AN ANALYSIS OF THE PERFORMANCE OF THE PAGE FAULT FREQUENCY (PFF) REPLACEMENT ALGORITHM.*

E. Sadeh**
Dept. of Electrical Engineering
University of Waterloo
Waterloo, Ont., Canada N2L 3G1

Abstract: Most of the replacement algorithms devised and implemented largely depend on program behavior, in other words, to optimally select the parameters of these algorithms program behavior or at least a probability model of it should be known. The page fault frequency (PFF) algorithm adapts to dynamic changes in program behavior during execution. Therefore its performance is expected to be less dependent on prior knowledge of the program behavior during execution. Therefore its performance is expected to be less dependent on prior knowledge of the program behavior and input data.

The PFF algorithm uses the measured page fault frequency (by actually monitoring the inter-page fault interval) as the basic parameter for memory allocation decision process.

In order to analyze the performance of the PFF algorithm, a mathematical model was developed. The resultant random process is the memory space allocation for a program as a function of the processor time (virtual time). This random process can be analyzed using the method of imbedded Markov chains. The parameter obtained from this analysis are the distributions of the memory allocation during processing interval and during page waiting intervals, the average page fault rate and the expected space time product accumulated by the program.

The input parameters for the model were obtained from address traces of two programs. The results of the model were validated by simulation.

Key words and phrases: paged virtual memory systems, pff replacement algorithm, reference string, imbedded markov chain, memory requirement process, page fault rate, space time product.

CR categories: 4.3, 4.35, 8.1

Introduction

The PFF algorithm was first suggested in 1972 by Chu and Opderbeck (C1). It attempts to dynamically control the rate of page faults produced by a program running in a paged virtual memory environment, by varying the memory space allocated to the program. The underlying idea is to make use of the inverse relation between page fault rate and memory allocation.

The PFF algorithm measures the inter page fault intervals during execution. At page fault times, it compares them with an a priori selected threshold T. If the inter page fault interval exceeds T then all the pages in main memory which were not referenced during this interval are dumped. Otherwise no page is dumped and the allocation increases by one page frame (allocated for the missing page). Since the PFF algorithm is based on the measured inter page fault intervals, no process under the management of this algorithm should be allowed to collect all its N referenced pages in main memory, lest no more page faults will be generated by the process for the rest of its execution. In practice N is not known a priori. Therefore, the solution of this problem requires a time limit, Z, on the measured inter page fault interval. Whenever the time limit is reached, some memory allocation decision is made without waiting for a page fault to occur. The PFF algorithm suggested by Chu and Opderbeck (C1) does not provide for the time limit interrupt feature.

* This work was supported in part by the Office of Computing Activities of the National Science Foundation under grant GJ-32504.

** The author was with the Department of Electrical Engineering, University of Minnesota, Minneapolis, Minnesota 55455.

There are three basic approaches for performance evaluation of memory management algorithms,

(1) Actual construction and measurement,
(2) Simulation,
(3) Analytical model.

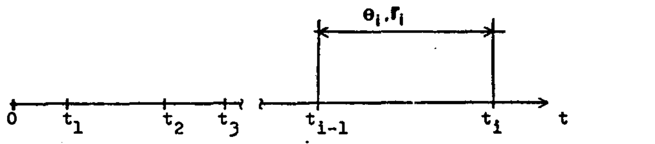
The first approach is expensive and therefore impractical. Simulation consumes a large amount of computer time and by its very nature is inconclusive. It has been used in many analyses of paging algorithms (C1, O1, S3) more so because analytical models are very difficult to form than because of its own merits. The approach considered here is the third approach. The purpose of this paper is to develop a mathematical model for the page fault frequency (PFF) replacement algorithm.

In the development of an analytical model for a paging algorithm's performance, the algorithm may be considered as a system which processes the reference string (D2), generated by the program, as an input source and generates, as output, sequences of memory allocations and page faults. The performance of the paging algorithm is measured by cost functions defined on the output such as average memory requirement, page fault rate and space time product (A2, C2, D1, D2, D4, S1).

