Technical Report CS74023-R

Principle of Optimal Page Replacement and the LRU Stack Model[1]

Felix L. Lam[2]

Domenico Ferrari[3]

1. Work reported herein was supported in part by the U. S. Navy under Grant (NESC) N00039-71-C-0255.

2. Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061.

3. Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California 94720.

Abstract

Program reference strings generated by the LRU stack model are considered, and expressions for the expected times to next reference for all pages occupying different LRU stack positions are derived. Using these expressions, necessary and sufficient conditions as well as sufficient conditions on the distance distribution are obtained which guarantee implementation by the LRU replacement algorithm of the "informal principle of optimality" for page replacements. The sufficient conditions are found to be the same as those under which the LRU replacement algorithm is shown to be optimal. Relaxed conditions are also obtained for special cases where the number of page frames is fixed.

Key Words and Phrases : replacement algorithms, program models, program behavior, virtual memory, dynamic storage allocation.

CR Categories : 4.3

I. Introduction

One criterion that measures the performance of page replacement algorithms is the number of page faults, and therefore the cost, a replacement algorithm induces in processing a program reference string. The principle of optimality underlying the MIN replacement algorithm proposed by Belady [1] states that, to achieve a minimum number of page faults, the page with the longest time until its next reference should be replaced. The MIN replacement algorithms has been shown to be optimal by various authors [2, 3]. Clearly, the MIN replacement algorithm requires exact knowledge about future references in a program reference string and is therefore unrealizable. In cases where advance knowledge of program references is unknown and where only statistical statements can be made, a heuristic "informal principle of optimality" [4] has been used in designing realizable replacement algorithms which states that the page to be replaced is the one with the longest expected time until its next reference. The application of this heuristic principle is supposed to lead to a minimum expected number of page faults. Under general assumptions about program behavior, however, this is not always true as is shown in a counter-example in [5].

To determine the expected number of page faults, a few models for program reference string generation have been studied. The simplest of these models is known to be the independent reference model and a replacement algorithm (denoted by A_0) which implements the "informal principle of optimality" has been shown to be optimal (in the sense of inducing a minimum expected number of page faults) for reference strings generated by this model [4,6,7]. A second model that is often studied is called the LRU (Least Recently Used) stack model which has been found to be a better model for real program reference strings [6,8,9].

Moreover, the well-known LRU replacement algorithm has been proven optimal for reference strings generated by the LRU stack model provided that some conditions are imposed upon the distance distribution for the model. But it has not been determined whether the LRU replacement algorithm implements the "informal principle of optimality" in such cases.

To answer this question, we derive in the following the expected time until next reference for all pages occupying different LRU stack positions. Necessary and sufficient conditions on the distance distribution are found under which the LRU replacement algorithm is shown to implement the "informal principle of optimality". Furthermore, the same sufficient conditions that make the LRU replacement algorithm optimal are also shown to be sufficient for the implementation of the "informal principle of optimality".

II. The LRU stack model [6,9]

Let $N = \{1,2,\ldots,n\}$ be a set of pages and let $R = r_1 r_2 \ldots r_t \ldots r_\ell$ denote a program reference string of length ℓ, where $r_t \in N$. For all t, $1 \leq t \leq \ell$, the LRU stack $\underline{S}(t) = (s_1(t), s_2(t), \ldots s_n(t))$ is a vector, the i^{th} component of which, $s_i(t) \in N$, is the i^{th} most recently referenced page in the reference string up to and including time t [3]. Clearly, $s_1(t) = r_t$. Thus, the LRU stack is a list that orders pages in a program according to their recency of reference, with the most recently referenced page on top of the stack. Furthermore, for all time $t (1 \leq t \leq \ell)$ during a program's execution, the topmost m pages in the LRU stack with $1 \leq m \leq n$ are exactly those kept in a memory of m page frames when the LRU replacement algorithm is used.

