AN L=S CRITERION FOR OPTIMAL MULTIPROGRAMMING[1]

Peter J. Denning[2]
Kevin C. Kahn

Abstract: Balancing interpagefault lifetime (L) against page swap time (S) has always been a performance criterion of great intuitive appeal. This paper shows that, under normal conditions, controlling the memory policy parameter to enforce the constraint $L \geq S$, and allowing the multiprogramming load to rise as high as demand warrants without violating this constraint, will produce a load slightly higher than optimum. Equivalently, using the criterion $L = uS$ for some u slightly larger than 1 will approximate an optimal load. Using simulations, this criterion is compared with two others reported in the literature, namely the "knee criterion" (operate with L at the knee of the lifetime curve) and the 50% criterion (operate with the paging device at 50% utilization). The knee criterion produced optimal loads more often than the L=S criterion, which in turn produced optimal loads more often than the 50% criterion. Since no practical implementation of the knee criterion is known, the L=S criterion is the most attractive of the three.

INTRODUCTION

Designers of multiprogrammed systems using virtual memory are perenially interested in the optimal degree of multiprogramming. The optimum is characterized by maximal system service rate, and in turn by maximal processor utilization and minimal responce time [Bra74, BBG74, BGL75, Den68b, DeG75, MuW74, RoD72, RoD73, WeO69]. It is difficult, not to demonstrate theoretically that an optimum exists, but instead to find some practical adaptive optimal control. A direct control on the degree of multiprogramming in response to measured changes in the utilization of the processor (or indeed any other single device in the system) is likely to be unstable because of local fluctuations in the utilization estimator, or because utilization is sensitive to current load. (However, controls based on aggregates of device utilizations may be more stable [BGL75].)

Let L denote the system paging lifetime, i.e., the execution interval between page faults averaged over all programs, and let S denote the mean paging service time of the system. We consider a class of scheduling policies with this property: the load (degree of multiprogramming) is allowed to rise as high as the demand warrants, as long as $L \geq uS$ for some constant u. Typically the demand is sufficient to make $L = uS$ with such a policy in operation. We shall show that, as long as the system is neither execution-bound nor I/O-bound, a policy using some small constant $u>1$ will cause the load to be approximately optimal. For simplicity, we refer to these as L=S policies.

The success of an L=S policy relies on a well-behaved single-parameter memory management policy for controlling lifetime. The memory policy parameter is adjusted (dynamically if necessary) so that an estimate of L stays higher than the target S. Any working-set based memory policy, such as the moving-window working set [Den68a, Mor72] or the PFF policy [ChO72], is suitable. In either of these increasing the policy parameter increases both L and the mean resident set size, producing an indirect decrease in the load on a fixed memory.

The concept of balancing lifetime against swapping time has always presented great intuitive appeal. In 1962 Corbato et al. proposed the multilevel scheduler for CTSS, in which every job was guaranteed a quantum at least as long as the time required to swap it into main memory [CDD62]. In 1967 Belady defined a "parachor" as the space at which the mean lifetime equates the mean page swap time [Bel67; also BeK69]. In 1968 this was suggested as one method of determining the working set window [Den68a; also Pri73]. Brinch Halsen mentions it as a condition for a simple balanced system [BrH73].

ANALYSIS

Figure 1 is a diagram of the type of system under consideration: a standard network of m+1 stations, in which station 0 is the processor, station 1 is the paging device, and stations 2,...,m are the input/output devices. The source supplies transactions to the system from a finite population and may control the load n (degree of multiprogramming). Each transaction is implemented by an active program which is imagined to circulate through the network, receiving service portions from various stations sequentially, until it leaves the system.

1 Work reported herein supported in part by NSF Grant GJ- 41289.
2 Addresses: Computer Science Department, Purdue University, West Lafayette, IN 47907

Let b_i denote the service completion rate of station i (i=0,...,m), i.e., $1/b_i$ is the mean service time at station i. In particular, the page service time is

$$(1) \qquad S = 1/b_1 .$$

For the moment, we assume that the parameters $b_1,\ldots,b_m$ are independent of n (i.e., the stations contain just one server each), but b_0 may change with n because of page faults. Except that no server is idle when a task is in its queue, no other assumptions are made about service distributions or stations.

The system lifetime L(n) is the mean virtual time between paging requests when the load is n. Specifically, let L_j denote the length of the interfault interval in the program that caused the jth system page fault while the load is n; for a sequence of k such faults,

$$(2) \qquad L(n) = \frac{1}{k} \sum_{j=1}^{k} L_j .$$

To be representative, k should be at least n. It is important to note that a program need not have executed continuously for a time L_j prior to generating the jth system page fault, for it may have stopped for some I/O operation; L_j is thus the aggregated processor time of a program since its prior page fault.

We suppose that the main memory capacity is fixed at M page frames. Under this assumption it is almost always true that L(n) is strictly decreasing in n, since most memory policies will decrease the average space allocated to all old programs in order to increase n.

Let a_i denote the request rate for station i (i=1,...,m); that is, $1/a_i$ is the mean virtual time between two requests for station i. When such a request is made, the program departs the processor and joins the queue at station i. The average execution time of a transaction is taken as $1/a_0$, so that a_0 is the service completion rate. We suppose that a_0, and the I/O request rates $a_2,\ldots,a_m$, are intrinsic to the programs and independent of n. However, the paging rate a_1 is a function of n; in fact,

$$(3) \qquad L(n) = 1/a_1 .$$

Under these assumptions the processor completion rate is $b_0 = a_0+\ldots+a_m$. The frequency of transitions from station 0 to station i is

$$(4) \qquad q_i = \frac{a_i}{a_0+\ldots+a_m} = a_i/b_0 ,$$

where q_0 interprets as the frequency of departures from the system..

Under the reasonable assumption that the transition rates within the system are much faster than the rates at which transactions are submitted by the source, the long run system dynamics will depend primarily on the equilibrium values attained by the system during intervals of constant load [Cou71, Cou75].

In evaluating load controls, therefore, it is reasonable to approximate the actual behavior of the system in terms of its equilibrium for fixed multiprogramming load n.

Define $\phi_i(n)$ as the equilibrium work completion rate (tasks per unit time) of station i, for i=0,...,m. The system service rate is $\phi(n)=\phi_0(n)q_0$. In equilibrium, the output rate $\phi_i(n)$ must equate the input rate at station i, whereupon $\phi_i(n)=\phi_0(n)q_i$. Combining these facts with equation 4,

$$(5) \qquad \phi(n) = \begin{cases} \phi_i(n)a_0/a_i , & i=1,\ldots,m \\ \phi_0(n)a_0/b_0 , & i=0 \end{cases}$$

Since b_i is the maximum service rate at station i,

$$(6) \qquad \phi(n) \le \begin{cases} a_0 b_i/a_i , & i=1,\ldots,m \\ a_0 , & i=0 \end{cases}$$

Under our assumption that the I/O stations are load-independent, we can define the constant

$$(7) \qquad I = \min \left\{ 1, b_2/a_2, \ldots, b_m/a_m \right\}$$

with which (6) reduces to

$$(8) \qquad \phi(n) \le a_0 \min \left\{ I, \frac{L(n)}{S} \right\}.$$

Note that $\phi(n)$ can never exceed a_0, the intrinsic program completion rate.

