On optimal performance in self-organizing paging algorithms

By R. I. PHELPS* and L. C. THOMAS**

Introduction

A large body of literature has grown up concerned with paging algorithms [e.g. 1, 3, 4, 6, 10]. In the terminology of Arato [2] a program's address space is divided into equal size blocks called page. There are two levels of memory: the first level is a fast access device and the second level is a larger, slower backing store. These levels are each divided into page frames. If a program references a page in the second level a page fault is said to occur. In this case the referenced page is brought into the first level and to make room for it another page is selected by the paging algorithm to be removed from the first to the second level.

The objective of the algorithm is to minimize the expected number of page faults. Most authors have considered this problem when either the probability distribution of the string of program references is known or when the algorithm stores information on the number of times each page has been referenced in order to estimate the distribution [1, 2, 3, 4, 10].

Interest has recently been shown in a different type of paging algorithm, when the distribution is unknown and no information on page references is collected [6, 7]. The page references are assumed to form an independent sequence. These are self-organizing algorithms. Since no information about the reference probabilities is available they can only select the page to be removed from first level memory on the basis of its position in the memory. The advantages of this approach are that no prior knowledge of reference probabilities is required, and no memory needs to be used in collecting information on these probabilities changing, the algorithm will automatically adjust to the new distribution.

The problem of finding the optimal self-organizing paging algorithm is related to another problem that has recently received much attention: there is a linear array of n storage positions $1, \ldots, n$, containing n pages $I_1, I_2, \ldots, I_n$, together with a self-organizing algorithm τ. τ acts so that if the page referenced was in position j it is moved to position $\tau(j)$. If $\tau(j)<j$ then the pages in positions $j-1$ to $\tau(j)$ are all moved back one position each to make room for the page moved from j to $\tau(j)$. If $\tau(j)>j$, then the pages in positions $j+1$ to $\tau(j)$ all move forward one position each. No other pages are moved.

The objective of the algorithm is to minimize the cost, which is the asymptotic expected position of the next page referenced. This is known as the library problem.

In this context the algorithm $\tau(j)=1,\ 1\leqq j\leqq n,$ known as the "move to the front" algorithm, has been studied by several authors [5, 8, 11, 12, 13]. McCabe [13] found an expression for the cost and for its variance. Hendricks [8] gives the stationary distribution of arrangements of pages in the store and Burville and Kingman [5] show that the cost is less than $2m-1$, where m is the cost if the reference distribution is known and the pages arranged optimally. Letac [12] considers the extension of this system in an infinite set of storage positions. Hendricks [9] gives stationary distributions for the algorithms $\tau(j)=k,\ 1\leqq j\leqq n.$

McCabe [13] suggested that the algorithm $\tau(j)=j-1,\ 2\leqq j\leqq n,\ \tau(1)=1,$ the 'transposition' algorithm, would have a cost less than that of the move to the front algorithm. Rivest [14] gives the stationary distribution of the transposition algorithm and shows that the cost of the transposition is always less than or equal to the cost of move to the front. He suggested that transposition is optimal for all reference distributions and supports this with simulation results. Thomas [15] proves that if such an optimal algorithm exists, it must be the transposition.

Returning to the specific paging algorithm setting with two levels of memory, we take the first level to have a capacity of M pages and the second level to have capacity $n-M$ pages. The page frames in the first level are numbered 1 to M and those in the second level $M+1$ to n. We can only consider paging algorithms of form $\tau(i)\leqq M,\ i>M.$

To minimize the number of page faults we take the following cost structure. A cost of 1 is incurred whenever a page in any of the positions $M+1$ to n is referenced. The cost is 0 otherwise. From among the possible paging algorithms, the closest to the transposition algorithm is the paging algorithm 'CLIMB' for which

$$\tau(j)=\begin{cases} M, & j>M \\ j-1, & 2\leqq j\leqq M \\ 1, & j=1 \end{cases}$$

Franaszek and Wagner [6] suggest that CLIMB is the optimal self-organizing paging algorithm for all reference distributions.

The purpose of this paper is to provide supporting analytical evidence for these suggestions by showing that in the special case of any reference distribution of form $p_1=ka, p_2=\ldots=p_n=a,$ where p_i is the reference probability of page I_i, transposition and CLIMB are indeed optimal for their respective cost functions.