For any LRU stact $\underline{S}(t)$, if $r_{t+1} = s_k(t)$, $\underline{S}(t+1)$ is formed as follows:

$$s_1(t+1) = s_k(t)$$

$$s_i(t+1) = s_{i-1}(t) \qquad 1 < i \leq k$$

and $$s_i(t+1) = s_i(t) \qquad k < i \leq n \qquad \text{(II.1)}$$

At the same time, an LRU stack distance at time t+1 is defined to be $d_{t+1}=k$. Note that (II.1) for updating the LRU stack characterizes actions of the LRU replacement algorithm. Hence, for any initial LRU stack $\underline{S}(0)$ and for any program reference string R, there corresponds a unique LRU distance string $D = d_1 d_2 \ldots d_t \ldots d_\ell$. On the other hand, given an initial LRU stack $\underline{S}(0)$ and any LRU distance string D, a program reference string R can be uniquely defined as follows: If $d_{t+1}=k$, $1 \leq k \leq n$, then $r_{t+1}=s_k(t)$ and $\underline{S}(t+1)$ is obtained according to (II.1).

The LRU stack model for program reference string generation assumes (1) an initial LRU stack $\underline{S}(0)$ and (2) the LRU distance string $D=d_1 d_2 \ldots d_t \ldots d_\ell$ to be a sequence of independent and identically distributed random variables such that, for all t and k with $1 \leq t \leq \ell$ and $1 \leq k \leq n$, $\Pr[d_t = k] = P_k$ and $\sum_{k=1}^{n} P_k = 1$. Consequently, any realization of the LRU distance string according to this distance distribution P_j defines a program reference string R as discussed in the last paragraph. It is obvious that, without any loss of generality, we can let $\underline{S}(0)=(1,2,\ldots,n)$. It should also be pointed out that the LRU stack model described here is the same as that presented in [6,9] but is somewhat simpler than that used in [8]. Finally, it is very important to note that the sequence of LRU stacks $(\underline{S}(0),\underline{S}(1),\ldots,\underline{S}(\ell))$ used in the generation of reference string $R=r_1 r_2 \ldots r_t \ldots r_\ell$ is precisely the same sequence of LRU stacks resulting from processing this same reference string. In the following, we shall denote program reference strings generated by this model the LRU reference strings.

III. Derivation of expected times

To compute the expected times to next reference for all pages occupying different stack positions in the LRU stack, we assume that the reference string $R=r_1 r_2 \ldots r_t \ldots r_\ell$, and hence the distance string D, is infinitely long, i.e., $\ell = \infty$.

Suppose that, for some $t \geq 0$, the LRU stack $\underline{S}(t)=(s_1(t),\ldots,s_n(t))$ is known. Let T(k) be the random variable denoting the time from time t to the time when the page $s_k(t)$ is referenced again. We compute E[T(k)], the expected value of T(k), by conditioning on the next stack distance d_{t+1}. For $1 < k < n$,

$$T(k) = \begin{cases} 1 & \text{if } d_{t+1}=k \\ 1+T'(k) & \text{if } d_{t+1} < k \\ 1+T'(k+1) & \text{if } d_{t+1} > k \end{cases} \qquad \text{(III.1)}$$

Expressions in (III.1) reflect the LRU stack updating procedure discussed in Section II and the random variables T'(k) and T'(k+1) are respectively counterparts of T(k) and T(k+1) at time t+1. Since, according to the LRU stack model for reference string generation, the distance string $D=d_1 d_2 \ldots d_t \ldots$ is a sequence of independent and identically distributed random variables, it is clear that the distance string process probabilistically restarts itself at every point in time, in particular, at time t+1. Thus it follows that E[T(k)]=E[T'(k)] and E[T(k+1)]=E[T'(k+1)]. Taking expectations, (III.1) becomes:

$$E[T(k) \mid d_{t+1}=k] = 1$$

$$E[T(k) \mid d_{t+1}<k] = 1 + E[T(k)]$$

$$E[T(k) \mid d_{t+1}>k] = 1 + E[T(k+1)].$$

Then

$$E[T(k)] = \left\{1+E[T(k)]\right\} \cdot \sum_{j=1}^{k-1} P_j + P_k + \left\{1+E[T(k+1)]\right\} \cdot \sum_{j=k+1}^{n} P_j \ ,$$

or

$$E[T(k)] = \left(\sum_{j=k}^{n} P_j\right)^{-1} \cdot \left\{E[T(k+1)] \cdot \sum_{j=k+1}^{n} P_j + 1\right\} \qquad \text{(III.2)}$$

