An Optimality Proof of the LRU-*K* Page Replacement Algorithm

ELIZABETH J. O'NEIL AND PATRICK E. O'NEIL

University of Massachusetts, Boston, Massachusetts

AND

GERHARD WEIKUM

University of the Saarland, Saarland, Austria

Abstract. This paper analyzes a recently published algorithm for page replacement in hierarchical paged memory systems [O'Neil et al. 1993]. The algorithm is called the LRU-*K* method, and reduces to the well-known LRU (Least Recently Used) method for $K = 1$. Previous work [O'Neil et al. 1993; Weikum et al. 1994; Johnson and Shasha 1994] has shown the effectiveness for $K > 1$ by simulation, especially in the most common case of $K = 2$. The basic idea in LRU-*K* is to keep track of the times of the last K references to memory pages, and to use this statistical information to rank-order the pages as to their expected future behavior. Based on this the page replacement policy decision is made: which memory-resident page to replace when a newly accessed page must be read into memory. In the current paper, we prove, under the assumptions of the independent reference model, that LRU-*K* is optimal. Specifically we show: given the times of the (up to) K most recent references to each disk page, no other algorithm A making decisions to keep pages in a memory buffer holding $n - 1$ pages based on this information can improve on the expected number of I/Os to access pages over the LRU-*K* algorithm using a memory buffer holding n pages. The proof uses the Bayesian formula to relate the space of actual page probabilities of the model to the space of observable page numbers on which the replacement decision is actually made.

Categories and Subject Descriptors: H.2.2 [**Database Management**]: Physical Design

General Terms: Algorithms, Design, Performance, Theory

Additional Key Words and Phrases: Buffering, LRU, LRU-*K*, optimality, page-replacement

1. *Introduction*

Computer systems have a hierarchy of data storage media. We will speak in terms of how to buffer data from inexpensive, slow-access *disk* in expensive,

---

Authors' addresses: E. J. O'Neil and P. E. O'Neil, Department of Computer Science, University of Massachusetts/Boston, 100 Morrisseey Boulevard, Boston, MA 02125-3393, e-mail: {eoneil, poneil}@cs.umb.edu; G. Weikum, University of the Saarland, Department of Computer Science, P.O. Box 15110, D-66041 Saarbrücken, Germany, e-mail: weikum@cs.uni-sb.de.

fast-access *memory*, but the buffering algorithm presented here can be used for any pair of media in the hierarchy, as long as there is a very significant access-speed difference between the two. Thus, the algorithm could apply to buffering low-speed bank memory on high-speed memory cache, or low-speed optical disk on high-speed magnetic disk.

Concentrating on disk and memory in what follows, the storage capacity of both are divided into equal-sized pages, typically of 4K bytes (2^{12} bits). All pages belonging to a single medium, memory or disk, are accessible in (approximately) equal time, but the access speed varies greatly between the two with current technology. Disk access time is typically .01 second, compared to 10^{-8} seconds for accessing (one word) of a page in memory, an access-speed ratio of 10^6. Memory is used to hold copies of data pages brought in from disk during the time that programs need to access data on those pages. Typically a fixed number of memory pages are allocated for this purpose, and these memory pages are called *buffers.* Given that we can allocate a reasonably large amount of memory space for buffering, the challenge is to retain popular disk pages in buffer even when they are not being accessed for the moment, so as to reduce the number of disk accesses that must be performed in a long-term workload. In doing this, we are paying for extra memory in order to reduce expenditures on disk arms needed to perform the required rate of access. At the same time, we are also reducing the average delay (often called *latency*) for pages to be accessible in memory. See Gray and Putzolu [1987] for theoretical justification for this approach.

A critical decision in buffer management arises when a new buffer is needed for a page about to be read in from disk and all buffers are in use. What current page of data should be dropped from the memory buffer system to free a buffer? This is known as the *page replacement policy*, and the different buffering algorithms take their names from the type of replacement policy they impose (see, e.g., Coffman and Denning [1973] and Effelsberg and Haerder [1984]). The algorithm historically utilized by almost all commercial database and file systems is known as LRU, for Least Recently Used. When a new buffer is needed, the LRU policy drops the page from the memory buffers that has not been accessed for the longest time. LRU buffering was developed originally for patterns of use in instruction logic (see, e.g., Denning [1968] and Coffman and Denning [1973]).