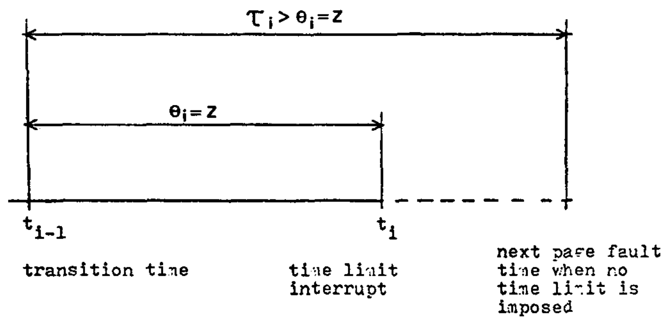
A description of the PFF algorithm

Some notation used in this paper is given in Fig. 1. Let t be the processing time of the process (this excludes time spent by the process during I/O operations). The unit of time is the average memory access time. Let $t_1, t_2, \ldots, t_{i-1}$ ... be the instances when a transition takes place (either a page fault occurs or the time limit is reached). The ith inter transition interval is $\theta_i = t_i - t_{i-1}$. The number of distinct pages referenced during θ_i is denoted by r_i (the last reference before the transition being the first reference in the new interval). Let the ith forward page fault time be denoted by τ_i. This is the time which it takes for the process to produce a page fault after the transition at t_{i-1} when no time limit is enforced. If $\tau_i < Z$ then $\tau_i = \theta_i$, otherwise $\tau_i > \theta_i = Z$. The description of the operation of PFF algorithm in this notation is as follows. At transition time τ_i one of three allocation decisions is made according to the length of τ_i.

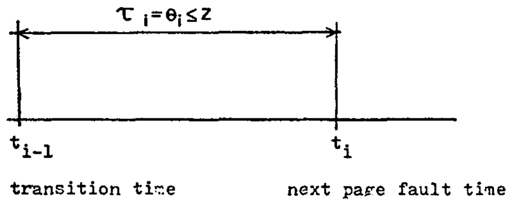
If $\tau_i < T$, the allocation is increased by allocating an extra page frame for the missing page while dumping no page from main memory.



a. Transition times and inter transition intervals



b. Forward page fault time $\tau_i > \theta_i$



c. Forward page fault time $\tau_i = \theta_i$

Figure 1. Notation

If, however, $T < \tau_i < Z$ pages in main memory which are not referenced during τ_i (thus not referenced for at least T memory references) are dumped out.

Finally if the time limit Z is reached before a page fault occurs, i.e., τ_i is going to exceed Z, an interrupt occurs and the pages not referenced during this period are freed. No extra page frame is allocated. For the development of the analytical model of the performance of the PFF algorithm it is sufficient to make some general assumption about program behavior. These assumptions are discussed in the next section. However to reduce the number of parameters we restrict in this paper the input source characteristics to these which are described by the simple LRU stack model of program behavior (A2, C2, D2). This model is briefly described below.

The model for reference string generation

The property which seems universal in programs is the locality of references, (C2, D1, D2, D3, D4, S2, S3) consequently, models for reference string generation should reflect this phenomenon. Several models which are analyzed and compared with simulations are reported by Spirn and Denning (S3), Coffman and Ryan (C2), Arvind et al. (A2) and Shedler and Tung (S2).

The most successful model reported in (S3) is the simple LRU stack model. The model is based on the LRU stack in which the descriptors of the program's pages are arranged according to recency of use. The LRU stack may be described as a vector $S(t) = (S_1(t), S_2(t), \ldots, S_N(t))$ in which $S_i(t)$ is the page descriptor in the i^{th} position of the stack after the t^{th} reference (i.e., the i^{th} most recently referenced page) and N is the number of referenced pages in the program. The stack distance d_t is the position of the page referenced at time t in the stack $S(t-1)$.

To represent program behavior the model associates probabilities $b_1, b_2 \ldots, b_n$ with stack positions $1, 2, \ldots, k$ such that

$$\Pr[d_t = i] = b_i \qquad 0<b_i< 1,\ t>0$$

and

$$\sum_{i=1}^{n} b_i = 1.$$

A descriptor of a referenced page is moved to the top of the LRU stack while all the descriptors above it in the stack are moved one position down. The sequence $S_1(1), S_1(2), \ldots, S_1(t), \ldots$ is the reference string generated by the model.

The Memory Requirement Process

Let w(t) be the memory requirement process where t is the virtual processing time and w(t) is the memory requirement (in pages) at time t. This process is a discrete random process, characterized by $\Pr\, w(t)=k$, $k<N$, the probability of memory requirement of size k at any given processing time t where N is the total number of pages referenced during the execution of the program. The process w(t), under the PFF algorithm, is a staircase function of the processing time (Fig.2), where transition from level to level occurs only at transition times.

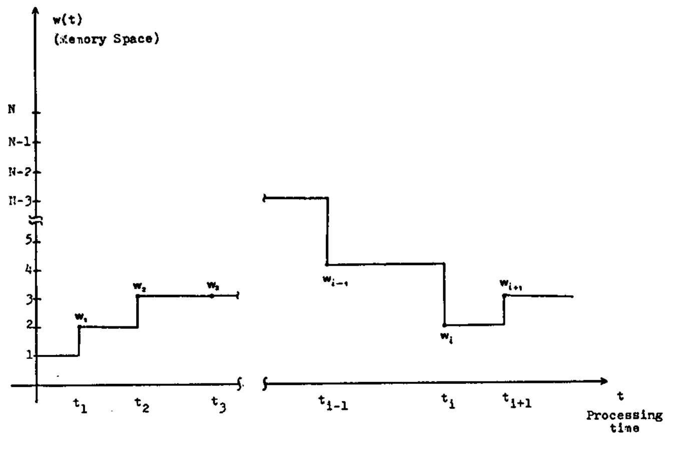


Figure 2. The Memory Requirement Process

Let w_i be the allocation size immediately after the allocation decision at t_{i-1} (i.e., the process level in the interval (t_{i-1}, t_i)). There are two assumptions about program behavior on which the development of the model is based:

A1. The i^{th} forward page fault time τ_i is a random variable which depends only on the number w_{i-1} of page frames allocated for the process during this time.