We have also to take care of two boundary conditions: k=1 and k=n. Following similar developments, since

$$T(1) = \begin{cases} 1 & \text{if } d_{t+1} = 1 \\ 1+T'(2) & \text{if } d_{t+1} > 1 \end{cases}$$

then

$$E[T(1)] = E[T(2)] \cdot \sum_{j=2}^{n} P_j + 1 \qquad \text{(III.3)}$$

Similarly, since

$$T(n) = \begin{cases} 1 & \text{if } d_{t+1} = n \\ 1+T'(n) & \text{if } d_{t+1} < n \end{cases}$$

then

$$E[T(n)] = \frac{1}{P_n} \qquad \text{(III.4)}$$

Since (III.4) gives an explicit expression for E[T(n)], back substitution into (III.2) and (III.3) yields, for $1 \le k \le n$,

$$E[T(k)] = \frac{n-k+1}{\sum_{j=k}^{n} P_j} \qquad \text{(III.5)}$$

Thus, for LRU reference strings, (III.5) gives the expected time since time t to its next reference of a page at the k^{th} LRU stack position in terms of the distance distribution. It should be obvious that (III.5) is valid for all $t \ge 0$.

IV. Implementation of the "informal principle of optimality"

Since the organization and the updating procedure of the LRU stack completely characterize the LRU replacement algorithm, the question of whether (or under

what conditions) the LRU replacement algorithm implements the "informal principle of optimality" in processing LRU reference strings can be answered by examining (III.5). Suppose that the LRU replacement algorithm is used to manage a memory of m page frames ($m \leq n$) and that the LRU stack at time t, $\underline{S}(t)$, is known, then in order that the "informal principle of optimality" be implemented, it is necessary that the page occupying the m^{th} LRU stack position has a longer expected time to next reference than any other page occupying a smaller stack position, i.e.,

$$E[T(k)] \leq E[T(m)] \quad \text{for} \quad 1 \leq k \leq m-1 \tag{IV.1}$$

In (IV.1), it is assumed that m is fixed. But if the LRU replacement algorithm is to implement the "informal principle of optimality", (IV.1) has to hold for all values of m, $1 \leq m \leq n$. This leads us to the following requirement:

$$E[T(1)] \leq E[T(2)] \leq \ldots \leq E[T(n-1)] \leq E[T(n)] \tag{IV.2}$$

Now for LRU reference strings, let us examine E[T(k)] and E[T(k+1)] for some k, $1 \leq k < n$.

$$E[T(k)] \leq E[T(k+1)]$$

$$\Longleftrightarrow \quad \frac{n-k+1}{\sum_{j=k}^{n} P_j} \leq \frac{n-k}{\sum_{j=k+1}^{n} P_j}$$

$$\Longleftrightarrow \quad (n-k) \cdot \sum_{j=k+1}^{n} P_j + \sum_{j=k+1}^{n} P_j \leq (n-k) \cdot \sum_{j=k}^{n} P_j$$

$$\Longleftrightarrow \quad \sum_{j=k+1}^{n} P_j \leq (n-k) \cdot P_k \tag{IV.3}$$

Clearly, (IV.3) gives necessary and sufficient conditions that are to be satisfied by the probability mass distribution $\{P_j\}$ on the LRU stack distances in order for the LRU replacement algorithm to implement the "informal principle of optimality" in processing LRU reference strings. However, if

$$P_1 \geq P_2 \geq \ldots \geq P_{n-1} \geq P_n \qquad \text{(IV.4)}$$

it is clear that (IV.2) and (IV.3) will be satisfied. Thus, (IV.4) gives sufficient conditions.

It is very interesting to observe that sufficient conditions shown in (IV.4) are exactly the same sufficient conditions under which the LRU replacement algorithm is shown to be optimal (in terms of a minimum expected number of page faults) in processing LRU reference strings [9,10]. In other words, for the class of LRU reference strings satisfying $P_1 \geq P_2 \geq \ldots \geq P_n$, the LRU replacement algorithm that replaces the page with the longest expected time until its next reference (i.e., the least recently used page) is optimal.

