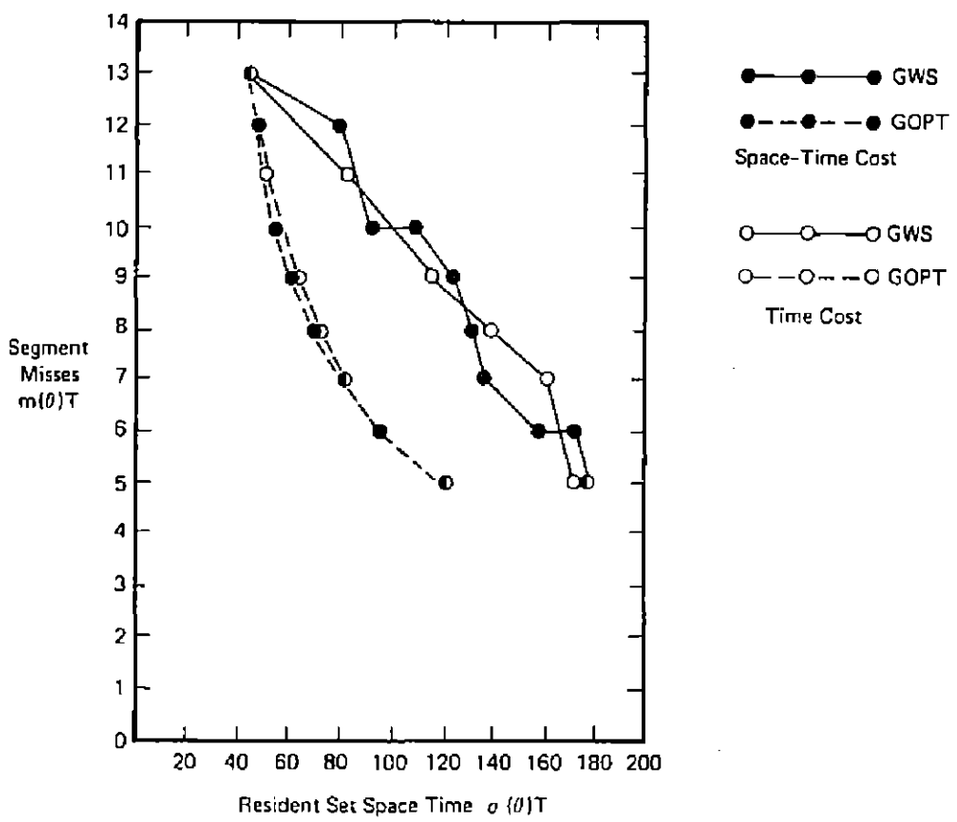

Figure 5. Demand curves of example reference string.