BASIS OF THE L=S CRITERION

Figure 2 shows the intersection of the two bounds of (8) occuring at load n_1 (that is, $L(n_1)=IS$). The L=S criterion derives from the intuition that, when I=1, $\phi(n)$ ought to be decreasing for $n>n_1$ -- i.e., n_1 is in the vicinity of and (slightly) larger than the optimum n_0. This intuition is weak, and requires refinement according to two gross properties of the system, viz., its execution and I/O bounds.[1]

In an execution-bound system, programs either of inherently short execution time or subject to a maximum limit on main memory allocation, give rise to a lifetime function which does not exceed some value $L_{max} < IS$. In this case, n_1 cannot be defined as a crossing point. However, it can be defined as the largest value of n at which $L(n) = L_{max}$. As suggested in Figure 3, the intuition is now that this value of n will be a good approximation to n_0. (Do not confuse the terms "execution-bound" and "CPU-bound"; the latter refers to the relative amounts of computing and I/O in a program.)

---

1 For a simple system with m=1, and using $L(n) = c(M/n)^k$ for memory size M, Brandwajn et al. [BGL74] show that $n_0=(c/S)^{1/k}(M/e)$ where e is the base of the natural logarithm. In this case $L(n_0)=e^kS$ and $n_1=en_0$. The policy criterion for this case is $L=e^kS$, and n_1 is not close to n_0. However, various practical factors will operate to force n_1 closer to n_0 than this model predicts, such as flattening of L(n) for $n<n_1$ or the presence of I/O devices in the system [LeP75].

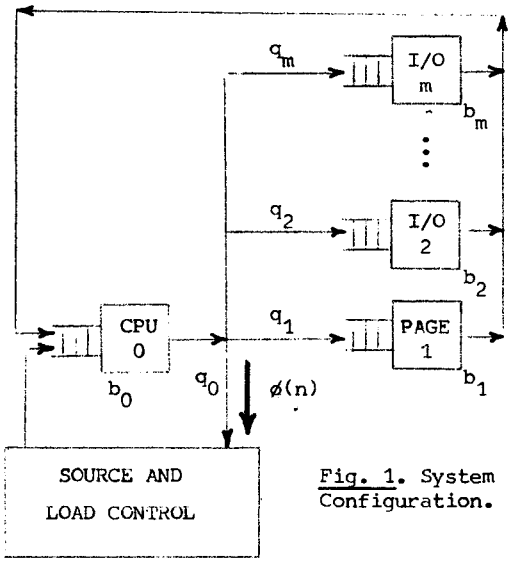

Fig. 1. System Configuration.

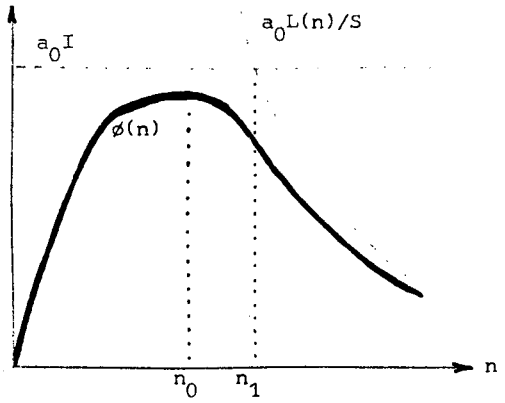

Fig. 2. Bounds on System Service Rate.

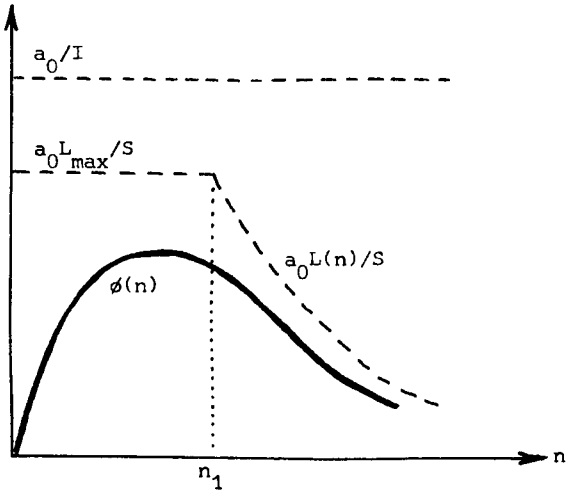

Fig. 3. Execution Bound System

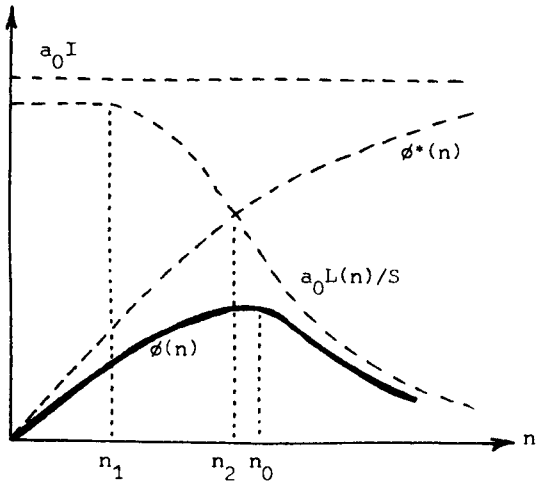

Fig. 4. Execution and I/O Bound System.

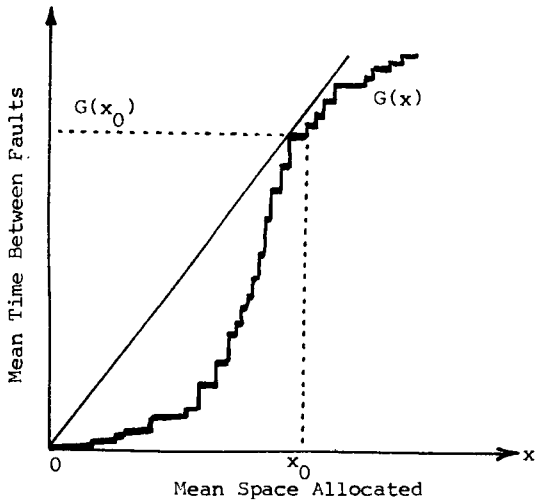

Fig. 5. Lifetime Curve Knee.

In an I/O-bound system, the service completion rate of some I/O station is smaller than the request rate ($b_i < a_i$), which implies from (7) that $I < 1$. Moreover, the rate of growth of $\phi(n)$ toward the asymptote $a_0 I$ will be influenced by the ratios b_i/a_i: with all other system parameters held fixed, a reduction in any one ratio will cause $\phi(n)$ to grow more slowly in n.

If a system is both execution bound and I/O bound, the L=S criterion may be poor. In this case, it may happen that the crossing point n_1 is smaller than the optimum, and the policy criterion L=uS would require $u < 1$. Figure 4 shows a new bound $\phi^*(n)$, which is the system service rate that would be observed under the (unrealizable) hypothesis that S=0:

$$(9) \qquad \phi(n) \leq \min\left\{\phi^*(n), \frac{a_0 L(n)}{S}\right\}.$$

Let n_2 denote the crossing point of these two bounds. The combination of the execution bound and the slow growth of $\phi^*(n)$ caused by the I/O bound can create a circumstance under which $n_0 > n_1$. Since the slow growth of $\phi^*(n)$ and early decay of L(n) in this case makes the "plateau" of $\phi(n)$ narrow, a fixed constant $u<1$ is not likely to produce a robust policy L=uS for this case.

