A unified approach to the evaluation of a class of replacement algorithms

Erol Gelenbe

IRIA
laboria

laboratoire de recherche
en informatique
et automatique

A Unified Approach to the Evaluation of a Class of Replacement Algorithms

Erol Gelenbe

Institut de Recherche
d'Informatique
et d'Automatique

Domaine de Voluceau
Rocquencourt
78150 - Le Chesnay
Tél. 954.90.20

Rapport de Recherche n°5
décembre 1972

A UNIFIED APPROACH TO THE EVALUATION OF REPLACEMENT ALGORITHMS*

E. Gelenbe

ABSTRACT

The replacement problem arises in computer system management whenever the executable memory space available is insufficient to contain all data and code which may be accessed during the execution of an ensemble of programs. An example of this is the page replacement problem in virtual memory computers. The problem is solved by using a replacement algorithm which selects code or data items which are to be removed from executable memory whenever new items must be brought in and no more free storage space remains. An automaton theoretic model of replacement algorithms is introduced for the class of "random, partially pre-loaded" replacement algorithms, which contains certain algorithms of practical and theoretical interest. An analysis of this class is provided in order to evaluate their performance, using the assumption that the references to the items to be stored are identically distributed independent random variables. With this model, it is shown that the well-known page replacement algorithms FIFO and RAND yield the same long-run page fault rates.

* *To Appear in IEEE Transactions on Computers.*

1. Preface

The replacement problem is basic to certain situations in which a limited number of resources are multiplexed among a larger number of users. The problem may be stated as follows. Consider a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of identical resources $R = \{r_1, r_2, \ldots, r_m\}$ with $m \leq n$. At each instant of time

$$t_1, t_2, \ldots, t_k, \ldots$$

exactly one of the users requests and utilizes a resource without consuming it. At that time, one may discover that all of the resources have already been allocated, so that if our policy is to systematically satisfy each request we shall have to de-allocate a resource from one of the users. A replacement algorithm is applied at such times to select the user who is going to lose his resource, and the replacement problem is to select an algorithm based upon certain criteria and to evaluate the performance of the algorithm. Various generalizations of this problem may be imagined; one may consider several types of resources, for instance.

A typical application of this problem is to computations carried out with limited memory space. Suppose that the set of resources corresponds to a set of memory units of equal size from which instructions can be executed, and let the set of users be an ensemble of data items (all of equal size) which are stored normally on some peripheral memory unit and brought into a memory unit (one data item per memory unit) at execution time. In such a situation, the replacement algorithm is selected so as to reduce the number of times data items have to be transferred to and from the peripheral memory since this is usually a time consuming operation.

A class of replacement algorithms of interest are the random partially pre-loaded algorithms we shall study here. One may imagine that in many cases it is imperative that certain users never lose their resources, or that certain data items never be removed from their memory units. A random partially pre-loaded algorithm maintains the resources allocated to these users, and selects the user who will lose his resource at random from the remaining ones with equal probability. This class of algorithms is also of particular interest on theoretical grounds, as will be seen below.

The mathematical model we shall use to describe and evaluate this class of algorithms is a stochastic automaton [6]. These automata have appeared in the literature primarily as models of communication channels and sequential machines with random failures, as well as to represent adaptive or learning automata.

In the sequel, we shall discuss the problem using the terminology of storage allocation problems [1] both to simplify the discussion and to provide motivation for the reader. It should be stressed, however, that we consider the problem as being of broader interest.

In some virtual memory computer systems [1], a program's address space is divided into equal size blocks called pages. Similarly, primary memory space is divided into page frames each of which may contain a page of some program. At a given instant of time, not all of a program's pages need reside in memory so that when the program references a page not in memory, a page fault occurs suspending computation until the referenced page can be brought in.

Suppose that a program's set of pages is $N = (p_1, p_2, \cdots, p_n)$ and that exactly m of these can be kept in memory. Then, if $m < n$, each time a page fault occurs one of the pages currently in memory must lose its page frame to accommodate the incoming page. An algorithm which selects the page to be removed from memory when a page fault occurs is called a page replacement algorithm (PRA).

Since a high page fault rate will cause a deterioration of system performance, it is of great interest to determine the page fault rate caused by various PRA [2].

The class of random partially pre-loaded (RPPL) PRA contains both RAND 2 which selects the page to be replaced at random with equal probability among the pages in memory, and A_0 which was shown in 4 to be the "optimal" algorithm if program references are represented as a sequence of independent identically distributed random variables. RAND is used as a benchmark [2] since no PRA used in practice should have a worse performance than RAND. The class of RPPL PRA contains

$$\sum_{i=0}^{m-1} \binom{n}{i}$$

algorithms, and the evaluation presented in this paper gives us a single expression for the long run page fault rate for any algorithm in this class, based on a model of the page reference string.