A2. r_i, the number of distinct pages referenced during r_i is a random variable which depends on w_{i-1} and τ_i only.
The simple LRU stack model of program behavior satisfies the above assumptions.

The level w_i of the process w (t) during the i^{th} inter transition interval θ_i, is a random variable. Based on assumptions A1 and A2 it can be shown that the sequence $w_1, w_2, \ldots, w_k, \ldots$ forms a Markov chain. This is the imbedded Markov chain of the process w(t). The transition matrix governing the imbedded Markov chain will be denoted by $\Pi = \pi_{\ell k}$ where

$$\pi_{\ell k} = \Pr[w_i = k \mid w_{i-1} = \ell]$$

From state k the possible next states are 1, 2,3,..., min(k+1,N). It is not difficult to show that the Markov chain is aperiodic and irreducible, and therefore steady state probabilities exist. We assume the length of the page waiting interval (i.e. the interval between a page fault interrupt and the resumption of execution) to be a constant R. This is justified by the fact that the actual page waiting intervals are independant of the paging algorithm employed.

Considering time limit interrupt extremely rare (for reasonably large Z), the imbedded Markov chain represents the memory requirement during page waiting intervals. (The exact random process can be developed but it is complicated).

The memory requirement process, w, is not a Markov chain since the holding times in each state are random variables. However, the knowledge of the distributions of these holding times and the transition probabilities of the imbedded Markov chain, is sufficient to find $\Pr[w(t) = K]$. This is done in the following way:

Assume that at a given time $\sigma = o$ the process w enters state ℓ. What is the probability that at time $\sigma = t$ the process is in state k? Let us denote this probability by $P_{\ell k}(t)$. Further, let θ_1 be the inter transition interval starting at $\sigma=0$. Following techniques of Renewal Theory (F1) we consider two cases: a. $\theta_1 > t$, i.e. no transition takes place in the interval (0,t). b. $\theta_1 < t$, at least one transition occurs. In case a. the process must be in state ℓ at the end of the interval (0,t). The probability of this case is $d_\ell(t) \triangleq \Pr[\theta_1 > t \mid w = \ell]$ In the other case, the next state n ($n=1,2,\ldots\ell+1$) and the length m ($m=1,2,\ldots,t$) of θ_1 is determined by the joint probability

$$q_{\ell k}(m) = \Pr\ \theta_1 = m,\ w_1 = k \mid w_o=\ell]$$

Finally, due to assumptions A1 and A2, the probability that the process w is in state k at time $\sigma=t$ given that it enters state n at time $\sigma=m$ is $P_{nk}(t-m)$.

Considering all the cases we thus obtain the recursion formula:

$$p_{\ell k}(t) = d_\ell(t)\,\delta_{\ell k} + \sum_{n=1}^{N} \sum_{m=1}^{t} q_{\ell n}(m)\, p_{nk}(t-m)$$

$$\ell,\ k = 1, 2, \ldots, N;\ t=1,2,\ldots$$

where $\delta_{\ell k} = \begin{cases} 1 & k=\ell \\ 0 & k\neq\ell \end{cases}$

Next let P(t), D(t), Q(t) be NxN matrices such that $P(t) \triangleq \{p_{\ell k}(t)\}$, $D(t) \triangleq \{d_\ell(t)\,\delta_{\ell k}\}$, $Q(t) \triangleq \{q_{\ell k}(t)\}$.

Then writing the above equation in a matrix form we have the following theorem.

Theorem 1

The matrix P(t) satisfies the following matrix difference equation.

$$P(t) = D(t) + \sum_{m=1}^{t} Q(m)\, P(t-m) \qquad (1)$$

with the initial condition

$$P(0) = 1.$$

where I is the unit matrix.
Since, initially one page frame is allocated to a program under the PFF replacement algorith ($w_o = 1$) it is easy to see that the probabilities are the entries of the first row of matrix P(t). To solve equation (1), the entries of D(t) and Q(t) should be calculated.

$$q_{\ell k}(t) \triangleq \Pr[\theta_1=t,\ w_1=k|w_o=\ell] =$$

$$\Pr[\theta_1=t|w_1=k,\ w_o=\ell].\quad \Pr[w_1=k|w_o=\ell]$$

$$= \Delta_{\ell k}(t)\ \pi_{\ell k}.$$

where $\Delta_{\ell k}(t)$ is probability mass function of the holding time in state ℓ given that the next state is k.