To simplify our language, we speak of pages being *in memory* when they reside in one of the buffers of the page buffering system, and *dropped from memory* when they lose their buffer, following Coffman and Denning [1973]. This reflects current practice for database systems: the only in-memory location for disk pages is one of the memory buffers.

The LRU buffering algorithm has a flaw, in that it decides what page to drop from memory based on too little information, limiting itself to only the time of last reference. Specifically, LRU is unable to differentiate between pages that have relatively frequent references and pages that have very infrequent references until the system has wasted a lot of resources keeping infrequently referenced pages in memory for an extended period. In this paper we analyze the idea of using the history of the last two references, or more generally the last K references, $K \geq 2$, for each page. The specific algorithm, presented originally in O'Neil et al. [1993], that takes into account knowledge of the last two references to a page is named LRU-2, and the natural generalization is the LRU-K algorithm. We refer to the classical LRU algorithm within this taxonomy as LRU-1.

Previous simulation studies [O'Neil et al. 1993; Weikum et al. 1994; Johnson and Shasha 1994] show that the LRU-2 algorithm performs very well under several types of simulation tests. For $K > 2$, the LRU-*K* algorithm provides somewhat improved performance over LRU-2 for stable patterns of access, but requires more overhead storage and is less responsive to changing access patterns, an important consideration for some applications. Thus, LRU-2 is the most flexible case and is commonly used in practice. For large *K*, this algorithm approaches the LFU (least frequently used) algorithm, where all the page references are used to estimate the use frequencies. In contrast to LFU, however, LRU-*K* is adaptive to evolving distributions of page reference frequencies.

In this paper, we prove the optimality of LRU-*K* among all replacement algorithms that have the limited knowledge of the *K* most recent references to a page and no knowledge of future accesses. Optimality refers to the expected costs for processing a page reference string where the expectation is derived from a probability distribution for accessing individual pages under the assumption that the access probabilities at different points in the reference string are independent. These assumptions are known in the literature as the independent reference model [Coffman and Denning 1973].

An alternative notion of optimality would be based on the competitiveness of a page replacement algorithm [Sleator and Tarjan 1985] which is an asymptotic worst-case bound for the factor by which the algorithm's access costs exceed the costs of an optimal off-line algorithm with perfect *a priori* knowledge of a reference string. Note that this consideration is independent from a probability distribution of page references, and further note that the optimal off-line algorithm is, of course, infeasible in a system that requires on-line decisions. It has been shown in Sleator and Tarjan [1985] that both LRU and FIFO (i.e., a first-in-first-out replacement policy) have an optimal competitiveness (among all deterministic algorithms) of m where m is the number of memory buffers. However, this worst-case analysis is based on a fictitious adversary that, given a replacement algorithm, may select the worst possible reference string with regard to the algorithm under consideration. The fact that algorithms of significantly different practical quality such as LRU and FIFO fall into the same (optimal) competitiveness class indicates the inadequacy of the notion of competitiveness for a practically relevant assessment of buffer replacement algorithms [Koutsonpias and Papadimitriou 1994].

The remainder of this paper has the following outline. In Section 2, we present the basic concepts of the LRU-*K* approach to page replacement. In Section 3, we give some mathematical underpinnings, and demonstrate that the LRU-*K* algorithm is optimal under the independent page reference model.

2. *LRU-K Page Replacement Algorithm Definition*

Assume we are given a set $N = \{1, 2, \ldots, n\}$ of disk pages, denoted by positive integers, and that the database system under study makes a succession of references to these pages specified by the *reference string*: $r_1, r_2, \ldots, r_t, \ldots$, where $r_t = p$ $(p \in N)$ means that term numbered t in the references string refers to disk page p. In the following discussion, we will measure all *time* in terms of counts of successive page accesses in the reference string (t for r_t) rather than clock time.