Any theoretical evaluation of a PRA requires a mathematical model of the reference string. The model we shall use here is the same as the one used by King [3], namely the independent identically distributed model. This model captures the fact that programs do not refer to their pages with the same frequency; i.e. some pages are referred to more frequently than others. It does not capture the correlations which exist between successive page references or any time-varying behaviour which may exist. Thus this model is of interest over relatively short lengths of program activity; a detailed discussion of its region of validity is given by Denning and Schwartz [7]. Since replacement algorithms are of more general interest than in their applications to virtual memory machines, the independent reference model yields an evaluation which is more universal than a more realistic but more restricted representation of the reference process.

2. Introduction

In this section we shall define some concepts of use to us. Subsequent sections will contain the results of this paper.

Definition 1 A page reference string is a sequence of symbols from N, the set of pages. It represents the sequence of pages referenced by the program during execution.

Definition 2 A memory state s is an m-element subset of N. There are $\binom{n}{m}$ distinct memory states, where m is fixed and $1 \leq m \leq n$. S_m is the set of all memory states.

We will now define formally a page replacement algorithm. Before doing that, however, let us describe informally what it does. A PRA is a control mechanism which examines the page reference string and the memory state, and with this information (and other information which it may store) changes the memory state with the following constraints:

(a) If the last page referenced is in memory, the new memory state will still contain it.

(b) If the last page referenced is not in memory, the new memory state will contain it.

This notion of a PRA may be generalized to randomized algorithms in which memory and control state transititions are probabilistic, A <u>probability distribution</u> on the finite set $V = (v_1, v_2, \cdots, v_f)$ is the row vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_f)$

where

$$\alpha_i \geq 0, \; 1 \leq i \leq f, \text{ and}$$

$$\sum_{i=1}^{f} \alpha_i = 1$$

This is interpreted as follows. Let v be a random variable taking values in V. Then

$$\alpha_i = \text{Prob } [v = v_i]$$

We say that α is <u>degenerate</u> if for some j, $1 \leq j \leq f$, $\alpha_j = 1$.

<u>Definition 3.</u>

A PRA is the system

$$B = (S, Q, N, s_o, q_o, \{M(r)\})$$

where

> S is a non-empty $|S|$[1] - element subset of S_m,
>
> Q is a finite $|Q|$ - element set of control states,
>
> N is an n-element $(n \geq m)$ set of pages,
>
> s_o is the initial memory state, $s_o \in S$.
>
> q_o is the initial control state, $q_o \in Q$.

1) $|X|$ is the cardinality of the finite set X.

$\{M(r)\}$ is an n-element set of stochastic[2] matrices each of which is $|S|\cdot|Q|$ by $|S|\cdot|Q|$, and contains a matrix $M(r)$ for each $p_r \in N$.

Each pair $(s,q) \in SxQ$ is called a configuration. To each distinct configuration (s,q), we assign a distinct integer $i = g(s,q)$, $1 \leq i \leq |S|\cdot|Q|$, with the restriction that $g(s_0,q_0) = 1$. The set $\{M(r)\}$ is interpreted as follows. For any $p_r \in N$, let $m_{ij}(r)$ be the i-th row, j-th column entry of $M(r)$, where $1 \leq i \leq |S|\cdot|Q|$, $1 \leq j \leq |S|\cdot|Q|$. Then $m_{ij}(r)$ is the probability that when the program references page p_r and B is in configuration (s,q), the new configuration will be (s',q'), where $i = g(s,q)$ and $j = g(s',q')$.

Let

$$x = p_{r_1} p_{r_2} \cdots p_{r_k} p_{r_{k+1}} \cdots$$

be a page reference string. The PRA B responds to x by passing through a sequence of configurations

$$Y_1, Y_2, \cdots, Y_k, Y_{k+1}, \cdots$$

where

$$\text{Prob}\ \{g(Y_1) = j \mid (s_0,q_0),\ p_{r_1}\} = m_{1j}(r_1) \tag{1}$$

and

$$\text{Prob}\ \{g(Y_{k+1}) = j \mid g(Y_k) = i,\ p_{r_{k+1}}\} = m_{ij}(r_{k+1}) \tag{2}$$

---

2) A matrix is stochastic if each of its entries is non-negative, and the sum of the entries along each row is 1.

for any $1 \leq i, j \leq |S| \cdot |Q|$, and $k > 1$.

We say that a PRA is <u>deterministic</u> if for each $p_r \varepsilon N$, $m_{ij}(r)$ is either 0 or 1 for all $1 \leq i, j \leq |S| \cdot |Q|$.

In Definition 3, S is a subset of S_m to indicate that certain PRA (such as A_0 [2]) keep the memory state in a proper subset of S_m.