And

$$d_\ell(t) = 1 - \sum_{m=1}^{t} \sum_{k=1}^{N} \Delta_{\ell k}(m)$$

The probabilities $\Delta_{\ell k}(m)$, $m = 1,2,\ldots,t$; $\ell,k = 1, 2, \ldots, N$, as well as the transition probabilities of the imbedded Markov chain, can be calculated in terms of the parameters T,Z and the stack probability vector $(b_1,b_2,\ldots, b_N)$ of the simple LRU stack model (S1).

These values can, alternatively, be collected as relative frequencies from reference string of real programs. Equation (1), then, can be solved numerically on a computer.

We are mostly concerned however, with programs with very long reference strings, and with the memory requirement of the process after long execution times. Therefore, the long term behavior of the process w is of the prime interest. In the next section it is shown that the process w(t) reaches a steady state, and as usually is the case the solution of equation (1) becomes much simpler when we restrict it to the steady state.

3. The steady state behavior of the process w(t)

Intuitively, one would expect w(t) to reach a steady state after a sufficiently long time. This plausible conclusion follows from assumptions A1 and A2. The imbedded Markov chain possesses steady state probabilities and the length of the inter transition intervals depends on the states of the process. Thus for each state the expected value of the holding time at that state is constant, and it is independent of the particular time the process enters that state.

Let μ_ℓ denote the expected value of the holding time at state ℓ, i.e.,

$$\mu_\ell = \sum_{m=1}^{\infty} m \Pr[\theta_i=m|w_{i-1}=\ell] \qquad \ell = 1,2,\ldots,N$$

Let γ_ℓ ($\ell = 1,2,\ldots,N$) denote the steady state probability of occupying state ℓ, in the imbedded Markov chain. Then the following theorem determines the steady state probabilities of the random process w(t).

Theorem 2

The steady state probabilities of w(t) are independent of the initial state and are given by

$$P_k \triangleq \lim_{t\to\infty} \Pr[w(t)=k] = \frac{\mu_k}{\mu}\gamma_k \quad k = 1,2,\ldots,N \qquad (13)$$

where

$$\mu = \sum_{k=1}^{N} \gamma_k\,\mu_k$$

This theorem has an intuitive appeal. One would expect that the steady state probabilities of the process w(t) will be proportional to both the conditional expectation of the holding time and the steady state probabilities of the imbedded Markov chain. Thus

$$P_k = \alpha\mu_\ell\gamma_\ell$$

where α is a normalizing constant
Since

$$\sum_{k=1}^{N} P_k = 1$$

$$\alpha = [\sum_{k=1}^{N} \gamma_\ell\mu_\ell]^{-1} = \frac{1}{\mu}$$

A rigorous proof of theorem 2, however, is involved. It is approached by the use of generating functions. The generating function $\hat{P}(x)$ of P(t) is found easily from equation (1). Then employing the final value theorem, i.e.

$\lim_{t\to\infty} P(t) = \lim_{x\to 1} (1-x)\,\hat{P}(x)$ the theorem is proved.

From theorem 2, the evaluation of the steady state probabilities, $P_k(k=1,\ldots,N)$, requires the knowledge of the steady state probabilites of the imbedded Markov chain γ_ℓ ($\ell = 1,2,\ldots,N$) and the expected values of the holding times: μ_ℓ ($\ell = 1,2,\ldots,N$). The steady state probabilities of the imbedded Markov chain are found by solving the equations

$$\gamma = \gamma\Pi$$

and

$$\sum_{\ell} \gamma_\ell = 1$$

Again, the parameters μ_ℓ ($\ell=1,2,\ldots,N$) can be calculated directly in terms of the parameters T, Z and the stack probability vector $(b_1,b_2\ldots,b_N)$ (S1), or alternatively, can be collected as statistics from reference strings of real programs.

Based on these results we can calculate the following parameters in steady state: E_w, the expected memory requirement during processing intervals. And the approximation (for large Z) of E_π, the expected memory requirement during page waiting intervals and the average page fault rate f.

$$E_w = \sum_{k=1}^{N} kp_k \;,\; E_\pi \simeq \sum_{k=1}^{N} k\gamma_k$$

$$f \simeq [\sum_{k=1}^{N} \mu_k \mu_k]^{-1} = \frac{1}{\mu} .$$

An exact expression can be found for f as function of T and Z (S1).

In the next section a recursive formula for the expected STP (ESTP) as a function of the processing time is developed and the steady state behaviour of the ESTP as the processing time becomes large is investigated. An approximate value of the steady state rate of growth of the ESTP in real time is obtained. The space is measured in pages and the time unit is the main memory access time.

The Expected Value of the Space Time Product (ESTP)

Let $s_\ell(t)$ denote the expected value of the space time product accumulated by the process in the interval (0,t) on the processing time axis given that at time zero it enters state ℓ.