Two independent conditions sufficient to cause $n_0 \leq n_1$ and a wide plateau in $\phi(n)$ are easily deduced. The first is that the bound $a_0 L(n)/S$ crosses $a_0 I$ with steep descent, a condition frequently encountered in practice.[2] In this case L=uS is a suitable policy function, and u between 1 and 2 may approximate peak service rate (see below). The second is that $L_{max} > IS$ and $\phi^*(n_2)$ is close to $a_0 I$. This condition is more easily achieved when there is sufficient I/O capacity, which increases the growth rate of $\phi^*(n)$.

The foregoing may be summarized as follows. If the system is neither execution bound nor I/O bound, the criterion L=S would tend to allow a load n_1 (slightly) larger than the optimum n_0 - i.e., L=uS for some value of parameter $u > 1$ will approximate maximal service rate. If the system is I/O bound, the criterion is L = uIS. If the system is execution bound, the criterion degenerates to $L = L_{max}$. If the system is both execution bound and I/O bound, a criterion L = uIS for $u < 1$ may be required, but is not likely to be robust. However, the last case is not of great practical interest, since the two bounds will severely limit the system service rate -- the system designer has problems more important than load optimization to solve. In the cases of most interest, a simple control on the lifetime will achieve nearly the maximal throughput.

2 As an illustration, let G(x) denote a program's lifetime as a function of its mean resident set size x. Take L(n)=G(M/n) for main memory size M. Assuming that n_1 within 1 of n_0 is tolerable, the slope of the bound $a_0 L(n)/S$ should be at most -1 at $n=n_1$. Taking derivatives, this translates to the condition $G'(M/n) \leq n_1^2 S/a_0 M$. For typical values (e.g., $S=10^4$ microseconds, $a_0 = 10^{-6}$/microsecond, M=100 pages) this condition is easily satisfied by empirical lifetime functions.

EXPERIMENTS

To explore the quality of three criteria of optimal control, we conducted a series of simulations of queueing networks. First is the L=S criterion, whose strong intuitive basis has been outlined above. Second is the knee criterion suggested by Denning and Graham [DeG75]; it asserts that operating a working set policy at the knee of its lifetime curve will define a load at which system service rate is close to its peak value. The strong intuitive basis of this criterion is described in the next paragraph. Third is the 50% criterion suggested by Leroudier and Potier [LeP75]; it asserts strong correlation between optimal load and 50% utilization of the paging device. Beyond the statement that one ought not overload the paging device, this criterion has no obvious intuitive basis; it resulted as an empirical observation from a series of simulations [LeP75].

The intuition behind the knee criterion is as follows. A program's lifetime curve G(x) is an increasing function specifying, for given mean resident set size x generated by a given memory policy, the mean virtual time between successive page faults. The "knee" is a point x_0 beyond which the graph G(x) flattens; it is defined as the tangency point of a ray eminating from G(0)=1 [DeK75]. See Figure 5. A straightforward geometric argument shows that purturbations from the equipartion of an M-page memory, in which each of $n_0 = M/x_0$ programs is allocated mean space x_0, tend to reduce the mean system lifetime from $L(n_0)$ [DeG75]. Our locality experiments show further that the knee lifetime $G(x_0)$ is approximated by the ratio h/m, in which h is the mean holding time between phase transitions and m is the mean number of pages entering locality at a transition in the program [DeK75]. In other words, the knee lifetime is governed principally by program behavior at phase transitions.

Figures 6-20 represent a series of one hundred simulations of a queueing network with one I/O station (a file disk); Buzen's method was applied in each to evaluate station utilizations [Buz73]. For a given program lifetime function G(x), a memory limit of Y pages per program (i.e., $x \leq Y$), and a main memory capacity of M pages, we used for the system lifetime function

$$(10) \qquad L(n) = \begin{cases} G(M/n), & n \geq M/Y \\ L_{max} = G(Y), & n < M/Y \end{cases}$$

Figures 7-13 show various configurations corresponding to the lifetime function to Figure 6, which has an idealized, smooth convex/concave shape; it was generated from a locality model [DeK75]. Figures 15-20 show configurations corresponding to the lifetime function of Figure 14, which was obtained from a real program. Each figure is labelled with the remaining parameters:

M, main memory size (pages);
Y, space limit per program (pages);
L_{max}, maximum lifetime, G(Y);
S, page device service time; and
R = b_2/a_2, the ratio of disk service to request rates.

Each figure shows four curves: the normalized service rate $\phi(n)/a_0$ which is, in this case, the processor utilization; the utilization U_1 of the paging device; the I-bound of relation (8); and the lifetime bound L(n)/S of relation (8). The curve $\phi(n)/a_0$ is marked to show the operating point of the knee (Δ) and 50% (O) rules. It is also marked with the point at which L=min (L_{max},S), which may not be an operating point. The caption "5% u range" tells the range of u-values to be used in the control relation L=uS to produce an operating point n at which $\phi(n)$ is within 5% of the optimal $\phi(n_0)$.

Figures 7-10, and again 15-17, demonstrate the effect of progressively increasing S while holding other parameters fixed in a system having no significant execution or I/O bound. Figures 11-13, and again 16-20, demonstrate the effects of increasingly stringent I/O and execution bounds, with Figures 13 and 20 representing the extreme cases -- in which all but the knee criterion have failed.

The L=S criterion behaved according to expectations across the range of experiments. When the system was not execution or I/O bound, operating with L=1.3S consistently defined a load whose service rate was within 5% of peak. (See Figures 7, 8, 9, 11, 15, 16 and 18.) In systems which were only execution bound (i.e., $L_{max} < S$ and $R > 1$) operating all programs at their maximum allowable space (Y pages) consistently defined a load whose service rate within 5% of peak (See Figures 10 and 17). In the system both execution and I/O bound, the criterion L=uS required a value u<1 to be optimal (see Figures 10, 12, 13, 15, 19, and 20).

The experiments showed the knee criterion to be almost always within 5% of peak. Only when S exceeded the knee lifetime by a significant amount (e.g., 80% in Figure 10) did this criterion worsen; we observed no case in which it deviated by more than 10% from peak. As might be expected, the exact value of load generated by this criterion does depend on S: the load tended larger than optimum with S beyond the knee, and smaller with S below the knee. When S was at the knee, this agreed with the L=S criterion. Interestingly, the knee criterion was the most reliable when the system was both execution and I/O bound.

The 50% criterion seemed very accurate as long as S was less than the knee lifetime; for relatively smooth lifetime curves, this corresponds to operating in the convex region. (See Figures 7, 8, 9, 15, 16 and 18.) However, as soon as S was comparable to or larger than the knee lifetime, this criterion tended to grossly underestimate optimal load.[3] The potential utility of this criterion apparently depends on knowing that the swap time S is always less than the mean knee lifetime of the system.

3 In studying this property, Leroudier and Potier [LeP75] assumed that $G(x) = cx^k$ which has no knee and is everywhere convex. Thus our findings do not contradict theirs.