1. Optimality for the special case of library problem

The intuitive justification for self-organizing algorithms is that when a page is referenced its posterior reference probability will increase. Since it is clear that it is optimal to have the pages with higher probability in the first positions, it follows that when a page has been referenced it should be moved forward. In this way the pages which are referenced most will tend to be moved into the first positions.

Because of this we restrict attention to algorithms for which $\tau(j)\leqq j,\ 1\leqq j\leqq n,$ and initially we will consider only 'forward moving' algorithms, that is $\tau(j)<j,$ $1\leqq j\leqq n.$ (The results can easily be extended to $\tau(j)\leqq j,$ as will be indicated later.)

With the reference distribution $p_1=ka, p_2=\ldots=p_n=a$, pages $I_2, \ldots, I_n$ have the same probability of reference and hence are equivalent as far as the self-organising system is concerned, and it only has n different states, depending on the position of page I_1. Any algorithm τ then gives rise to a Markov chain with n states, state i corresponding to I_1 being in the i^{th} position. Let P^τ be the one step transition matrix for this algorithm, whose elements p_{ij}^τ are the probabilities that referencing one page and applying τ changes the system from state i to state j. The limiting "steady-state" probabilities for each state under τ are $\underline{\pi}^\tau=(\pi_1^\tau, \pi_2^\tau, \ldots, \pi_n^\tau)$ where $\underline{\pi}^\tau=\underline{\pi}^\tau P^\tau$, provided this exists. Thus the average expected position of the next page referenced is

$$a \sum_{i=1}^{n} \pi_i^\tau \left[\frac{1}{2} n(n+1)+i(\alpha-1)\right].$$

We will show that the transposition algorithm, which we denote as T, where $T(1)=1$, $T(j)=j-1$, $2 \leqq j \leqq n$, minimises this expected position among all algorithms τ, where $\tau(j)<j$, $j \neq 1$ and $\tau(1)=1$.

For any algorithm τ of $n+1$ positions, define an algorithm $D\tau$ on n positions by

$$D\tau(i) = \begin{cases} \tau(i+1)-1 & \text{if } \tau(i+1) \neq 1, \\ 1 & \text{if } \tau(i+1) = 1. \end{cases}$$

As an example, suppose τ is defined on four positions with $\tau(1)=1$, $\tau(2)=1$, $\tau(3)=1$, $\tau(4)=3$. If the probability of referencing I_1 is ka and of referencing the others is a, then

$$P^\tau = \begin{pmatrix} ka+a & 2a & 0 & 0 \\ ka & 2a & a & 0 \\ ka & 0 & 2a & a \\ 0 & 0 & ka & 3a \end{pmatrix}.$$

The corresponding algorithm $D\tau$ on 3 positions satisfied $D\tau(1)=1$, $D\tau(2)=1$, $D\tau(3)=2$, and if the probabilities of referencing I_1, I_2, I_3 are ka', a', a' respectively, then

$$P^{D\tau} = \begin{pmatrix} ka'+a' & a' & 0 \\ ka' & a' & a' \\ 0 & ka' & 2a' \end{pmatrix}.$$

It can be shown (Appendix 1) that

$$\pi_i^\tau = (1-\pi_1^\tau)\pi_{i-1}^{D\tau}, \quad i = 2, \ldots, n+1, \ldots \tag{1}$$

Lemma 1. For any forward moving algorithm τ on n positions with $p_1=ka$, $p_2=p_3=\ldots=p_n=a$,

$$\text{(i)} \quad k \leqq \frac{\pi_i^\tau}{\pi_n^\tau} \leqq k^{n-i} \quad \text{if} \quad k \geqq 1, \tag{2}$$

$$\text{(ii)} \quad k \geqq \frac{\pi_i^\tau}{\pi_n^\tau} \geqq k^{n-i} \quad \text{if} \quad k \leqq 1. \tag{3}$$

Proof. Appendix.

Lemma 2. With the conditions of lemma 1, $\pi_1^T \geqq \pi_i^\tau$, $1 \leqq i \leqq n$.

Proof. Appendix.