Definition 2.1. *Backward K-distance* $b_t(p, K)$; *LRU-K age of a page.* Given a reference string known up to time t, $r_1, r_2, \ldots, r_t$, the backward K-distance $b_t(p, K)$ is the distance backward in subscript from t to the Kth most recent reference to the page p.

$$\begin{aligned} b_t(p, K) &= g, \quad \text{if } r_{t-g} \text{ has the value } p \text{ in the page reference string } r_1, r_2, \ldots, \\ &\qquad\quad r_t \text{ and there have been exactly } K - 1 \text{ other positions } i \text{ with} \\ &\qquad\quad t - g < i \le t, \text{ where } r_i = p \\ &= \infty, \quad \text{if } p \text{ does not appear at least } K \text{ times in } r_1, r_2, \ldots, r_t \end{aligned}$$

We will often refer to $b_t(p, K)$ as the *LRU-K age* of the page p.

Definition 2.2. *LRU-K Algorithm.* The LRU-K Algorithm specifies a page replacement policy when a buffer is needed for a new page being read in from disk: the page p to be dropped (i.e., selected as a replacement victim) is the one whose Backward K-distance, $b_t(p, K)$, is the maximum of all pages in memory buffers. The only time the choice is ambiguous is when more than one page has $b_t(p, K) = \infty$. In this case, a subsidiary policy may be used to select a replacement victim among the pages with infinite Backward K-distance. Pages referenced the smallest number of times would always be chosen first for page replacement, but within each group of pages referenced only L times, $L < K$, LRU-L could be employed as a subsidiary policy.

Note that LRU-1 corresponds to the classical LRU algorithm.

We are imposing what seems to be a rather artificial restriction on our use of page reference history: We propose to use only the most recent K references to all pages at time t, not the entire history of references. We map this information into a single LRU-K age value $b_t(p, K)$ for each page. At time $t + 1$ after another reference to some page p has occurred, assuming that there have now been at least $K + 1$ references to page p, we will have forgotten the oldest reference to this page. The practical justification for this definition is that page reference probabilities may change over time (typifying a quasi-stable system) and so we want to weight recent references more heavily than older ones. Further, page reference history data takes resources to maintain, and we will argue a bit later that an algorithm that keeps a history of an equal number of references for all pages likely to be popular will use these resources in the most effective manner.

The LRU-2 algorithm has been shown in simulation [O'Neil et al. 1993; Weikum et al. 1994] to significantly improve on LRU-1 in a number of different situations. Note that by taking into account the last two references to a page, we are able for the first time to estimate the expected page reference probability by measurement of an actual interarrival *interval* between references, rather than estimating the interarrival interval only by the time from the current instant back to the most recent reference, as is done with LRU-1. We are using more relevant information and our estimates are immensely improved as a result, especially for pages that are infrequently referenced and should therefore be dropped from memory quickly; LRU-1 is never able to do this. Note that we make no general assumptions about the probabilistic distribution of page references. In Section 3, we will assume a model with stationary and independent page reference probabilities (the independent reference model) to demonstrate optimality of the LRU-K algorithm in that case, but the LRU-K approach can be experimentally shown to respond well in workloads that have other access patterns, such as those arising from looping behavior in a query (a distribution of interarrival times that is highly *skediastic*, in statistical terminology). In the independent reference model case, we can generalize slightly to allow the page reference probabilities to have the potential for occasional changes over time, only assuming that changes are infrequent enough that a statistical approach to future page access based on past history is usually valid, that is, we assume quasi-stability of the page reference probabilities. It is in this sense that we refer to the LRU-2 approach as being responsive to changes in access patterns.

3. *Optimality Analysis*

In this section, we demonstrate that the LRU-2 algorithm, under a standard model of page reference behavior, provides essentially optimal buffering behavior based on the information given. The proof generalizes easily to LRU-*K*.