EXTENSIONS

The L=S criterion applies to a large variety of systems of the configuration shown in Figure 1: the argument assumes only that the rates a_i and b_i exist, and that all tasks submitted to a given service station are eventually completed. Several generalizations can be made.

One generalization permits load-dependent service rates for the nonprocessor stations (1,...,m). Make the reasonable assumption that the service rates $b_i(n)$ are nondecreasing in n. Assume as before that the paging request rate $a_1(n)$ is increasing in n while the I/O request rates $a_2,\ldots,a_m$ are independent of n. Define

$$(11) \quad I(n) = \min\left\{1, b_2(n)/a_2,\ldots,b_m(n)/a_m\right\}$$

and note that I(n) is nondecreasing in n. The service rate bound is now

$$(12) \quad \phi(n) \leq a_0 \min\left\{I(n), \frac{L(n)}{S(n)}\right\}.$$

Under the reasonable assumption that lifetime decreases more rapidly than paging service time, there will exist a unique crossing point load n_1 at which

$$(13) \quad L(n_1) = I(n_1)\, S(n_1),$$

with the same interpretation as before. The load control in this case is more difficult to implement, since the load dependent page service rate must be used.

Another generalization assumes that the service rates b_i are constant but all the request rates a_i are load dependent. In this case we take $S_i = 1/b_i$ and $L_i(n) = 1/a_i(n)$. The service rate bound is

$$(14) \quad \phi(n) \leq a_0 \min\left\{1, L_1(n)/S_1,\ldots,L_m(n)/S_m\right\}$$

Assume the system is not I/O bound, i.e., $L_i(n) > S_i$ for small n and all i. In this case $\phi(n)$ will attain its peak value at $n_0 \leq n_1$, where n_1 is the least n for which $L_i(n) \geq S_i$ for all i. In other words, the control must apply the criterion to the device i for which L=S at the smallest load; this need not be the paging device.

Both generalizations can be combined to yield as L=S criterion for cases where both service rates and request rates are load dependent.

Some systems of the form of Figure 1 employ time slicing and swapping: that is, a complete program is swapped into main memory at the start of a quantum and out again at the end, without paging in the interim. In this case the L=S criterion is interpreted to mean that the quantum should be slightly longer than the swapping time. This may not be feasible: for short transactions of such systems are execution bound. However, the L=S criterion then degenerates to the usual policy, of granting each transaction program whatever space it requires, and fitting as many such programs as possible into memory.

CONCLUSIONS

Figure 21 is a summary of the results of this paper, showing the control rule that will maximize system service rate and approximate an optimal degree of multiprogramming for various combinations of execution and I/O bounds. Except when the system is both execution and I/O bound, an L=S rule applies; and when no such rule applies, the knee criterion can be used. The shaded area suggests the possibility that an L=S rule may be inapplicable in a system on the verge of being both execution and I/O bound.

Our simulations showed that we can be much more optimistic about generating a load n at which the service rate $\phi(n)$ is within a few per cent of the optimal $\phi(n_0)$, than we can be about causing n itself to be within a few per cent of n_0. If the source population contains N users, of which each spends average time T thinking between submitting requests to the system, the mean response time will be

$$W = \frac{N}{\phi(n)} - T$$

(See [Mun75, Cou75]). In other words, any load on the "plateau" of $\phi(n)$ will approximate optimum throughput and response time. Now: if you account for other factors in the system, such as overhead of paging, you may decide that two values of load on the plateau are not equivalent. Specifically, $n_1 < n_2$ with $\phi(n_1)=\phi(n_2)$ will produce less paging at n_1; unless page fault overhead is negligible compared to S, the smaller load would be the more desirable. (If paging overhead o(n) is significant, we can use the load dependent control of relation (13) in which S(n) = S + o(n).)

Our simulations suggest that the knee criterion is the most robust, producing near-optimal loads over the widest range of conditions. While giving astonishingly good results, the tangency definition of knee (Figure 5) does not appear useful as the basis of an implementation. By exploiting strong correlation between knee lifetime and program behavior at single phase transitions, a good locality estimator would constitute a basis for implementing the knee criterion [DeK75]. Since we know of no reliable method of controlling the working set window size so that each program operates near its knee lifetime, we are attracted to the more easily realized L=S rules.

The application of these results depends on the stability of the control rule. For example, the system lifetime L(n) would be estimated as an average of the program lifetimes preceding the last k system page faults (for $k \geq n$; see eq. (2)). If the character of the active set of jobs has changed significantly over the estimation interval, the estimate may be inaccurate at the time it is used to adjust the active load. In other words, there is a possible problem of excessive delay between obtaining and using an estimate. Since multiprogramming systems employing working set memory management have been successfully implemented, we can be optimistic about stable implementations of the L=S criterion. Our present formal methods of dealing with stability in computer systems are primitive [Cou75].

An interesting property of systems controlled by a policy L=uS is that system lifetime will be approximately uS for all main memory sizes M larger than that required to permit one program to achieve lifetime of uS. (Likewise, in the case considered in footnote (1), the lifetime at the optimum is a constant independent of memory size.) This does not contradict Saltzer's observation of linear growth in L as a function of M [Sal74]. In Saltzer's experiment, some of the (execution bound) programs sharing memory were inactive, belonging to terminals in "think state"; references to the paging device are thus decreased for larger M because larger numbers of programs making transitions into "active state" require no paging. In our case, all the programs sharing memory are active, and increasing M permits more such programs to be resident.

ACKNOWLEDGEMENTS

We should like to thank J. Buzen, D. Ferrari, J. Leroudier, and D. Potier for many helpful comments and criticisms on an earlier draft of this paper.

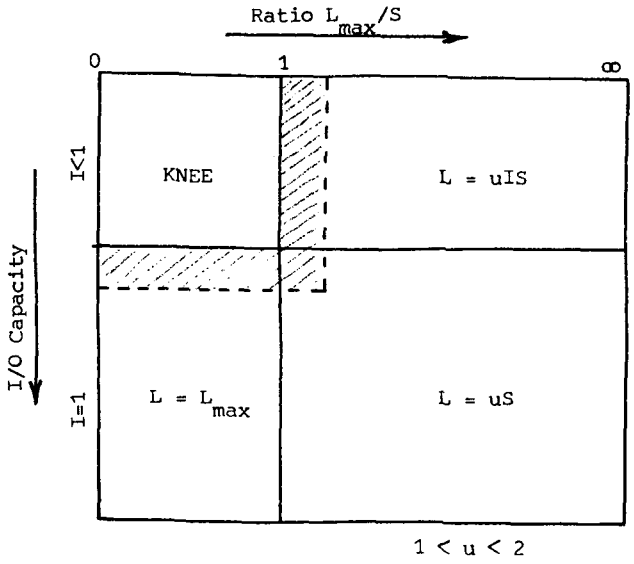

Fig. 21. Control Rule Summary.

BIBLIOGRAPHY

[BBG74] Brandwajn, A., Buzen, J., Gelenbe, E., and Potier, D. "A model of performance for virtual memory systems." Proc. ACM SIGMETRICS Symp. (October 1974), 9.

[Bel67] Belady, L. A. "Biased replacement algorithms for multiprogramming." IBM T. J. Watson Research Center Note NC697 (March 1967).