Theorem 1. For any reference distribution on pages $I_1, \ldots, I_n$ of the form $p_1 = ka$, $p_2 = \ldots = p_n = a$, then amongst all algorithms τ of the form $\tau(1) = 1$, $\tau(i) < i$, the transposition algorithm minimises the expected position of the next page referenced.

Proof. We proceed by induction on the number of positions. The result is true if there are only two positions, as T is the only forward moving algorithm. So suppose the theorem is true for n-positions and consider a forward moving algorithm on $n+1$ positions. The expected position under algorithm τ is

$$\frac{1}{2}(n+1)(n+2)a + a(k-1)\sum_{i=1}^{n+1} i\pi_i^\tau.$$

Thus we want to show

$$\sum_{i=1}^{n+1} i\pi_i^T \leqq \sum_{i=1}^{n+1} i\pi_i^\tau \quad \text{if} \quad k \geqq 1$$

and

$$\sum_{i=1}^{n+1} i\pi_i^T \geqq \sum_{i=1}^{n+1} i\pi_i^\tau \quad \text{if} \quad k \leqq 1$$

for any algorithm τ. Using (1) we have

$$\sum_{i=1}^{n+1} i\pi_i^\tau = \pi_1^\tau + \sum_{i=2}^{n+1} i(1-\pi_1^\tau)\pi_{i-1}^{D\tau} = \pi_1^\tau + (1-\pi_1^\tau)\sum_{i=1}^{n} \pi_i^{D\tau} + (1-\pi_1^\tau)\sum_{i=1}^{n} i\pi_i^{D\tau}$$

$$= 1 + (1-\pi_1)\sum_{i=1}^{n} i\pi_i^{D\tau}. \tag{4}$$

Assume $k \geqq 1$, then by the induction hypothesis $\sum_{i=1}^{n} i\pi_i^{DT} \leqq \sum_{i=1}^{n} i\pi_i^{D\tau}$, since DT is in fact transportation on $n-1$ items. Also from lemma 2, $\pi_1^T \geqq \pi_1^\tau$. Hence the induction is completed.

2. Optimal paging algorithm for the special case

We can now use these results to prove the optimality of CLIMB as a paging algorithm. We consider a 2-level memory with M page frames in level 1 and $n-M$ page frames in level 2. We denote the algorithm CLIMB by C.

Lemma 3. $\pi_1^C \geqq \pi_i^\tau$, $1 \leqq i \leqq n$, for all τ of form $\tau(1) = 1$, $\tau(i) < i$, $2 \leqq i \leqq M$; $\pi(i) \leqq M$, $i > M$; $M < n$.

Proof. Appendix 4.

Theorem 2. For any reference distribution $p_1=ka$, $p_2=\ldots=p_n=a$, with M page frames in first-level memory, $M<n$, the algorithm CLIMB asymptotically minimizes the number of paging faults among all algorithms of form $\tau(1)=1$, $\tau(i)<i$, $2\leqq i\leqq M$; $\tau(i)\leqq M$, $i>M$.

Proof. We use induction on n. If $n=2$, then independent of whether $M=1$ or 2, CLIMB is the only policy in the class we are dealing with. Moreover, for any n, if $M=1$, then again CLIMB is the only possible policy. Assume the results holds for n positions, and look at the case with $n+1$ positions and first level memory of $M+1$ items, $M>1$. For any algorithm τ in this class, $D\tau$ is an algorithm on n items with a first level memory of M items. The expected paging cost using algorithm τ is

$$\sum_{i=1}^{M+1}\pi_i^{\tau}(n-M)a+\sum_{i=M+2}^{n+1}\pi_i^{\tau}(n-M+k-1)a=(n-M)a+(k-1)a\sum_{i=M+2}^{n+1}\pi_i^{\tau}.$$

Hence we need to show that

$$\sum_{i=M+2}^{n+1}\tau_i^C\leqq\sum_{i=M+2}^{n+1}\pi_i^{\tau}.$$

By (1)

$$\sum_{i=M+2}^{n+1}\pi_i^{\tau}=(1-\pi_1^{\tau})\sum_{i=M+1}^{n}\pi_i^{D\tau}.$$

The induction hypothesis implies that $\sum_{i=M+1}^{n}\pi_i^{DC}\leqq\sum_{i=M+1}^{n}\pi_i^{D\tau}$, and lemma 3 gives $\pi_1^C\geqq\pi_1^{\tau}$, so $\sum_{i=M+1}^{n+1}\pi_i^C\leqq\sum_{i=M+2}^{n+1}\pi_i^{\tau}$.