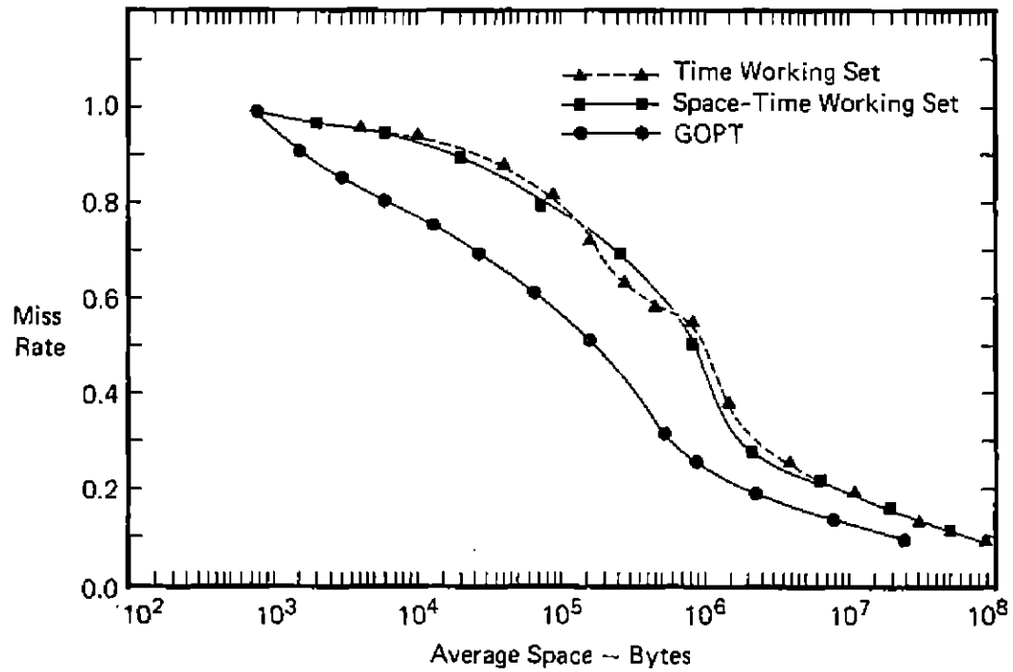

Figure 6. Demand curves for a data base system.

REFERENCES

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 multilevel MIN algorithm," IBM J R & D 18, 1 (January 1974), 2-19.

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).

CoR72 Coffman, E. G. Jr., and Ryan, T. A., "A study of storage partitioning using a mathematical model of locality," Comm. ACM 15, 3 (March 1972), 185-190.

Cou77 Courtois, P. J., Decomposability, Academic Press (1977).

Den68 Denning, P. J., "The working set model for program behavior," Comm. ACM 11, 5 (May 1968), 323-333.

Den78 Denning, P. J., "Optimal multiprogrammed memory management," in Advances in Programming Methodology III (R. Yeh & K. M. Chandy, Eds.), Prentice-Hall (1978), 300ff.

Den75 Denning, P. J., "The computation and use of optimal paging curves," Computer Science Dept., Purdue University, W. Lafayette, IN 47907, Technical Report CSD-TR-154 (June 1975).

DeG75 Denning, P. J., and Graham, G. S., "Multiprogrammed memory management," Proc. IEEE 63, 6 (June 1975), 924-939.

DKL76 Denning, P. J., Kahn, K. C., Leroudier, J., Potier, D., and Suri, R., "Optimal multiprogramming," Acta Informatica 7, 2 (1976), 197-216.

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

DeK75 Denning, P. J., and Kahn, K. C., "A study of program locality and lifetime functions," Proc. 5th ACM Symp. on Op. Syst. Princs. (November 1975), 207-216.

Ea877 Easton, M. C., and Bennett, B. T., "Transient-free working set statistics," Comm. ACM 20, 2 (February 1977), 93-99.

FGG78 Franklin, M. A., Graham, G. S., and Gupta, R. K., "Anomalies with variable partition paging algorithms," Comm. ACM 21, (1978), .

Ed.: please fill in proper citation.

GrD77 Graham, G. S., and Denning, P. J., "On the relative controllability of memory policies," Proc. Int'l Symp. on Computer Performance Modeling, Measurement, and Evaluation, North-Holland (August 1977), 411-428.

MaB76 Madison, A. W., and Batson, A. P., "Characteristics of program localities," Comm. ACM 19, 5 (May 1978), 285-294.

MGS70 Mattson, R. L., Gecsei, J., Slutz, D. R., and Traiger, I. L., "Evaluation techniques for storage hierarchies," IBM Sys. J. 9, 2 (1970), 78-101.

PrF76 Prieve, B. G., and Fabry, R. S., "VMIN - an optimal variable space paging algorithm," Comm. ACM 19, 5 (May 1976), 295-297.

RaR76 Ragaz, N., and Rodriguez-Rosell, J., "Empirical studies of storage management in a data base system," IBM Research Report RJ 1834 (May 1976).

Rod76 Rodriguez-Rosell, J., "Empirical data reference behavior in data base systems," IEEE Computer 9, 11 (November 1976), 9-13.

Slu75 Slutz, D. R., "A relation between working set and optimal algorithms for segment reference strings," IBM Research Report RJ 1623 (July 1975).

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

Smi76 Smith, A. J., "A modified working set paging algorithm," IEEE Trans. Computers C-25, 9 (September 1976), 907-914.

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

TQT75 Tran-Quoc-Te, "An open formulation of working set policies," Report 10, Project MIMOSA, Facultes Universitaires N.-D. de la Paix, Belgium (December 1975).