The model of program behavior under which we will evaluate a PRA is identical to the one defined by King [3] . Let $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ be a probability distribution on N, the set of pages. We shall assume that for any page reference string

$$p_{r_1} p_{r_2} \cdots p_{r_k} p_{r_{k+1}} \cdots$$

the following properties hold

(I) Prob $[r_k = i] = \beta_i$ for any $k \geq 1$ and $1 \leq i \leq n$.

(II) For any $k \neq \ell$, $\ell \geq 1$, $k \geq 1$, the event $[r_k = i]$ is independent of the event $[r_\ell = j]$ for any $1 \leq i, j \leq n$,

(III) $\beta_i \neq 0$, for each $1 \leq i \leq n$.

This simple model is known as the <u>independent reference model of program behaviour</u>. Henceforth, it will be understood that the page reference string constitutes a random process governed by rules (I), (II), and (III) above. Let

$$Y_1, Y_2, \cdots, Y_k, Y_{k+1}, \cdots$$

be the sequence of configurations the PRA B passes through in response to a page reference string. Due to property (II) of the page reference string and (1), (2) we have that

$$\text{Prob}\,\{g(Y_{k+1}) = j \mid g(Y_k) = i,\ Y_{k-1}, \ldots, Y_2, Y_1, (s_0, q_0)\}$$

$$= \text{Prob}\,\{g(Y_{k+1}) = j \mid g(Y_k) = i\}$$

so that the sequence (1) is a Markov chain [5]. The states of this chain are the configurations of B and the transition probabilities are easily obtained as follows; let c_{ij} be the probability of transition from the configuration numbered i to that numbered j, $1 \leq i,\ j \leq |S| \cdot |Q|$. Then

$$c_{ij} = \text{Prob}\,\{g(Y_{k+1}) = j \mid g(Y_k) = i\}$$

$$= \sum_{r=1}^{n} \beta_r \cdot m_{ij}(r) \tag{3}$$

and we denote by C the matrix whose i-th row, j-th column entry is c_{ij}. Evidently C is a stochastic $|S| \cdot |Q|$ by $|S| \cdot |Q|$ matrix.

<u>Definition 4.</u>

A PRA is said to be a <u>demand paging</u> algorithm if for any $i = g(s,q)$, $j = g(s',q')$ such that $m_{ij}(r) \neq 0$ and $p_r \notin s$ we have that p_r is the only element in s' which is not in s. That is, only the page which has been demanded is loaded into memory.

In a demand paging PRA, a transition from configuration (s,q) to (s',q') is called a page fault transition if $s \neq s'$.

All PRA studied in this paper are demand paging algorithms.

Now let us turn to the performance measure for a PRA which we will use in this work. This too is identical to the one used by King [3].

Again, consider the sequence of configurations

$$Y_1, Y_2, \ldots, Y_k, Y_{k+1}, \ldots, Y_w, \ldots$$

which the PRA produces in response to a page reference string. Let $Y_0 = (s_0, q_0)$ and define $f_k(s,q)$, $k \geq 1$, as follows:

$$f_k(s,q) = \begin{cases} 1, & \text{if the transition from } Y_{k-1} \text{ to } Y_k \text{ is a page} \\ & \text{fault transition, and } Y_k = (s,q), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

for any configuration (s,q). Let

$$N_w(s,q) = \sum_{k=1}^{w} f_k(s,q) \tag{5}$$

Definition 5.

The expected long-run page fault rate for the PRA B is

$$F(B) = \sum_{\substack{\text{all} \\ (s,q) \in S \times Q}} \left[\lim_{w \to \infty} E\left\{ \frac{N_w(s,q)}{w} \right\} \right]$$

if the limit exists.

For the class of algorithms studied in this paper, the limit in Definition 5 always exists.

Let C^t be the matrix obtained by multiplying C by itself t times. We say that the Markov chain is irreducible and aperiodic if there exists a natural number t_o such that each entry in C^t is non-zero for all $t \geq t_o$. This is equivalent to stating that the probability of transition from any one configuration to any other one (including itself) in t steps is non-zero for all $t \geq t_o$. A chain with this property is also called regular [5]. Regular chains have useful properties, some of which will be applied here since we will be dealing with PRA for which the chain with transition matrix C is regular. The following is a well known theorem.

Theorem 1 [5]. Let the Markov chain with transition matrix C be regular. An $|S| \cdot |Q|$ - element stochastic row vector ξ exists such that

$$\xi \circ C = \xi$$

The i-th entry of ξ, denoted by $\xi(i)$, $i = g(s,q)$, is the long run probability of finding the chain in configuration (s,q), for any $(s,q) \in SxQ$. ξ is unique.