Let R be the expected value of the page waiting time.

Let J be the overhead time needed to handle a time limit interrupt.

Finally, let

$$\psi_\ell(t) \;\underline{\Delta}\; \sum_{n=t+1}^{\infty} \beta_\ell(n) = \Pr[r_1>t|w_o=\ell]$$

and

$$S(t) \;\underline{\underline{\Delta}}\; \begin{bmatrix} s_1(t) \\ s_2(t) \\ \cdot \\ \cdot \\ \cdot \\ s_N(t) \end{bmatrix} \qquad V(t) = \begin{bmatrix} v_1(t) \\ v_2(t) \\ \cdot \\ \cdot \\ \cdot \\ v_N(t) \end{bmatrix}$$

The following theorem determines a recursive formula for S(t).

Theorem 3

$$S(t) = V(t) + \sum_{m=1}^{t} Q(m)\, S(t-m) \tag{3}$$

where

$$v_\ell(t) = \begin{cases} \ell(t\psi_\ell(t)+R(1-\psi_\ell(t))+ \Sigma \;\; m\beta_\ell(m) & \text{if } t \leq Z \\ \ell(J\psi_\ell(Z)+R(1-\psi_\ell(Z))+ \mu_\ell) & \text{if } t > Z \end{cases}$$

and the initial condition is

$$S(0) = 0$$

Equation (15) can be solved numerically for the expected STP at any given processing time.

Again the long term behaviour of the ESTP is of interest.

Long Term Behaviour of the ESTP

Clearly the space time product tends to infinity with processing time. But since the space time process is essentially an integration over the memory requirement process and since the memory requirement process reaches a steady state (i.e., the expected value becomes constant) one would expect the ESTP to become asymtotically linear with the processing time.

Let E_w, E_π be the expected steady state values of the process w(t) and the imbedded Markov chain, respectively.

Let

$$E_\pi(\tau\leq Z) \;\underline{\underline{\Delta}}\; \sum_{k=1}^{N} k\gamma_k \Pr[\tau_i \leq Z|w_{i-1}=k]$$

and

$$E_\pi(\tau>Z) \;\underline{\underline{\Delta}}\; E_\pi - E_\pi\,(\tau\leq Z)$$

The long term behaviour of the ESTP of a process running under the PFF replacement algorithm is determined by the following theorem.

Theorem 4

For large t and $\ell = 1,2,\ldots,N$

$$s_\ell(t) = \psi t + \psi + \nu_\ell \tag{4}$$

where

$$\psi = E_w + \frac{1}{\mu}[RE_\pi(\tau\leq Z) + JE_\pi\,(\tau>Z)]$$

and ν_ℓ is a constant depending on ℓ.

The theorem indicates that the ESTP approaches asymptotically a straight line with a slope ψ and an intercept $\psi + \nu_\ell$, as the processing time becomes large.

Theorems 3,4 are developed in much the same way theorems 1 and 2 are. This time, however, the question to be answered is: given that the process w enters state ℓ at time σ=o what is the expected value of the space time product accumulated by the process in the interval (σ = o,σ = t)? Again, we consider two cases according to whether

$\theta_1 > t$ or $\theta_1 \leq t$ in order to find a recursive formula. This leads to theorem 3. Applying again the generating function technique to equation (3) and using the final value theorem we arrive at theorem 4.

We have calculated the ESTP in processing time t. However the observed time is the real time σ which consists of the processing time t and the time spent by the program waiting for missing pages and the overhead incurred by the time limit interrupts. The fraction of references causing either page faults or time limit interrupts is approximately $1/\mu$ where μ is the average inter transition interval, the fraction of those causing page faults is $f = 1/u$ where u is the average inter page fault interval.

Hence

$$\sigma/t \simeq (1 + fR + (\frac{1}{\mu} - f)\,J) \triangleq \varepsilon$$

and from Theorem 4 the rate of growth of the ESTP in real time is approximately

$$g = \frac{\psi}{c} = \frac{E_w + \frac{1}{\mu}\,[RE_\pi(\tau \leq Z) + JE_\pi(\tau > Z)]}{[1 + fR + (\frac{1}{\mu} - f)\,J]}$$

6. NUMERICAL AND SIMULATION RESULTS

In the previous sections we have developed a mathematical model for the performance of the PFF paging algorithm. In this section numerical and simulation results are presented and discussed.

Numerical Results

Ideally we wish to have closed form expressions for performance (output) parameters such as page fault rate, STP, and average memory allocation, in terms of the model's input parameters: the stack probability vector, T, and Z.

Such expressions would give direct functional relations between output and input parameters, thus making both optimization and comparison with other algorithms easier.