Other algorithms. Two other paging algorithms which can be run in a self organizing manner are [6] namely:

Least Recently Used — where the page which is moved from first-level memory is the one least recently referenced. This corresponds to an algorithm $\tau(j)=1$, $1\leqq j\leqq n$, which is forward moving and hence inferior to CLIMB at least for this reference distribution.

First In, First Out — where the one to leave is the one which arrived first, of those presently in the first level. This is an algorithm where

$$\tau(i)=1,\quad i=1,\ldots,M,\quad \tau(j)=1,\quad j>M.$$

This algorithm does not seem to be covered by the theorem. However, all the above results can be extended to include the cases where $\tau(j)\leqq j$. The difficulty is that one can then have algorithms which give rise to a Markov chain in which it is not possible to get from some states to others. Thus the 'steady state' probabilities depend on the initial ordering of the pages. However, if we assume that the initial ordering is equally likely to be any of the possible orderings, the above results still hold. This is because any set of connected states corresponds to the page with reference probability ka being in a set of consecutive positions, and an overall transposition algorithm is better than one which is a transposition algorithm on

each set individually. Thus Theorems 1 and 2 can be extended to allow algorithms where $\tau(j) \leqq j$.

APPENDIX 1. Proof of $\pi_i^\tau = (1-\pi_1^\tau)\pi_{i-1}^{D\tau}$, $i=2, \ldots, n+1$.

For a forward moving τ, let $S(i|\tau) = |\{r | r > i, \tau(r) \leqq i\}|$, where $|A|$ denotes the cardinality of the set A. Let

$$T(i|\tau) = \{r | r > i, \tau(r) = i\}.$$

The $i+1^{\text{th}}$ component of the equation $\underline{\pi}^\tau = \underline{\pi}^\tau P^\tau$ then reads

$$\pi_{i+1}^\tau = \Big(ka \sum_{r \in T(i+1|\tau)} \pi_r^\tau\Big) + \pi_{i+1}^\tau\big(1-ka-aS(i+1|\tau)\big) + \pi_i^\tau aS(i|\tau), \quad i \geqq 1. \tag{1.1}$$

This follows because as τ is forward moving a page will only move if it is referenced or if the referenced page moves from behind it to in front of it. Thus I_1 is in the $i+1^{\text{th}}$ position, because either it was in the r^{th} position previously and was referenced, where $\tau(r)=i+1$, or it was in the $i+1^{\text{th}}$ position and the page referenced did not move it, or was in the i^{th} position and the page referenced moved in front of it. Thus 1.1 becomes

$$S(i|\tau)\pi_i^\tau + \pi_{i+1}^\tau\big(k+S(i+1|\tau)\big) - k \sum_{r \in T(i+1|\tau)} \pi_r^\tau. \tag{1.2}$$

Applying the same procedure to the i^{th} component of $\underline{\pi}^D = \underline{\pi}^{D\tau} P^{D\tau}$, when the probabilities of referencing the pages are $(ka', a', \ldots, a')$ gives

$$S(i-1|D\tau)\pi_{i-1}^{D\tau} = \pi_i^{D\tau}\big(k+S(i|D\tau)\big) - k \sum_{r \in T(i|D\tau)} \pi_r^{D\tau}, \quad i \geqq 2. \tag{1.3}$$

By the definition of D it is obvious that

$$T(i|D\tau) = \{j-1 | j \in T(i+1|\tau)\}, \quad S(i|D\tau) = S(i+1|\tau), \quad i = 2, \ldots, n. \tag{1.4}$$

Thus if we identified $\pi_{i-1}^{D\tau}$ with π_i^τ, equations 1.2 and 1.3 would be the same.