We take as a starting point for our analysis the *Independent Reference Model* for paging presented in Section 6.6 of Coffman and Denning [1973]. Our notation comes from this text, but we make a few minor departures from the model presented. We begin with a set $N = \{1, 2, \ldots, n\}$ of disk pages already mentioned, and a set $M = \{1, 2, \ldots, m\}$ of memory page buffers, $1 \leq m \leq n$. According to the independent reference assumption, the reference string $\omega = r_1, r_2, \ldots, r_t, \ldots$ is an infinite sequence of independent random variables with the common stationary distribution $\{\beta_1, \beta_2, \ldots, \beta_n\}$, one probability for each of the n disk pages, where $P(r_i = p) = \beta_p$, for all $p \in N$ and all references r_i in ω. Let the random variable $d_t(p)$ denote the time interval forward to the next occurrence of p in the reference string after time t, reference r_t. From the assumptions above, $d_t(p)$ has the stationary exponential distribution:

$$P(d_t(p) = j) = \beta_p(1 - \beta_p)^{j-1}, j = 1, 2, \ldots \tag{3.1}$$

with mean interreference interval $I_p = 1/\beta_p$ for the page. Because of the stationary assumption, $P(d_t(p) = j)$ is independent of t. Further note that the independent reference assumption imposes a probability distribution $P(\omega_T)$ on all reference strings ω_T of any given length T. If $\omega_T = r_1, r_2, \ldots, r_T$, then $P(\omega_T) = \beta_{r_1} \cdot \beta_{r_2} \cdot \ldots \cdot \beta_{r_T}$, the product of the probabilities of the individual references making up the reference string.

Definition 3.1. *The A_0 Buffering Algorithm.* Let A_0 denote the buffering algorithm that replaces the page p in memory whose expected value I_p is a maximum, that is, the page for which β_p is smallest.

THEOREM 3.1 OPTIMALITY OF A_0 (THEOREM 6.3 IN COFFMAN AND DENNING [1973]). *Algorithm A_0 is optimal with respect to the probability distribution $P(\omega)$ of reference strings implied by the independent reference assumption.*

PROOF. See Coffman and Denning [1973] and Aho et al. [1971]. □

The A_0 algorithm can be described as an *optimal strategy with perfect probabilistic information but without an oracle.* Alternatively, Belady's algorithm [Belady 1966], designated as B_0 in Aho et al. [1971] and Coffman and Denning [1973] assumes complete knowledge of a specific reference string ω, and takes the strategy of retaining in memory those pages that will be re-referenced again the shortest time in the future. However, to achieve a strategy such as this Belady's algorithm requires an *oracle* that can look into the future; complete knowledge of all details of the probabilistic situation will not allow us to generate such predictions. For this reason, we claim that Belady's optimal strategy is unapproachable in real systems, and that we should use the A_0 strategy as the proper measure of how closely a strategy approaches optimal behavior.

In the development that follows, we take as a given that the independent reference model holds for our reference string $\omega = r_1, r_2, \dots, r_t, \dots$, with some reference probability vector $\boldsymbol{\beta} = \{\beta_1, \beta_2, \dots, \beta_n\}$ for the n disk pages of N. To reflect the normal state of ignorance concerning reference probabilities with which a buffering algorithm starts, we do not assume foreknowledge of the proper ordering of the pages, $v = 1, 2, \dots, n$, where the probability $P(r_i = v)$ of a reference to a page v obeys: $P(r_i = v) = P(v) = \beta_v$. Rather, we assume that the references in the string $\omega = r_1, r_2, \dots, r_t, \dots$ are given in terms of *observable page numbers*, $p = 1, 2, \dots, n$, and that there is some permutation mapping, x: $N \rightarrow N$, from observable page numbers to proper page numbers, initially unknown to us, such that the probability of a reference to observable page p is $P(r_i = p) = P(p) = \beta_{x(p)}$. In what follows, we statistically estimate $P(p)$ for each observable page p, based on a history of references to the page and a knowledge of the vector $\boldsymbol{\beta}$.