Unfortunately, the presence of a Markov chain in the analysis and the dimension of the transition matrix rule out such a possibility. In any case, even partial closed form solutions are so complicated that a recursive formula is usually preferred. Results for the model however can be obtained numerically using the computer.

As input parameters we have selected stack probability vectors which were extracted from the execution of programs on the CDC 6600 at the University of Minnesota computer center. (A2)

Two programs were selected for this study.

Program A: An APL program. The trace tape contains over a million references.

Program B: A Fortran compiler (MNF) in execution. The trace tape contains 660,000 references.

Numerical results were obtained for three purposes:
(i) to investigate relations between output parameters and input parameters; (ii) to find out about inter relations between output parameters; (iii) to validate the model by comparison with simulation results.

Figures 3 and 4 depict the relations between the average page fault (pfr) and the average memory allocation during processing intervals, E_w (solid lines) and the average memory allocation during page waiting intervals, E_π (dashed lines). For page size (PS) 1024, program A has 24 referenced pages and program B has 23. The parameter Z is fixed at 50,000 references, and the parameter T is varied from 10 to 10,000 references. In both programs the points for both E_π and E_w lie on a straight line as long as E_π, E_w are far enough from N the total number of references pages (below 15 pages in the figure). Hence within this range the pfr is an exponential function of E_w as well as an exponential function of E_π. This observation finds more support in (K1).

Program A shows a stronger locality then program B in the sense that the stack probability vectors, which were extracted from the reference strings of both programs, are such that for every i = 1,2,..,N

$$\sum_{K=1}^{i} b_K \text{ (prog. A)} \geq \sum_{K=1}^{i} b_K \text{(prog. B)}$$

If the PFF algorithm responds well to program requirements, one would expect that, for fixed T and Z, program A will yield, on the average, smaller memory allocation size and lower page fault rate than program B. That this is the case, is evident from a comparison of Figures 3 and 4. However, using the simple LRU stack model it is impossible to check transients. Thus, an important question such as how fast the PFF algorithm responds to a change in locality size, cannot be answered using this model.

Another phenomenon which is intuitively expected for any program is that $E_\pi < E_w$ for every T. This is because one expects that large allocation will yield on the average longer processing interval than a smaller allocation will. Therefore the distribution of the memory requirement during processing intervals should be biased toward the large allocations. The page waiting intervals, however, are independent of the allocation size and therefore no such bias exists. This is clearly the case in both Figure 3 and 4.

Simulation Results

The validation of the model may be divided into two separate problems:

1. How well the model describes the PFF algorithm performance given that assumptions A1 and A2 about program behaviour are correct.

2. How valid are assumptions A1 and A2.

The first problem can be tackled as follows: Instead of using the real reference strings as input to the simulation, synthetic reference

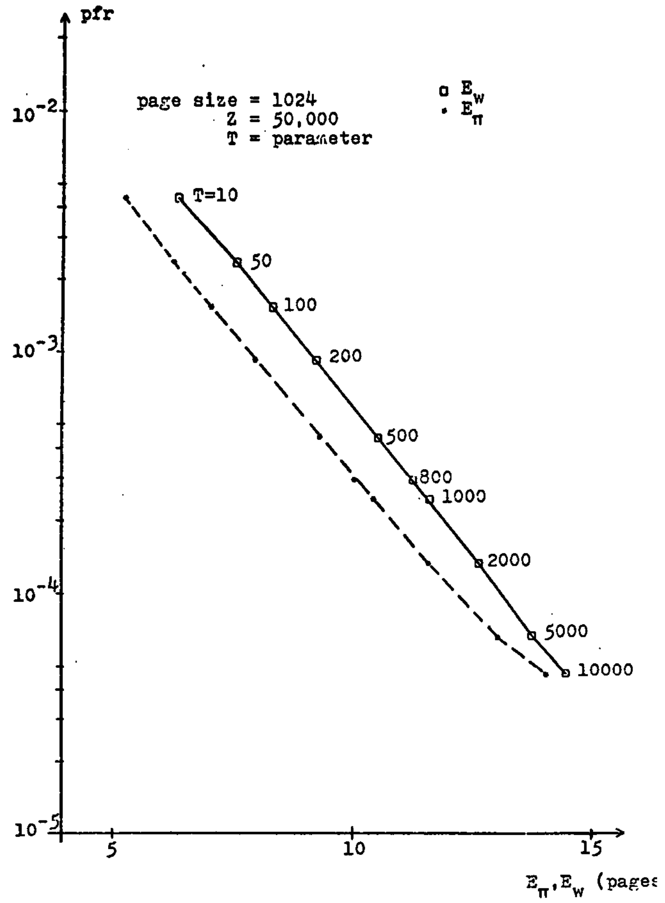


Fig. 3 pfr vs. E_π and E_W (Program A)