For a fixed value of m, however, conditions in (IV.3) and (IV.4) can be relaxed. To see this, we examine (IV.1). For $1 \leq k \leq m-1$,

$$E[T(m)] \geq E[T(k)]$$

$$\Longleftrightarrow \quad \frac{n-m+1}{\sum_{j=m}^{n} P_j} \geq \frac{n-k+1}{\sum_{j=k}^{n} P_j}$$

$$\Longleftrightarrow \quad (n-m+1)\sum_{j=k}^{n} P_j \geq (n-m+1)\sum_{j=m}^{n} P_j + (m-k)\sum_{j=m}^{n} P_j$$

$$\Longleftrightarrow \quad (n-m+1)\sum_{j=k}^{m-1} P_j \geq (m-k)\sum_{j=m}^{n} P_j \qquad \text{(IV.5)}$$

(IV.5) has to be satisfied for all k, $1 \leq k \leq m-1$, and again gives necessary and sufficient conditions for implementation of the "informal principle of optimality" by the LRU replacement algorithm for a fixed value of m. A sufficient condition for (IV.5) is

$$\min_{1 \leq j \leq m-1} \{P_j\} \geq \max_{m \leq j \leq n} \{P_j\} \qquad \text{(IV.6)}$$

It is also interesting to observe that (IV.6) is slightly different from the sufficient condition required for optimality in a similar case (i.e., for a fixed value of m) [9,10].

V. Conclusion

For LRU reference strings, the expected times to next reference for all pages in the LRU stack are derived. Using these expected values, necessary and sufficient conditions as well as sufficient conditions on the distance distribution are obtained which guarantee the implementation by the LRU replacement algorithm of the "informal principle optimality". It is observed that these sufficient conditions are identical to those under which the LRU replacement algorithm is proven to be optimal. Through these exercises, the LRU replacement algorithm (when processing a restricted class of LRU reference strings) is shown to be another algorithm besides the replacement algorithm Ao which not only implements the "informal principle of optimality" but also is optimal. Of course, the only truly optimal replacement algorithm remains to be MIN whose optimality does not depend on any program model.

Acknowledgment

The authors wish to thank Professor A. J. Thomasian of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, for his helpful discussions.

References:

1. Belady, L. A. A study of replacement algorithms for a virtual-storage computer. IBM Syst. J. 5, 2(1966), 78-101.

2. Horwitz, L. P., Karp, R. M., Miller, R. E., and Winograd, S. Index register allocation. J. ACM 13, 1 (Jan. 1966), 43-61.

3. Mattson, R. L., Gesei, J., Slutz, D. R., and Traiger, I. L. Evaluation techniques for storage hierachies. IBM Syst. J. 9, 2(1970), 78-117.

4. Aho, A. V., Denning, P. J., and Ullman, J. D. Principles of optimal page replacement. J. ACM 18, 1(Jan. 1971), 80-93.

5. Coffman, E. G., and Denning, P. J. Operating Systems Theory. Prentice-Hall, Englewood Cliffs, N. J., 1973, pp. 249-250.

6. Spirn, J. R., and Denning, P. J. Experiments with program locality. Proc. AFIPS 1972 FJCC, AFIPS Press, Montvale, N. J., pp. 611-621.

7. Coffman, E. G., and Denning, P. J. Operating Systems Theory. Prentice-Hall, Englewood Cliffs, N. J., 1973, pp. 268-275.

8. Shedler, G. S., and Tung, C. Locality in page reference strings. SIAM J. Comput. 1, 3(Sept. 1972), 218-241.

9. Coffman, E. G., and Dennings, P. J. Operating Systems Theory. Prentice-Hall, Englewood Cliffs, N. J., 1973, pp. 275-278.

10. Dennings, P. J., Savage, J. E., and Spirn, J. R. Models for locality in program behavior. Dept. of Elect. Eng., Princeton Univ., Princeton, N. J., Computer Science Report TR-107 (April 1972).