[BeK69] Belady, L. A., and Kuehner, C. J., "Dynamic Space Sharing in computer systems," Comm. ACM 12, 5 (May 1969), 282-288.

[BGL73] Brandwajn, A., Gelenbe, E., Lenfant, J., and Potier, D., "A model of program and system behavior in virtual memory.". Technical Report, IRIA-Laboria, Rocquencourt, 78150 Le Chesnay, France (October 1973).

[BGL75] Badel, M., Gelenbe, E., Leroudier, J., and Potier, D., "Adaptive optimization of a time sharing system's performance," Proc. IEEE: Special Issue on Interactive Computer Systems (June 1975), 958-965.

[Bra74] Brandwajn, A., "A model of a time sharing virtual memory system solved using equivalence and decomposition methods," Acta Informatica 4 (1974), 11-47.

[BrH73] Brinch Hansen, P., Operating Systems Principles, Prentice-Hall (1973).

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

[CDD62] Corbato, F. J., Bagget, M. M., and Daley, R. C., "An experimental time sharing system," AFIPS Conf. Proc. 21 (1962 SJCC), 279-294. [In Programming Systems and Languages, S. Rosen, Ed., McGraw (1967).]

[ChO72] Chu, W. W., and Opderbeck, H., "The page fault frequency replacement algorithm," AFIPS Conf. Proc. 41 (1972 FJCC), 597-609.

[Cou71] Courtois, P. J., "On the near-complete-decomposability of networks of queues and of stochastic models of multiprogrammed computer systems," Sci. Rpt. CMU-CS-72-11, Carnegie-Mellon University, (Nov 1971).

[Cou75] Courtois, P. J., "Decomposability, Instabilities, and Saturation in a multiprogrammed computer system," Comm. ACM 18, 7 (July 1975), 371-377.

[Den68a] Denning, P. J., "The working set model for program behavior." Comm. ACM 11, 5 (May 1968), 323-333.

[Den68b] Denning, P. J., "Thrashing: Its causes and prevention," AFIPS Conf. Proc. 33 (1968 FJCC), 915-922.

[DeG75] Denning, P. J., and Graham, G. S., "Multiprogrammed memory management," Proc. IEEE: Special Issue on Interactive Computer Systems (June 1975), 924-939.

[DeK75] Denning, P. J., and Kahn, K., "A study of program locality and lifetime functions," Proc. 5th ACM Symp. on Operating System Principles (Nov 1975), 207-216.

[LeP75] Leroudier, J., and Potier D., "New results on a model of virtual memory systems for performance evaluation," Technical Report, IRIA-Laboria, Rocquencourt, 78150 Le Chesnay, France (June 1975).

[Mor72] Morris, J. B., "Demand paging through the use of working sets on the Maniac II," Comm. ACM 15, 10 (October 1972), 867-872.

[Mun75] Muntz, R. R., "Analytic modeling of interactive systems," Proc. IEEE 63, 6 (June 1975), 946-953.

[MuW74] Muntz, R. R. and Wong, J., "Asymptotic properties of closed queueing network models," Proc. 8th Princeton Conf. on Information Science and Systems, Dept. Elec. Engrg, Princeton U. (March 1974).

[Pri73] Prieve, B. G., "Using page residency to determine the working set parameter," Comm. ACM 16, 10 (October 1973), 619-620.

[RoD72] Rodriguez-Rosell, J., and Dupuy, J. P., "The evaluation of a time sharing page demand system." AFIPS Conf. Proc. 40 (1972-SJCC), 759-765.

[RoD73] Rodriguez-Rosell, J., Dupuy, J. P., "The design, implementation, and evaluation of a working set dispatcher," Comm. ACM 16, 4 (April 1973), 247-253.

[Sal74] Saltzer, J. H., "A simple linear model of demand paging performance," Comm. ACM 17, 4 (April 1974), 181-185.

[WeO69] Weizer, N., and Oppenheimer, G., "Virtual memory management in a paging environment," AFIPS Conf. Proc. 34 (1969 SJCC), 234ff.