A form of F(B) which is more convenient for our purposes is given in the following lemma whose proof can be found in Appendix 1.

Lemma 1. Let $T(i)$ be the set of integers

$$T(i) = \{ j \mid i = g(s,q),\ j = g(s',q') \text{ and the transition from } (s,q) \text{ to } (s',q') \text{ is a page fault transition } \}$$

Then if the chain with matrix C is regular, for any initial configuration

$$F(B) = \sum_{i=1}^{|S|\cdot|Q|} \xi(i) \sum_{\substack{\text{all} \\ j \in T(i)}} c_{ij}$$

for any demand paging PRA B and the independent reference model of program behaviour.

This lemma simply states that if C is the transition matrix of a regular Markov chain, then the expected long run page fault rate is merely the probability of a page fault occurring at steady state.

3. Random, Partially Pre-Loaded Algorithms

In this section we introduce the class of PRA which are the subject of our study. A theorem giving the expected long run page fault rate for this class is stated. As an application of this theorem, we obtain $F(A_0)$, and show that it is equal to the expression for A_0 derived by King [3] by other methods.

Definition 6.

For $0 \leq k \leq m-1$, let ψ_k by any k-element subset of N; ψ_0 is the empty subset. A random, partially pre-loaded (RPPL) PRA is the system B of Definition 3 with the following restrictions:

(a) $S = \{s \mid s \in S_m \text{ and } \psi_k \subseteq s\}$ so that $|S| = \binom{n-k}{m-k}$

(b) $Q = \{q_0\}$, hence $|Q| = 1$ and each $M(r)$ is $|S|$ by $|S|$.

(c) For any $s, s' \in S$; any $p_r \in N$, $i = g(s, q_0)$, $j = g(s', q_0)$,

$$m_{ij}(r) = \begin{cases} 1 & \text{if } i = j \text{ and } p_r \in s. \\ \frac{1}{m-k} & \text{if } p_r \in s',\ p_r \notin s, \text{ and } p_r \text{ is the only} \\ & \text{page in } s' \text{ which is not in } s. \\ 0 & \text{otherwise.} \end{cases}$$

Note that for each k there are $\binom{n}{k}$ distinct RPPL algorithms.

A RPPL algorithm always maintains ψ_k in memory, where ψ_k has been chosen on the basis of some decision external to the algorithm. Each time a page fault occurs the page to be replaced is chosen with equal probability among those pages in memory which are not in ψ_k. Clearly, a RPPL algorithm is a demand paging algorithm.

With the following theorem we obtain an expression for the expected long run page fault rate of any RPPL PRA.

Theorem 2. Let B be a RPPL PRA with $\psi_k = (p_1, p_2, \cdots, p_k)$ such that $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ is the probability distribution on the set $N = (p_1, p_2, \cdots, p_n)$. (No special relation among the β_i, $1 \leq i \leq n$, is implied by this choice of ψ_k). Then

$$F(B) = \frac{\sum\limits_{\substack{\text{all}\\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m} \sum\limits_{p_\ell \notin s} \beta_\ell}{\sum\limits_{\substack{\text{all}\\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}$$

where $s = (p_1, p_2, \cdots, p_k, p_{i_{k+1}}, p_{i_{k+2}}, \cdots, p_{i_m})$ if $1 \leq k \leq m-1$, and $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ if $k = 0$.

Theorem 2 is proved in Section 4. The rest of this section is devoted to an application.

The PRA A_0 [4] maintains the m-1 pages whose probability of being referenced is highest, constantly in memory. King [3] obtained $F(A_0)$; we obtain the same result here as a corollary to Theorem 2.

Definition 7.

Let the relationship $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$ hold. Then A_0 is the RPPL PRA with $k = m-1$, $\psi_{m-1} = (p_1, p_2, \cdots, p_{m-1})$ and $s_0 = (p_1, p_2, \cdots, p_{m-1}, p_m)$.

Theorem 3.

$$F(A_o) = \frac{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_m} \sum_{p_\ell \notin s} \beta_\ell}{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_m}}$$

where $S = \{s \mid s \in S_m \text{ and } \psi_{m-1} \subset s\}$, and any $s \in S$ is of the form $s = (p_1, p_2, \cdots, p_{m-1}, p_{i_m})$, $m \leq i_m \leq n$.

$F(A_o)$ in Theorem 3 may be rewritten as

$$F(A_o) = \frac{\sum_{j=m}^{n} \beta_j \sum_{\substack{i=m \\ i \neq j}}^{n} \beta_i}{\sum_{j=m}^{n} \beta_j}$$

since $m \leq i_m \leq n$, and because $p_\ell \notin s$ implies that $m \leq \ell \leq n$.