Realistically, the buffering algorithm will have no knowledge of component values in the probability vector $\boldsymbol{\beta}$ either, but it turns out that we do not need the exact values of these components. Our analysis to seek a statistical estimate of $P(p)$ allows us to derive ordinal properties associated with these quantities and prove the following principle: given any reference probability vector $\boldsymbol{\beta}$ with at least two distinct values, we can conclude that for any two disk pages p and q, if $b_t(p, K) < b_t(q, K)$, then the estimated value for $P(p)$ is greater than or equal to the estimated value for $P(q)$ and thus page p should be retained in memory in preference to page q. Of course, this is exactly what the LRU-K algorithm of Definition 2.2 will do. If the reference probability vector $\boldsymbol{\beta}$ does not have two distinct values, this means that all pages have equally likely probability of reference, so the buffering algorithm used is immaterial.

3.1 LRU-*K* Page History Information. We need to develop some notation to prepare for the proofs that follow.

Definition 3.2. *$H(K, \omega, t)$ and the vector* **w**. We use the following notation for page history information used by the LRU-K algorithm. Given an integer $K \geq 1$, a reference string of observable page numbers $\omega = r_1, r_2, \dots, r_t, \dots$, and a specific time t, the function H maps the triple (K, ω, t) into a vector of length t:

$$H(K, \omega, t) = \mathbf{w} = (w_1, w_2, \dots, w_t),$$

where:

$$\begin{aligned} w_i &= r_{t-i+1}, && \text{if this value occurs at most } K \text{ times in } r_{t-i+1}, r_{t-i+2}, \dots, r_t, \\ &= 0, && \text{otherwise.} \end{aligned}$$

Basically, $\mathbf{w}$ reverses ω, reading back from time t; only the most recent K references in ω are replicated in $\mathbf{w}$ and the other references are replaced by 0.

In what follows, we often assume that $K = 2$ and t and ω are implicit, so that $\mathbf{w}$ can stand alone as a representation of ω. When we wish to emphasize the dependency, we will write $H(K, \omega, t)$.

Definition 3.3. *Gaps in* **w.** A position k in the vector $\mathbf{w}$ that contains the value 0 is referred to as a *gap* in $\mathbf{w}$. The set of all such positions in $\mathbf{w}$ is denoted by $G(\mathbf{w})$.

Example 3.1. Consider the following reference string ω with three disk pages, that is, $n = 3$.

$$\omega = 2, 3, 1, 3, 3, 1, 2, 3, 1, 3, 2, 3, 2, 3, 1, 3, \ldots .$$

We have listed 16 references of ω here. The corresponding vector $\mathbf{w}$, with $K = 3$ and $t = 16$, is:

$$H(3, \omega, 16) = \mathbf{w} = (3, 1, 3, 2, 3, 2, 0, 1, 0, 2, 1, 0, 0, 0, 0, 0).$$

If instead $K = 2$, we have:

$$H(2, \omega, 16) = \mathbf{w} = (3, 1, 3, 2, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0).$$

We see that $\mathbf{w}$ lists page references starting with the most recent in early entries, with subscripts 1, 2, 3, ..., and going back in time in later entries as the subscripts increase. Gaps in $\mathbf{w}$ can start as early as entry $K + 1$, for example with the third entry if $K = 2$ and the most recent three references in ω ending with entry t were identical, say: 3, 3, 3. Gaps generally become more and more prevalent as we move to later entries, until (typically) some point where all the remaining entries are 0. For $K = 2$, and assuming all n page numbers have been referenced at least twice by time t, there are exactly $2n$ observable page numbers in $\mathbf{w}$ and $(t - 2n)$ gaps.

We claim that the LRU-K algorithm conceptually maintains the information contained in the vector $\mathbf{w}$ as successive references occur in the reference string ω with larger terminal values t. Of course, the data structure used by a paging system to maintain this information will be more efficient than a vector with a large number of component values equal to 0. In fact, as shown in O'Neil et al. [1993], it turns out to be unnecessary to keep track of all K references to relatively unpopular pages whose oldest references have no bearing on retaining that page in buffer in any conceivable future sequence of references. However, we will assume for notational purposes that the LRU-K algorithm maintains the vector $\mathbf{w}$ in what follows.

3.2. PROBABILISTIC FUNDAMENTALS. As mentioned earlier, we assume that the page reference numbers in the string ω are not *proper* page numbers (indices of the vector $\boldsymbol{\beta}$) but rather *observable* page numbers.