LEGEND FOR FIGURES 6 TO 20

..... I Bound
_ _ _ _ _ _ L(n)/S Bound
..... U_1 (page device)
———— ø(n)/aø

Operating Points:
- □ L = S Rule
- △ Knee Rule
- ○ 50% Rule

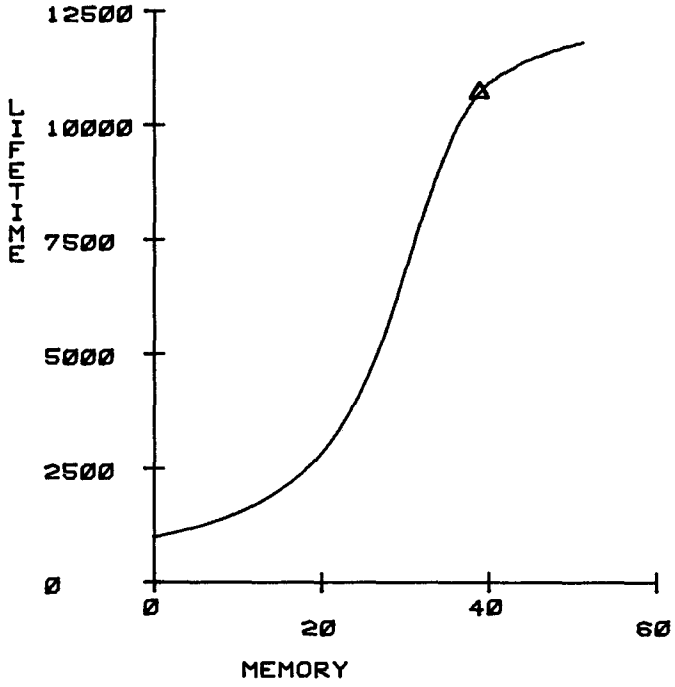

Figure 6: Lifetime Curve

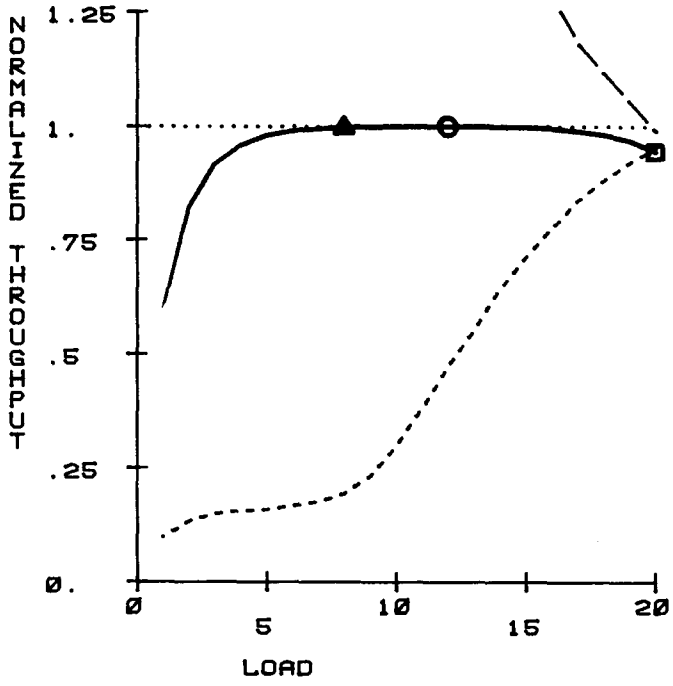

Figure 7: M=300. L_{max}=12
Y=60. S=2000. R=2.
5% U Range: 6.1 to 1.1

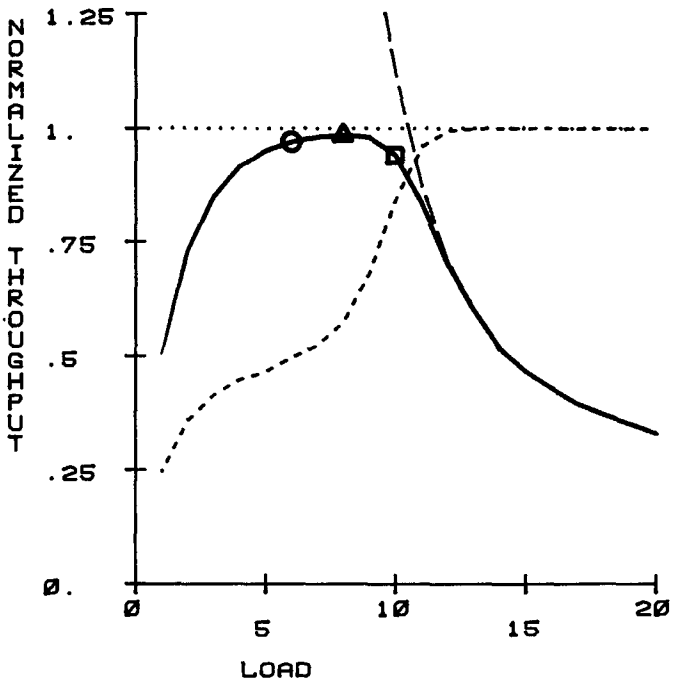

Figure 8: M=300. L_{max}=12
Y=60. S=6000. R=2.
5% U Range: 2. to 1.1

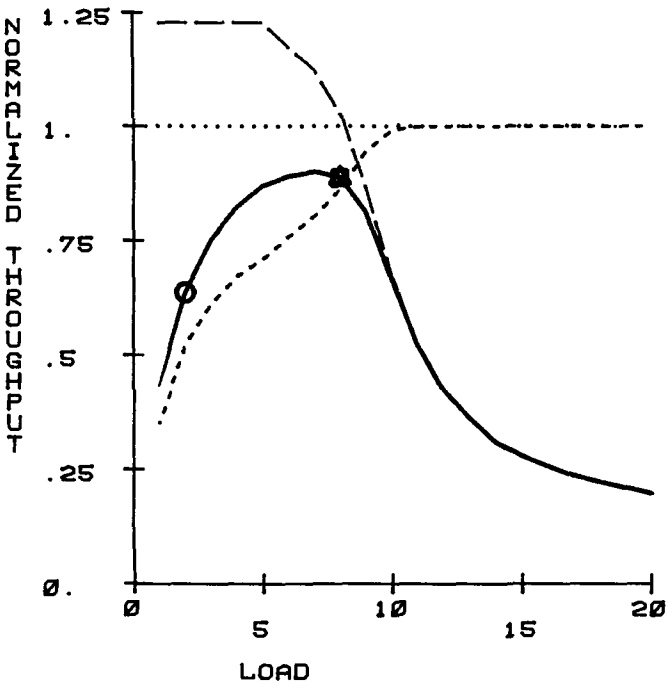

Figure 9: M=300. L_{max}=12
Y=60. S=10000. R=2.
5% U Range: 1.2 to 1.