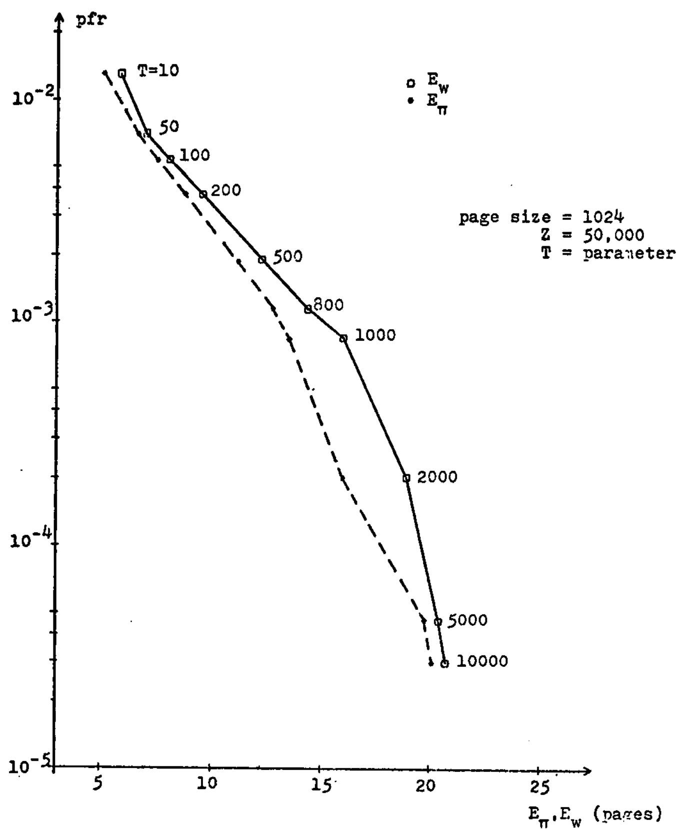


Fig. 4 pfr vs. E_π and E_W (Program B)

strings are used. They are generated by performing a Monte Carlo experiment in which stack distances are randomly generated in accordance with the stack probability vectors (which were extracted from the real reference strings). The results are compared then, with the numerical results obtained from the model where the input parameters are the same stack probability vectors.

Several such simulation experiments were carried out for program A and program B, and for various values of the parameter T. Parameter Z was fixed at 50,000. Each simulation experiment processed a reference string of one million references.

The results are shown in figures 5 and 6. These figures depict pfr average allocations of memory in real time and in steady state.

These figures show that the simulation results agree very well with the model in both page fault rate and memory allocation predictions as long as T is smaller than 5000.

The discrepancy between simulation results and model results for T = 5000 and T = 10,000 can be explained by lack of statistical significance. The actual sample size in the simulation relates to the number of page faults since this determines the number of transitions in the imbedded Markov chain. This number becomes very small for both programs as T becomes large.

An experiment to answer the second problem is easier to handle. Unfortunately such experiments were not attempted. There are two ways, however, to answer this problem:

1. Collect statistics about inter transition intervals for various allocation sizes by directly processing the real reference strings. The generality of the answer requires, however, that experiment will be done over a reasonable sample size of reference strings.

2. The simulation experiment of the PFF algorithm is fed by the real reference strings. The results are compared with the numerical results of the model where the input parameters are the steady state probabilities of the imbedded Markov chain represented by relative frequencies and the average time spent in each state. These parameters can be extracted by directly processing the real reference strings.

7. Conclusions

This paper discusses the development of a mathematical model for the performance of the PFF algorithm. The algorithm analyzed is essentially the same as the one suggested by Chu and Opderback (Cl) except for a necessary modification to handle the case where all the program's pages are accumulated in main memory.

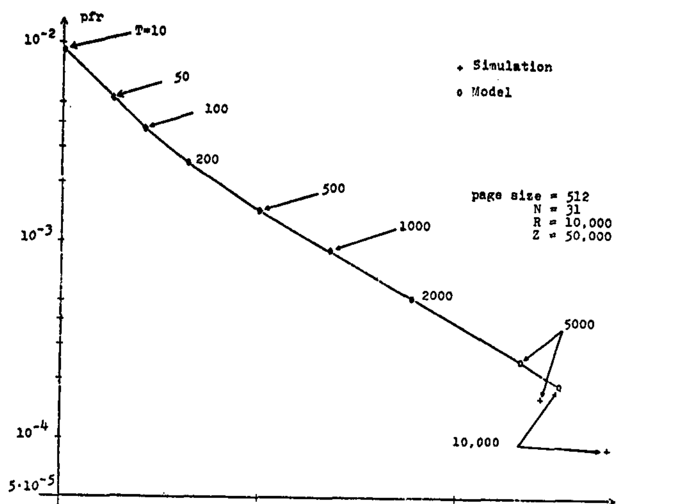


Fig. 5 Page Fault Rate vs. Average Memory Allocation (Program A)

This model uses the simple LRU stack model of program behaviour in order to get numerical results. The main drawback of the simple LRU stack model is that it generates reference strings that do not reflect transitions between localities. This drawback limits the model of the algorithm's performance in exploring the adaptability of the algorithm to changes in locality size and its response during transitions. However, no presently available satisfactory model of program behaviour incorporates localities of different sizes and the transitions between them. Furthermore, the simple LRU stack possesses parameters which can easily be extracted from tracing reference strings of programs in execution. It exhibits the phenomenon of locality fairly well, and finally the resulting model for the performance of the PFF algorithm is mathematically tractable, i.e., numerical results can be obtained.