Definition 3.4. *The permutation* x; *the vector* $\mathbf{s} = x(\mathbf{w})$. For notational purposes, we postulate a bijective mapping x from observable page numbers to proper page numbers, $x\colon N \to N$, so that the probability of a reference to observable page i is $P(i) = \beta_{x(i)}$. Given this mapping x, we can map the $\mathbf{w}$ vector into a vector $\mathbf{s}$ as follows.

$$\mathbf{s} = (s_1, s_2, \ldots, s_t) = x(\mathbf{w}) = (x(w_1), x(w_2), \ldots, x(w_t)),$$

We define $x(0)$ to have the value 0, so that the vector **s** has gaps (values of 0) in the same positions as the vector **w**.

Example 3.2. Consider the vector **w** defined with $K = 2$ in Example 3.1.

$$H(2, \omega, 16) = \mathbf{w} = (3, 1, 3, 2, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0).$$

If we assume that the permutation x sends (1, 2, 3) to (2, 3, 1), then:

$$\mathbf{s} = (1, 2, 1, 3, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0).$$

Given knowledge of $\boldsymbol{\beta}$, once we know the s_k values, we know their page probabilities, and we will be able to calculate the probability of the various histories **s**.

In what follows, we will assume a generic value for t and $K = 2$ unless otherwise noted. The discussion generalizes to $K > 2$.

Definition 3.5. *Double-k page; D_k*(**s**)*; D_k*(**w**). When we see a gap at position k in the vector **s** (or equivalently, **w**), $s_k = 0$, the gap represents some page that has already occurred twice to the left of position k in **s** (or **w**). For a general position k that is not necessarily a gap, we say that a page that occurs twice in **s** (or **w**) in positions less than or equal to k is a *double-k page*. Let:

$D_k(\mathbf{w})$ = the set of page numbers that occur twice in positions less than or equal to k in **w**.
$D_k(\mathbf{s})$ = the set of page numbers that occur twice in positions less than or equal to k in **s**.
$D_k(\mathbf{s})$ and $D_k(\mathbf{w})$ are equal in cardinality, and in fact:

$$D_k(\mathbf{s}) = \{x(j) | j \in D_k(\mathbf{w})\}.$$

Choosing $D_k(\mathbf{s})$ to describe, we see that it is the empty set for $k < 2$ and the sequence of $D_k(\mathbf{s})$ sets for increasing k grow by acquiring new elements, so: $D_{k-1}(\mathbf{s}) \subseteq D_k(\mathbf{s})$. Equality holds when position k represents a gap or the position of a page number occurring for the first time in **s**. If t is sufficiently large that all disk pages occur twice in **s**, then there exists a position $m \leq t$ such that: $|D_{m-1}(\mathbf{s})| = n - 1$ and $|D_m(\mathbf{s})| = n$ (the notation $|D_k(\mathbf{s}))|$ represents the *cardinality* of $D_k(\mathbf{s})$). All positions to the right of m are gaps in **s**.

If there is a gap at position k in **s** (equivalently **w**), and $|D_k(\mathbf{s})| = 1$, $p \in D_k(\mathbf{s})$, then *we know by Definition 3.2 that p is the forgotten page number in position k*. In general, given a gap at position k, it is precisely the pages in $D_k(\mathbf{s})$ that could possibly have been in the gap position originally. Thus, the probability associated with gap position k in **s** is the sum of the page probabilities of all the pages represented in $D_k(\mathbf{s})$. The probability for a non-gap entry at position k in **s**, where a page p occurs, is just the individual page probability β_p. (Note that this statement about probability depends strongly on the fact that the referenced page number is in **s**, rather than in **w**.) The independent page reference model says that the probability of the whole vector **s**, $P(\mathbf{s})$, is just the product of probabilities for each reference. This discussion has gone to show the following result.