But then

$$F(A_o) = \frac{\left[\sum_{j=m}^{n} \beta_j\right]^2 - \sum_{j=m}^{n} \beta_j^2}{\sum_{j=m}^{n} \beta_j}$$

which is equal to the expression obtained by King [3].

4. Proof of Theorem 2.

Let C be the transition matrix defined by equation (3) of Section 2. The proof of Theorem 2 consists of the following parts.

(A) The entries of C will be obtained.

(B) It will be shown that C is the transition matrix of a regular Markov chain.

(C) The stochastic row vector ξ satisfying

$$\xi C = \xi$$

will be obtained.

(D) F(RPPL) will be derived using Lemma 1.

PART (A) By Definition 6, C is $|S|$ by $|S|$. Let $u = g(s, q_0)$, $v = g(s', q_0)$; $s, s' \in S$. Then by (3) and (c) of Definition 6 we have

$$c_{uu} = \sum_{r=1}^{n} \beta_r \, m_{uu}(r) \tag{6}$$

$$= \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r$$

$$c_{uv} = \frac{\beta_r}{m-k}, \quad \text{if } p_r \notin s,\ p_r \in s', \text{ and } p_r \text{ is the only page in } s' \text{ which is not in } s. \tag{7}$$

$$c_{uv} = 0, \quad \text{in all cases not covered by (6) and (7).} \tag{8}$$

PART (B) To show that C is regular, it must be shown that the probability of reaching any configuration $w = g(s'', q_0)$ from any $u = g(s, q_0)$ in t steps is non-zero for any $t \geq t_0$, and t_0 fixed. This is easily proved. Let s'' contain y pages not in s; let these pages be

$$p_{z_1}, p_{z_2}, \cdots, p_{z_y}$$

Evidently $y \leq m-k$. Then $w = g(s'', q_0)$ is reached from $u = g(s, q_0)$ with no more than m-k transitions of the type whose probability is given in (7). Since $\beta_i \neq 0$, $1 \leq i \leq n$, by property (III) of the independent reference model of program behaviour it follows that the probability of this y step transition is non-zero since it can be no smaller than

$$\frac{\beta_{z_1} \cdot \beta_{z_2} \cdots \beta_{z_y}}{(m-k)^y} \tag{9}$$

The probability of reaching w from u in more than y steps is also non-zero since an arbitrary finite number transitions of the type whose probability is given in (6) can be used as well. Therefore C is regular and $t_0 \leq m-k$.

PART (C). For any $s \in S$, recall that without loss of generality we write

$$s = (p_1, p_2, \cdots, p_k, p_{i_{k+1}}, p_{i_{k+2}}, \cdots, p_{i_m}) \tag{10}$$

if $k \geq 1$ and $\psi_k = \{p_1, p_2, \cdots, p_k\}$, and

$$s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m}) \tag{11}$$

if $k = 0$ (since ψ_0 is empty). Further, note that for $0 \leq k \leq m-1$,

$$p_{i_{k+j}} \notin \psi_k \tag{12}$$

for any $1 \leq j \leq m-k$. Let (s, q_0), (s', q_0) be two configurations of the RPPL algorithms so that $u = g(s, q_0)$ and $v = (s', q_0)$. By Theorem 1 and PART (B) of this proof , there exists an $|S|$-dimensional row vector ξ whose u-th entry $\xi(u)$, $1 \leq u \leq |S|$, is the long run probability of finding the Markov chain C whose entries are given by (6), (7), (8), in configuration (s, q_0) where $u = g(s, q_0)$. Furthermore, ξ satisfies

$$\xi \cdot C = \xi \tag{13}$$

and

$$\sum_{u=1}^{|S|} \xi(u) = 1 \tag{14}$$

Equation (13) may be written as $|S|$ equations of the form (for each $1 \leq u \leq |S|$, $u = g(s, q_0)$, $s \in S$)

$$\xi(u) = \sum_{v=1}^{|S|} \xi(v)\, c_{vu}$$

which, using (6), (7) , (8) becomes

$$\xi(u) = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \sum_{\substack{\text{all} \\ v \in R(u,j)}} \xi(v) \cdot \frac{\beta_{i_{k+j}}}{m-k} \qquad (15)$$

where $R(u, j)$ is the set of all $v = g(s', q_0)$ such that if s' is the memory state when the program references page $p_{i_{k+j}}$, for some j, $1 \leq j \leq m-k$, then a page fault will occur and the new memory state will be s with probability (from (7))

$$c_{vu} = \frac{\beta_{i_{k+j}}}{m-k} \qquad (16)$$

Thus, each s' may be expressed as

$$s' = s \cup \{p_a\} - \{p_{i_{k+j}}\} \qquad (17)$$

where $p_a \notin s$, if $v = g(s', q_0) \in R(u, j)$.