Consider 1.2 for the case $i=n$. The right hand side of the equation can only contain terms in π_{n+1}^τ since τ is forward moving and so $T(n+1|\tau)$ is empty, while $S(n|\tau)$ is 1. We thus have a linear equation $\pi_n^\tau = K_n \pi_{n+1}^\tau$ for some constant K_n. Since 1.3 is identical with 1.2 except that $\pi_{i-1}^{D\tau}$ replaces π_i^τ throughout, in the corresponding case it becomes $\pi_{n-1}^{D\tau} = K_n \pi_n^{D\tau}$.

Now consider 1.2 with $i=n-1$. The right hand side can now only contain terms in π_n^τ and π_{n+1}^τ. We can substitute $K_n \pi_{n+1}$ for π_n^τ and so obtain another linear equation $\pi_{n-1}^\tau = K_{n-1}\pi_{n+1}^\tau$, for some K_{n-1}. The same argument implies that 1.3 will give $\pi_{n-2}^{D\tau} = K_{n-1}\pi_n^{D\tau}$. Repeating the procedure gives

$$\frac{\pi_{i-1}^{D\tau}}{\pi_n^{D\tau}} = \frac{\pi_i^\tau}{\pi_{n+1}^\tau} = K_i, \quad i = 2, \ldots, n. \tag{1.5}$$

Thus

$$\sum_{i=1}^{n} \pi_i^{D\tau} = \Big(\sum_{i=2}^{n} K_i + 1\Big)\pi_n^{D\tau} = 1$$

and

$$\sum_{i=2}^{n+1} \pi_i^\tau = \Big(\sum_{i=2}^{n} K_i + 1\Big)\pi_{n+1}^\tau = 1 - \pi_1^\tau$$

so $\pi^{\tau}_{n+1}=(1-\pi^{\tau}_1)\pi^{D\tau}_n$ and hence

$$\pi^{\tau}_i = (1-\pi^{\tau}_1)\pi^{D}_{i-1}, \quad i = 2, \ldots, n+1. \tag{1.6}$$

APPENDIX 2. Proof of Lemma 1.

Proof. Proceed by induction on n. The result is true when $n=2$ for the only algorithm satisfying the requirements is the transposition algorithm T where $T(1)=T(2)=1$, and $\pi^T_1/\pi^T_2=\alpha$. Assume the bounds hold for forward moving algorithms on n items, and look at τ, a forward moving algorithm on $n+1$ items. Since $D\tau$ is a forward moving algorithm on n items, (1) gives

$$\frac{\pi^{\tau}_i}{\pi^{\tau}_{n+1}} = \frac{(1-\pi^{\tau}_1)\pi^{D\tau}_{i-1}}{(1-\pi^{\tau}_1)\pi^{D\tau}_n} = \frac{\pi^{D}_{i-1}}{\pi^{D}_n}, \quad i = 2, \ldots, n \tag{2.1}$$

and so, because of the induction hypothesis,

$$k \leqq \frac{\pi^{\tau}_i}{\pi^{\tau}_{n+1}} \leqq k^{n-(i-1)} = k^{n+1-i} \quad \text{if} \quad k \geqq 1, \quad i = 2, \ldots, n. \tag{2.2}$$

Thus we only have to prove that $k \leqq \frac{\pi^{\tau}_1}{\pi^{\tau}_{n+1}} \leqq k^n$.

Consider the first component of the equation $\underline{\pi}^{\tau}=\underline{\pi}^{\tau}P$. Since $T(1|\tau)$ is the set of positions that are mapped on the first position by τ, page I_1 can only be in the first position if it was in one of the positions of $T(1|\tau)$ and was referenced, or if it was in the first position and the page referenced was not in the set $T(1|\tau)$. Then

$$\pi^{\tau}_1 = \Big(ka \sum_{r\in T(1|\tau)} \pi^{\tau}_r\Big)+\pi^{\tau}_1(1-a|T(1|\tau)|) \tag{2.3}$$

where $|T(1|\tau)|$ denotes the number of positions in the set $\{j|\tau(j)=1, j>1\}$. Thus $\frac{\pi^{\tau}_1}{\pi^{\tau}_{n+1}} = \frac{k}{|T(1|\tau)|} \sum_{r\in T(1|\tau)} \pi^{\tau}_r/\pi^{\tau}_{n+1}$. If $k\geqq 1$, (2.2) already gives $1 \leqq k \leqq \frac{\pi^{\tau}_r}{\pi^{\tau}_{n+1}} \leqq k^{n+1-r} \leqq$ $\leqq k^{r+1-2}$. Thus $k^n \geqq \frac{\pi^{\tau}_1}{\pi^{\tau}_{n+1}} \geqq k$ for $k\geqq 1$. The induction is complete and a similar proof works for $k<1$.