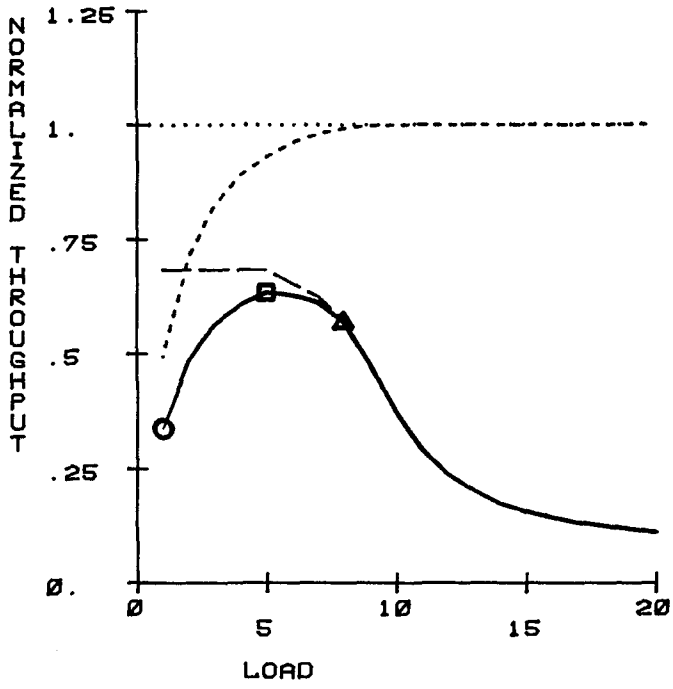

Figure 10: M=300. L_{max}=12
Y=60. S=18000. R=2.
5% U Range: .68 to .62

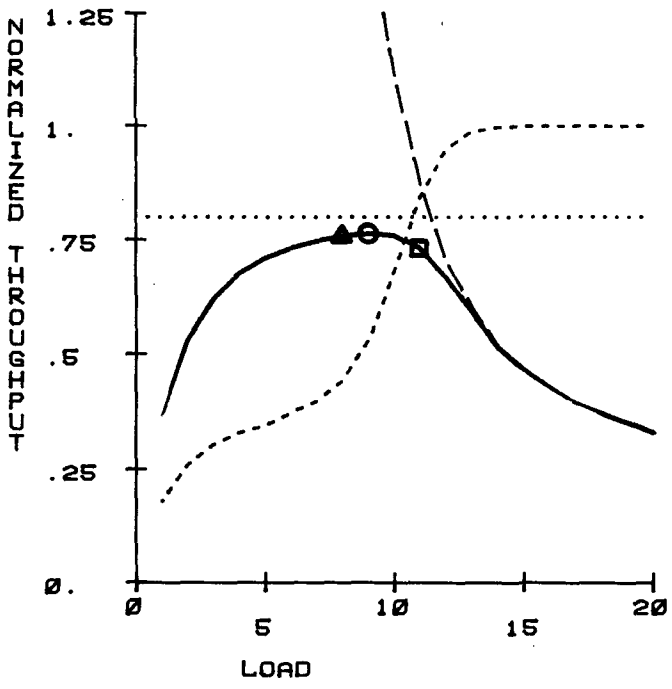

Figure 11: M=300. L_{max}=12
Y=60. S=6000. R=.8
5% U Range: 2. to .87

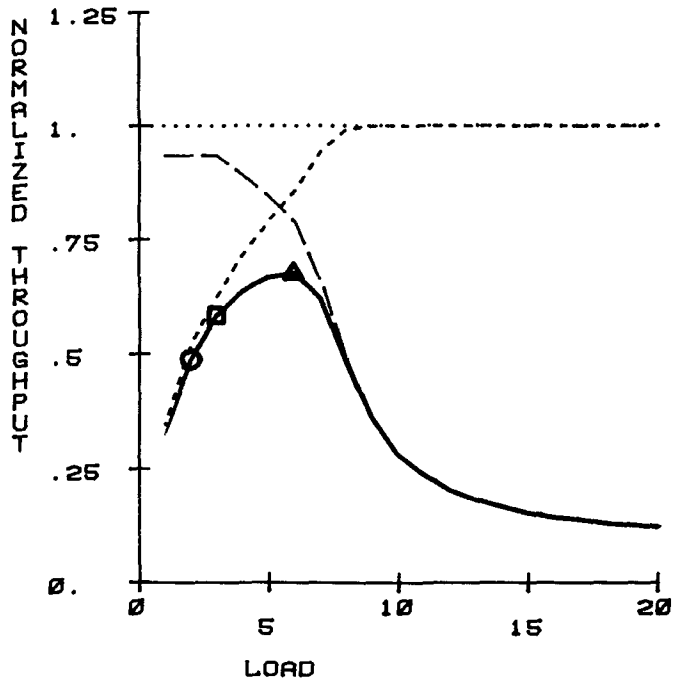

Figure 12: M=240. L_{max}=12
Y=80. S=13800. R=1.
5% U Range: .84 to .79

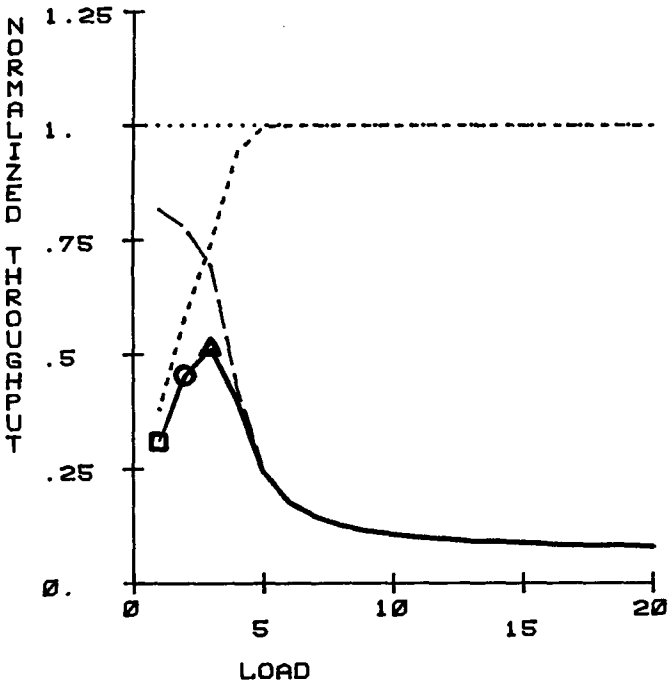

Figure 13: M=120. L_{max}=12
Y=80. S=15800. R=1.
5% U Range: .69 to .69

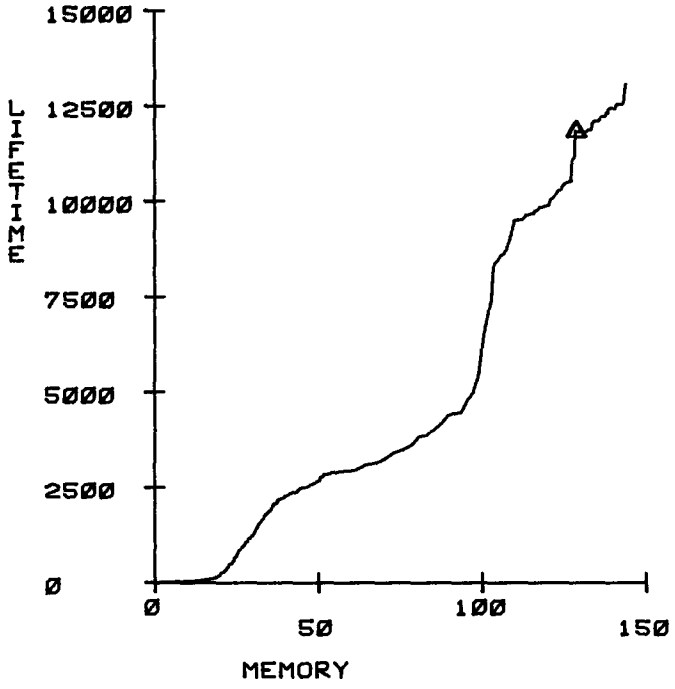

Figure 14: Lifetime Curve

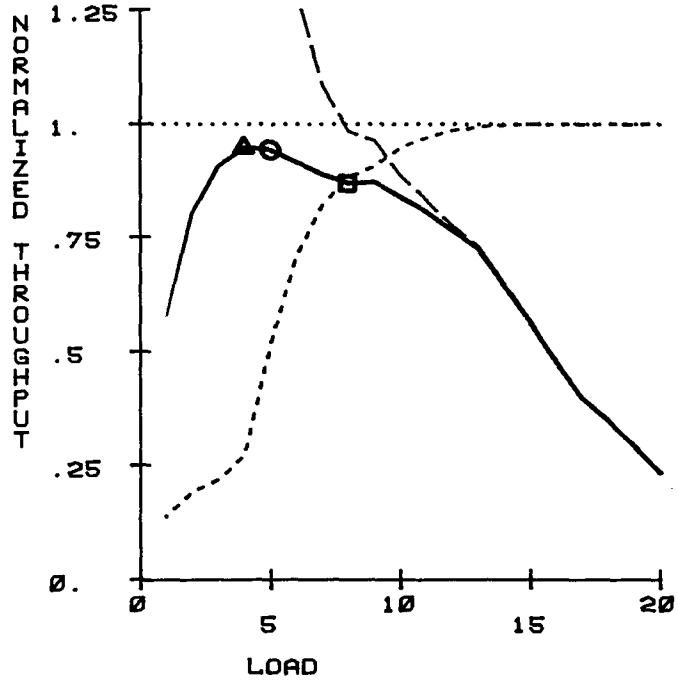

Figure 15: M=500. L_{max}=12420
Y=140. S=3000. R=2.