We shall show that for each u, $1 \leq u \leq |S|$,

$$\xi(u) = \frac{\beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}} \qquad (18)$$

satisfies (15), where $u = g(s, q_0)$ and s is given in (10).

By (17), (18):

$$\xi(v) = \frac{\beta_a}{\beta_{i_{k+j}}} \xi(u) \qquad (19)$$

Let the right-hand side of (15) be called Q. To show that $\xi(u)$ in (18) satisifies (15), it suffices to prove that by substituting (19) in Q we still obtain

$$Q = \xi(u)$$

Substituting (19) in Q we get that

$$Q = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \sum_{\substack{\text{all} \\ v \in R(u,j)}} \beta_a \frac{\xi(u)}{m-k}$$

But, for each $p_a \notin s$, $R(u,j)$ contains exactly one v and $|R(u,j)| = m-n$, independently of j. Therefore

$$Q = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \frac{\xi(u)}{m-k} \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r \tag{20}$$

But $\xi(u)$ (in (18)) is independent of j, therefore

$$Q = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \xi(u) \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r \tag{21}$$

$$= \xi(u)$$

so that the proof that $\xi(u)$ in (18) satisfies (15) and (13) is complete. The vector ξ defined by (18) is stochastic since its entries sum to unity. This completes Part (C) of the proof.

PART (D). By Lemma 1, and PART (B) of the proof, for any RPPL PRA, independently of the initial configuration,

$$F(\text{RPPL}) = \sum_{u=1}^{|S|} \xi(u) \sum_{\substack{\text{all} \\ v \in T(u)}} c_{uv} \tag{22}$$

Let $u = g(s, q_0)$ and let $p_r \notin s$. If the RPPL PRA is in configuration (s, q_0) when the program references page p_r, the PRA may enter any one of $(m-k)$ configurations of the form $(s', q_0), v = g(s', q_0)$, where $s' = s \cup \{p_r\} - \{p_{i_{k+j}}\}$ for any $1 \leq j \leq m-k$, with probability (see (7))

$$\frac{\beta_r}{m-k}$$

Evidently, each such v is in $T(u)$. Since this is true for each $p_r \notin s$, we have

$$\sum_{\substack{\text{all} \\ v \in T(u)}} c_{uv} = \sum_{\substack{\text{all} \\ p_r \notin s}} (m-k) \cdot \frac{\beta_r}{m-k}$$

$$= \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r \tag{23}$$

Turning to equation (22), we see that it may be rewritten as

$$F\ (RPPL) = \sum_{\substack{\text{all}\\ s \in S}} \xi(u) \sum_{\substack{\text{all}\\ p_r \notin s}} \beta_r \tag{24}$$

using (23) and the fact that for each $1 \leq u \leq |S|$ there is a unique $s \in S$ such that $u = g(s, q_0)$, and vice-versa. When (20) is used in (24) we obtain

$$F\ (RPPL) = \frac{\displaystyle\sum_{\substack{\text{all}\\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m} \sum_{\substack{\text{all}\\ p_r \notin s}} \beta_r}{\displaystyle\sum_{\substack{\text{all}\\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}$$

which completes the proof of Theorem 2.

5. The Algorithms RAND and FIFO

It is the purpose of this section to define and examine the algorithm RAND and to show that the expected long run page fault rate of RAND and of the well known PRA FIFO [3] are equal for the independent reference model of program behaviour.

RAND is, in some sense, the most trivial PRA. Whenever a page fault occurs, RAND chooses the page to be replaced at random among the pages in memory so that the probability of being removed is equal for all pages in memory. Thus any PRA being used should have a page fault rate which is at least no worse than RAND's, and RAND is useful as a basis for comparison.

FIFO (First-In-First-Out), on the other hand, always replaces the page which was first to enter memory among the pages currently in memory. FIFO is briefly discussed in [2]. King [3] has obtained F(FIFO) under the independent reference model of program behaviour.

Definition 8.

RAND is the RPPL PRA with $k = 0$ (so that ψ_0 is the empty subset of N, and $S = S_m$).

As an immediate consequence of Theorem 2 and Definition 8 we have

Theorem 4. The expected long run page fault rate of RAND is

$$F(\mathrm{RAND}) = \frac{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m} \sum\limits_{\substack{\text{all}\\ p_r\notin s}} \beta_r}{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}}$$

where $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ for any $s \in S$.