APPENDIX 3. Proof of Lemma 2.

Proof. Assume $k\geqq 1$ (the coresponding result for $k<1$ follows similarly), and assume that $\pi^T_1 \geqq \pi^{\tau}_i$ for algorithms acting on n positions. The result is trivially true for $n=2$, and so we proceed by induction. For $n+1$ positions, using (2.3) and (1) gives

$$\pi^{\tau}_1 = \Big(k \sum_{r\in T(1|\tau)} \pi^{\tau}_r\Big)/|T(1|\tau)| = \Big(k \sum_{r\in T(1|\tau)} \pi^{D\tau}_{r-1}(1-\pi^{\tau}_1)\Big)/|T(1|\tau)|. \tag{3.1}$$

By the inductive hypothesis, $\pi^{D\tau}_{r-1} \leqq \pi^{DT}_1$, so

$$\pi^{\tau}_1 \leqq \alpha\pi^{DT}_1(1-\pi^{\tau}_1). \tag{3.2}$$

Rivest's result [4] is that $\frac{\pi^T_i}{\pi^T_n}=k^{n-1}$, so $\pi^T_1=k\pi^T_2$, while (1) gives $\pi^T_2=\pi^{DT}_1(1-\pi^T_1)$.

Thus

$$\pi_1^T = k\pi_1^{DT}(1-\pi_1^T). \tag{3.3}$$

Thus we have

$$\frac{\pi_1^\tau}{1-\pi_1^\tau} \leqq \frac{\pi_1^T}{1-\pi_1^T}$$

and so

$$\pi_1 \leqq \pi_1^T. \tag{3.4}$$

For π_i^τ, $i=2, \ldots, n+1$ (1) gives

$$\pi_i^\tau = \pi_{i-1}^{D\tau}(1-\pi_1^\tau) \leqq \pi_1^{DT}(1-\pi_1^\tau) = \frac{\pi_1^T}{k}\frac{(1-\pi_1)}{(1-\pi_1^T)}. \tag{3.5}$$

So it is sufficient to show that

$$\frac{1-\pi_1}{1-\pi_1^T} \leqq k.$$

Writing $\pi_i = K_i\pi_{n+1}^\tau$ as before, we get

$$\pi_{n+1}^\tau = \left(1+\sum_{i=1}^{n} K_i\right)^{-1}.$$

Thus

$$1-\pi_1^\tau = \sum_{i=2}^{n+1}\pi_i^\tau = \pi_{n+1}^\tau\left(1+\sum_{i=2}^{n} K_i\right) = \frac{1+\sum_{i=2}^{n} K_i}{1+\sum_{i=1}^{n} K_i} \tag{3.6}$$

$$\leqq \frac{1+\max\sum_{i=2}^{n} K_i}{1+k+\max\sum_{i=2}^{n} K_i}. \tag{3.7}$$

The inequality follows since (3.6) is a maximum when K_1 is as small as it can be, which is k from Lemma 1, and the sum of the rest of the K_i are as large as they can be.

Using the inequalities of Lemma 1

$$\frac{1+\max\sum_{i=2}^{n} K_i}{1+k+\max\sum_{i=2}^{n} K_i} = \frac{1+\sum_{i=2}^{n} k^i}{1+k+\sum_{i=2}^{n} k^i} = \frac{k^n-1}{k^n+k^2-k-1}.$$

From Rivest's result $(1-\pi_1^T)=\frac{k^n-1}{k^{n+1}-1}$, so we have $(1-\pi_1)\leq k(1-\pi_1^T)$, since $\frac{k^n-1}{k^n+k^2-k-1}\leqq\frac{k(k^n-1)}{(k^{n+1}-1)}$ for k positive. This completes the induction

Appendix 4. Proof of Lemma 3.