5% U Range: 4.1 to 1.3

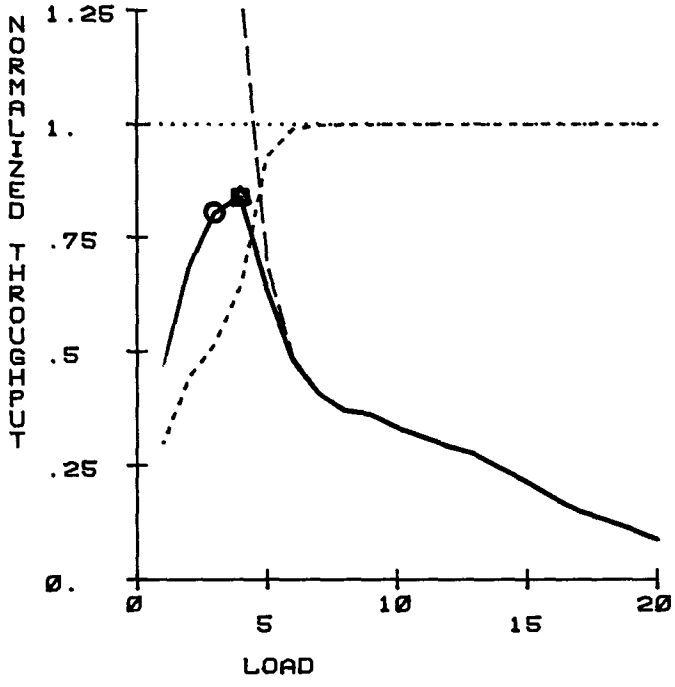

Figure 16: M=500. L_{max}=12420
Y=140. S=8000. R=2.
5% U Range: 1.6 to 1.3

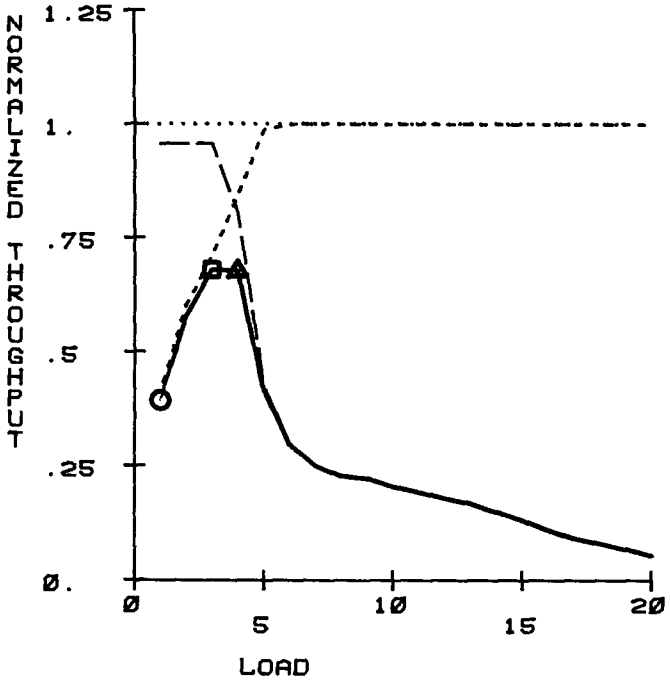

Figure 17: M=500. L_{max}=12420
Y=140. S=13000. R=2.
5% U Range: .96 to .8

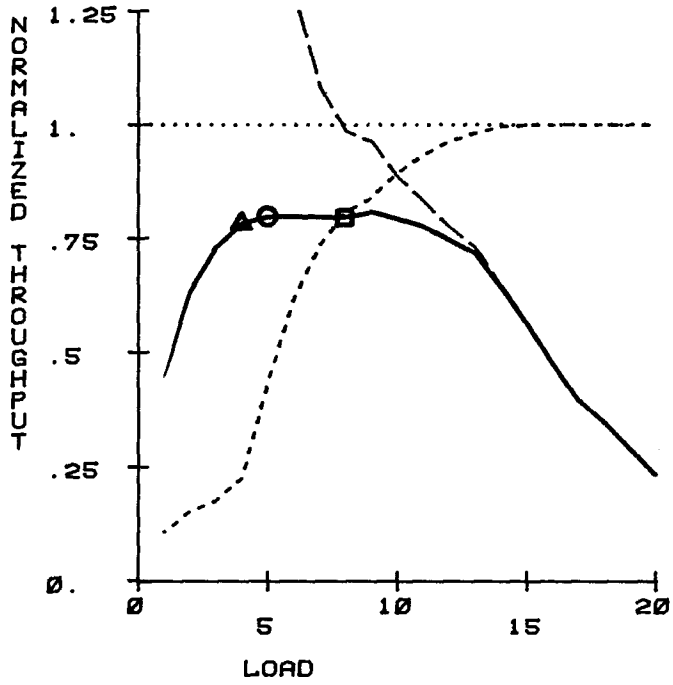

Figure 18: M=500. L_{max}=12420
Y=140. S=3000. R=1.
5% U Range: 3.5 to .85

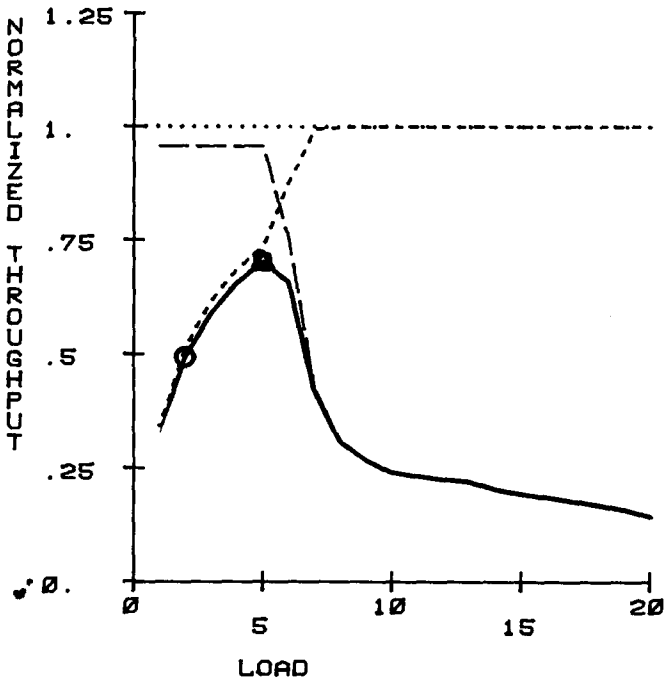

Figure 19: M=700. L_{max}=12420
Y=140. S=13000. R=1.
5% U Range: .96 to .96

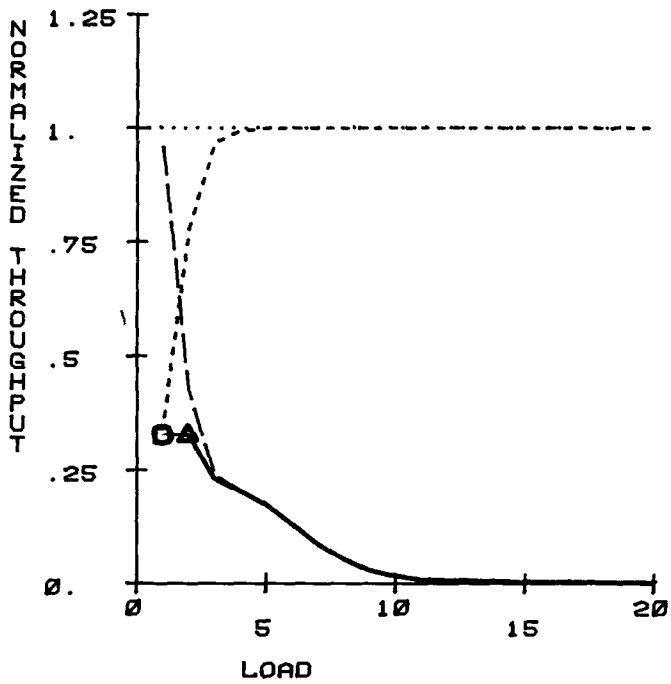

Figure 20: M=200. L_{max}=12420
Y=140. S=13000. R=1.
5% U Range: .96 to .42