From King's paper [3] we have an expression for F(FIFO) which we will presently show to be identical to F(RAND). But first let us describe the PRA FIFO. Informally, FIFO keeps a marker on that page which was first to enter memory among all pages currently in memory. When a page fault occurs, the marked page is removed from memory and the marker is updated. Thus, FIFO is a deterministic PRA with $S = S_m$; whenever $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ is the memory state, the control state is $q = (p_{j_1}, p_{j_2}, \cdots p_{j_m})$ where $(j_1, j_2, \cdots j_m)$ is a permutation of $(i_1, i_2, \cdots, i_m)$ so that $p_{j_1}, p_{j_2}, \cdots, p_{j_m}$ is the ordering of the members of s from left to right in order of first entry into memory (i.e. p_{j_1} is the First-In, etc.) If now the program references $p_a \notin s$ causing a page fault, the new memory state is

$$s' = s \cup \{p_a\} - \{p_{j_1}\}$$

and the new control state is

$$q' = (p_{j_2}, p_{j_3}, \cdots, p_{j_m}, p_a)$$

Thus, for each memory state there are m! possible control states. For a formal treatment of FIFO the reader is referred to [3] .

Theorem 5. (King [3]) The expected long run page fault rate of FIFO under the independent reference model of program behaviour is

$$F(\text{FIFO}) = \frac{\sum\limits_{\substack{\text{all} \\ q \in Q}} \beta_{j_1} \beta_{j_2} \cdots \beta_{j_m} (1 - \beta_{j_1} - \beta_{j_2} - \cdots - \beta_{j_m})}{\sum\limits_{\substack{\text{all} \\ q \in Q}} \beta_{j_1} \beta_{j_2} \cdots \beta_{j_m}}$$

where

$$q = (p_{j_1}, p_{j_2}, \cdots, p_{j_m}).$$

When the control state is q of Theorem 4, the memory state is $s = (p_{i_1}, p_{i_2}, \cdots p_{i_m})$ where $(j_1, j_2, \cdots, j_m)$ is a permutation of $(i_1, i_2, \cdots, i_m)$ so that

$$\sum_{\substack{\text{all}\\ q\in Q}} \beta_{j_1}\beta_{j_2}\cdots\beta_{j_m} = (m!) \sum_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}$$

Furthermore

$$\sum_{\substack{\text{all}\\ p_r\notin s}} \beta_r = (1-\beta_{j_1}-\beta_{j_2}-\cdots-\beta_{j_m})$$

so that

$$F(\text{FIFO}) = \frac{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m} \sum\limits_{\substack{\text{all}\\ p_r\notin s}} \beta_r}{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}}$$

and since $S = S_m$ for both FIFO and RAND, we have

<u>Theorem 6.</u> $\qquad F(\text{FIFO}) = F(\text{RAND})$

Appendix 1

Proof of Lemma 1

It is to be shown that if the Markov chain with transition matrix C, given by (3), is regular for the PRA B under the independent reference model of program behaviour, then the expected long run page fault rate F(B) defined in Definition 5 is given by:

$$F(B) = \sum_{i=1}^{|S|\cdot|Q|} \xi(i) \sum_{\substack{\text{all} \\ j \in T(i)}} c_{ij} \tag{1.1}$$

where $\xi(i)$ is the i-th entry of the vector ξ of Theorem 1, and c_{ij} is an entry of the matrix C defined by (3).

Let $u = g(s,q)$, $v = g(s',q')$, for any two configurations (s,q), (s',q'). Let

$$Y_1, Y_2, \cdots, Y_k, \cdots$$

be the sequence of configurations the PRA B passes through in response to a page reference string; B is started in $Y_0 = (s_0, q_0)$. Define the function

$$h_k(u) = \begin{cases} 1 & \text{if } g(Y_k) = u, \\ 0 & \text{otherwise,} \end{cases} \tag{1.2}$$

for all $k \geq 0$, and

$$M_w(u) = \sum_{k=0}^{w-1} h_k(u) \qquad (1.3)$$

We will need the following theorem known as the law of large numbers for regular Markov chains.

The Law of Large Numbers (Theorem 4.2.1 of [5])

Let C be the transition matrix of a regular chain with stochastic vector ξ satisfying

$$\xi C = \xi$$

Then, for any initial state (in this case "configuration") it follows that

$$\lim_{w \to \infty} E\left\{\frac{M_w(u)}{w}\right\} = \xi(u)$$

and for any $\varepsilon > 0$

$$\lim_{w \to \infty} \operatorname{Prob}\left\{\left|\frac{M_w(u)}{w} - \xi(u)\right| > \varepsilon\right\} = 0$$