Acknowledgement

The author would like to thank Dr. R.Y. Kain of the University of Minesota, the Department of Electrical Engineering, for his helpful comments and discussions and his advice which was very instrumental in the writing of the Ph.D. thesis on part of which this paper is based.

References

[A1] Arvind, "Experiments to evaluate working set properties," Dept. of EE, University of Minnesota, TR NSF-CCA-CJ 32504, Nov. 1973.

[A2] Arvind, R.Y. Kain and E. Sadeh, "On reference string generation process," Proc. 4th ACM Symposium on Operating System Principles, New York, Oct. 1973.

[C1] W.W. Chu and H. Opderbeck, "The page fault frequency replacement algorithm," FJCC 1972 AFIPS 41, pp. 597-609.

[C2] E.G. Coffman, T.A.Ryan, Jr., "A study of storage partitioning using a mathematical model of locality," Comm. ACM 15, 3, March 1972.

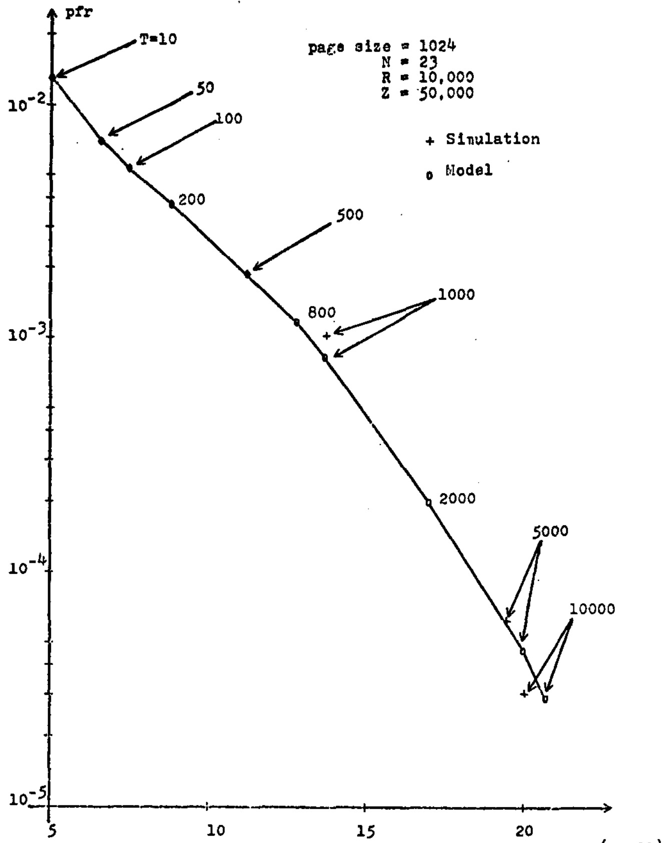


Fig. 6 Page Fault Rate vs. Average Memory Allocation (Program B)

[D1] P.J. Denning, "Virtual memory," Computing Surveys 2, 3, Sept. 1970.

[D2] P.J. Denning, J.E. Savage and J.E. Spirn, "Models of locality in program behavior," Princeton University, Dept. of EE, Computer Science TR-107, April 1972.

[D3] P.J. Denning, "On modelling program behavior," SJCC 1972, AFIPS 40, pp. 937-944.

[D4] P.J. Denning, and S.C. Schwartz, "Properties of working set model," Comm. ACM 15, 3, March 1972. Also see "Corrigendum: Properties of Working Set Model," Comm. ACM 16, 2, Feb. 1973.

[F1] N. Feller, An Introduction to Probability Theory and Its Applications, Vol. II, Wiley.

[K1] R.Y. Kain, "How to evaluate page replacement algorithm" Technical Report, Mpls., Minnesota, Oct. 1975.

[O1] H. Opderbeck and W.W. Chu, "Performance of the PFF replacement algorithm in a multiprogramming environment," IFIPS, Stockholm, Aug. 1974.

[S1] E. Sadeh, "Analysis of the page fault frequency algorithm," Ph.D. thesis, University of Minnesota, March 1975.

[S2] G.H. Shedler and C. Tung, "Locality in page reference strings," SIAM J. Comput. 1, 3, Sept. 1972.

[S3] J.R. Spirn and P.J. Denning, "Experiments with program locality," FJCC 1972 AFIPS 41, pp. 611-621.