Proof. The lemma is trivially true for $n=2$. Assume that it is true for up to n positions, and consider C and another such algorithm τ on $n+1$ positions. Since C acts as the transposition algorithm T, on the first $M+1$ positions, it follows that $\pi_1^C=k\pi_2^C$ just as $\pi_1^T=k\pi_2^T$. Thus (3.2), (3.3), (3.4) follow as in Appendix 3, replacing

T by C, giving $\pi_1^\tau \leqq \pi_1^C$. For $i=2, \ldots, n+1$, we have

$$\pi_i^\tau = \pi_{i-1}^{p\tau}(1-\pi_1^\tau) \leqq \pi_1^{pC}(1-\pi_1^\tau) = \frac{\pi_1^C}{k}\frac{(1-\pi_1^\tau)}{(1-\pi_1^C)}.$$

It is sufficient to show $\dfrac{1-\pi_1^\tau}{1-\pi_1^C} \leqq k$. As C is a forward moving algorithm the induction of theorem 1 gives $\pi_1^T \geqq \pi_1^C$ and so

$$\frac{1}{1-\pi_1^T} \geqq \frac{1}{1-\pi_1^C}.$$

So

$$\frac{1-\pi_1^\tau}{1-\pi_1^C} \leqq \frac{1-\pi_1^\tau}{1-\pi_1^T} \leqq k.$$

Abstract

A brief survey is given of developments in the study of self organizing paging algorithms and the associated library problem. It has been conjectured that two related algorithms, transposition and climb, are optimal in these fields and we establish this optimality for a specific distribution of page references.

* DEPARTMENT OF MATHEMATICS
POLYTECHNIC OF THE SOUTH BANK
LONDON

** DEPARTMENT OF DECISION THEORY
UNIVERSITY OF MANCHASTER
MANCHESTER

References

[1] AHO, A. V., P. J. DINNING, J. D. ULLMANN, Principles of optimal page replacement, *J. Assoc. Comput. Mach.*, v. 18, 1971, pp. 80—93.
[2] ARATÓ, M., A note on optimal performance of page storage, *Acta Cybernet.*, v. 2, 1976, pp. 25—30.
[3] BÉLÁDY, L. A., A study of replacement algorithms for a virtual storage computer, *IBM Systems J.*, v. 5, 1966, pp. 78—101.
[4] BENCZUR, A., A. KRÁMLI, J. PERGEL, On the Bayesian approach to optima performance of page storage hierarchies, *Acta Cybernet.*, v. 3, 1976, pp. 79—89.
[5] BURVILLE, P. J. and J. F. C. KINGMAN, On a model for storage and search, *J. Appl. Probab.*, v. 10, 1973, pp. 697—701.
[6] FRANASZEK, P. A. and T. J. WAGNER, Some distribution-free aspects of paging algorithm performance, *J. Assoc. Comput. Mach.*, v. 21, 1974, pp. 31—39.
[7] GELENBE, E., A unified approach to the evaluation of a class of replacement algorithms, *IEEE. Trans. Comput.*, v. 22, 1973, pp. 611—617.
[8] HENDRICKS, W. J., The stationary distribution of an interesting Markov chain, *J. Appl. Probab.*, v. 9, 1972, pp. 231—233.
[9] HENDRICKS, W. J., An extension of a theorem concerning an interesting Markov chain, *J. Appl. Probab.*, v. 10, 1973, pp. 886—890.
[10] INGARGIOLA, G. and J. F. KORSH, Finding optimal demand paging algorithms, *J. Assoc. Comput. Mach.*, v. 21, 1974, pp. 40—53.
[11] KNUTH, D. E., *The art of computer programming* (Volume 3: Sorting and Searching), Addison—Wesley Publishing Co. Reading, Mass., 1973.
[12] LETAC, G., Transcience and recurrence of an interesting Markov chain, *J. Appl. Probab.*, v. 11, 1974, pp. 818—824.
[13] MCCABE, J., On serial files with relocatable records, *Oper. Res.*, v. 13, 1965, pp. 609—618.
[14] RIVEST, R. L., On self-organising sequential search heuristics, *Comm. ACM*, v. 19, 1976, pp. 63—67.
[15] THOMAS, L. C., Optimal replacement of library type Markov chains, University of Manchester, Department of Decision Theory, Note No. 22, 1975.

(Received Oct. 24, 1979)