For any $v = g(s', q')$, $k \geq 1$, let

$$X_k(u, v) = \begin{cases} 1 & \text{if } g(Y_{k-1}) = u \text{ and } g(Y_k) = v, \\ 0, & \text{otherwise} \end{cases} \tag{1.4}$$

and

$$e_w(u, v) = E\left\{\frac{\sum_{k=1}^{w} X_k(u, v)}{w}\right\}$$

$$= \frac{1}{w} \sum_{k=1}^{w} E\left\{X_k(u, v)\right\} \tag{1.5}$$

But quite simply, by (1.4),

$$E\{X_k(u, v)\} = \text{Prob}\{X_k(u, v) = 1\} \tag{1.6}$$

and by Bayes' rule and (1.2)

$$\text{Prob}\{X_k(u, v) = 1\} = \text{Prob}\{v = g(Y_k) \mid u = g(Y_{k-1})\}$$

$$\cdot \text{Prob}\{h_{k-1}(u) = 1\}$$

But since

$$Y_1, Y_2, \cdots Y_k, \cdots$$

is a Markov chain, we have that

$$c_{uv} = \text{Prob}\,\{v = g(Y_k) \mid u = g(Y_{k-1})\}$$

independently of k, so that by (1. 5) and (1. 6) we obtain

$$e_w(u, v) = \frac{1}{w} \sum_{k=1}^{w} c_{uv}\ \text{Prob}\,\{h_{k-1}(u) = 1\} \tag{1. 7}$$

But then

$$e_w(u, v) = \frac{1}{w}\, c_{uv} \sum_{k=0}^{w-1} \text{Prob}\,\{h_k(u) = 1\}$$

and by (1. 2)

$$\text{Prob}\,\{h_k(u) = 1\} = E\,\{h_k(u)\}$$

therefore using (1. 3) we obtain

$$e_w(u, v) = c_{uv}\ E\left\{\frac{M_w(u)}{w}\right\}$$

so that by the Law of Large Numbers,

$$\lim_{w\to\infty} e_w(u, v) = \xi(u)\, c_{uv} \tag{1. 8}$$

for any u, v. In particular, this is true if the transition from (s, q) to (s', q') is a page fault transition, $u = g(s, q)$, $v = g(s', q')$. With reference to (4) of Section 2 and (1.4), notice that $f_k(s', q') = 1$ if and only if $X_k(u, v) = 1$ for any (s, q) such that the transition from (s, q) to (s', q') is a page fault transition. Therefore from (5):

$$N_w(s', q') = \sum_{k=1}^{w} \sum_{\substack{\text{all } u \\ \text{such that} \\ v \in T(u)}} X_k(u, v) \tag{1.9}$$

so that by (1.5)

$$E\left\{\frac{N_w(s', q')}{w}\right\} = \sum_{\substack{\text{all } u \\ \text{such that} \\ v \in T(u)}} e_w(u, v) \tag{1.10}$$

By (1.8) and (1.10)

$$\lim_{w \to \infty} E\left\{\frac{N_w(s', q')}{w}\right\} = \sum_{\substack{\text{all } u \\ \text{such that} \\ v \in T(u)}} \xi(u)\ c_{uv} \tag{1.11}$$

Finally from (1.11) and Definition 5 we obtain

$$F(B) = \sum_{\substack{\text{all} \\ (s', q') \epsilon SxQ}} \lim_{w \to \infty} E\left\{\frac{N_w(s', q')}{w}\right\}$$

$$= \sum_{v=1}^{|S| \cdot |Q|} \sum_{\substack{\text{all } u \\ \text{such that} \\ v \in T(u)}} \xi(u)\ c_{uv}$$

The two summations in the above expression are used to sum $\xi(u)c_{uv}$

over all pairs (u, v) such that $v \in T(u)$. Therefore we may rewrite this as

$$F(B) = \sum_{u=1}^{|S| \cdot |Q|} \sum_{\substack{\text{all} \\ v \in T(u)}} \xi(u)\, c_{uv}$$

completing the proof.

REFERENCES

[1] B. Randell and C. J. Kuehner, "Dynamic storage allocation systems", Commun. Ass. Comput. Mach., Vol. 11, No. 5, May 1968, pp. 297-305.

[2] L. A. Belady, "A study of replacement algorithms for a virtual storage computer", IBM Systems Journal, Vol. 5, No. 2, July 1966, pp. 78-101.

[3] W. F. King III, "Analysis of paging algorithms", IBM Research Report RC 3288, March 1971.

[4] A. V. Aho, P. Denning, and J. D. Ullman, "Principles of optimal page replacement", Journal Ass. Comput. Mach., Vol. 18, No. 1, Jan. 1971, pp. 80-93.

[5] J. G. Kemeny and J. L. Snell, Finite Markov Chains, Van Nostrand, Princeton, N. J., 1960.

[6] T. L. Booth, Sequential Machines and Automata Theory, New York: Wiley, 1967.

[7] P. J. Denning and S. C. Schwartz, "Properties of the working set model", in ACM Third Symposium on Operating Systems Principles, pp. 130-140, Oct. 1971.