THEOREM 3.2. *Given $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, with t fixed and $K = 2$. If we represent by $N(\mathbf{s})$ the non-gap positions in* **s** *and by $G(\mathbf{s})$ the gap positions in* **s**, *we have:*

$$P(\mathbf{s}) = \prod_{i \in N(\mathbf{s})} \beta_i \prod_{k \in G(\mathbf{s})} \sum_{i \in D_k(\mathbf{s})} \beta_i.$$

PROOF. The formula given represents a product of the probabilities for each of the gap positions and non-gap positions for the vector **s**. □

Example 3.3. Consider the vector **s** following, for $K = 2$ and $t = 16$:

$$\mathbf{s} = (1, 1, 0, 2, 0, 2, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0),$$

for which:

$$G(\mathbf{s}) = \{3, 5, 7, 9, 11, 12, 13, 14, 15, 16\}$$

$$D_3(\mathbf{s}) = D_5(\mathbf{s}) = \{1\}$$

$$D_7(\mathbf{s}) = D_9(\mathbf{s}) = \{1, 2\}$$

$$D_{11}(\mathbf{s}) = \cdots = D_{16}(\mathbf{s}) = \{1, 2, 3\}$$

Assume that $\boldsymbol{\beta} = (1/2, 1/3, 1/6)$, that is, $\beta_1 = 1/2$, $\beta_2 = 1/3$, and $\beta_3 = 1/6$. Then by Theorem 3.2:

$$\begin{aligned} P(\mathbf{s}) &= [\beta_1\beta_1\beta_2\beta_2\beta_3\beta_3][\beta_1\beta_1(\beta_1 + \beta_2)(\beta_1 + \beta_2)(\beta_1 + \beta_2 + \beta_3)^6] \\ &= [(1/2)(1/2)(1/3)(1/3)(1/6)(1/6)] \cdot [(1/2)(1/2)(5/6)(5/6)(1)(1)(1)(1) \\ &\quad \cdot (1)(1)]. \end{aligned}$$

The term in square brackets on the left takes the product of β_i for pages i occurring in the nongap positions: 1, 2, 4, 6, 8, and 10 respectively; the term in square brackets on the right is the product of sums of β_i for pages i that occur twice prior to the following gap positions: position 3 (only the page 1, so the sum is 1/2), 5 (again 1/2), 7 (both pages 1 and 2, 1/2 + 1/3 = 5/6), 9, (same, 5/6), 11, 12, 13, 14, 15, 16 (all pages occur twice to the left of position 11, so the sum is always 1).

Theorem 3.2 tells us the probability of a particular vector **s** occurring (as always, assuming a given t, $K = 2$, and a knowledge of the vector $\boldsymbol{\beta}$.) Since **w** and the transformation x together determine **s**, this is the same thing as saying that we have determined the probability of **w** occurring, given x, that is,

$$P(\mathbf{s}) = P(\mathbf{s}|\boldsymbol{\beta}) = P(\mathbf{w}|x \text{ and } \boldsymbol{\beta}).$$

Now what we would like to do is determine from the string of observed page numbers in **w**, the underlying likelihood of each of these page numbers occurring in the future. That is, we would like to determine from **w** how x maps observable page numbers to proper page numbers, so we will know how the observable page numbers map to the probabilities of $\boldsymbol{\beta}$. We can achieve this if we can determine $P(x|\mathbf{w})$ from what we know about $P(\mathbf{w}|x)$. This suggests the use of Bayes'

Theorem, which we return to below after a discussion of maximum likelihood estimation.

3.2.1. Maximum Likelihood Estimation (MLE). The technique of maximum likelihood estimation (see Berry and Lindgren [1990], for example) can work with the result of Theorem 3.2 directly, using $P(\mathbf{s}|\boldsymbol{\beta})$ as the likelihood function. The MLE method maximizes this function over the parameters of the underlying model, here the various individual page probabilities β_i. The individual page probabilities β_i that maximize $P(\mathbf{s}|\boldsymbol{\beta})$ are the most likely values of β_i consistent with the observed data, called the maximum likelihood estimators of β_i. Here we will show that the pages with least LRU-2 age are the ones with the largest maximum likelihood estimators of β_i.

Corollary 3.3. Maximum Likelihood Estimation. *The maximum likelihood estimators of β_i are monotonically decreasing in LRU-2 page order.*

Proof. Recall that the pages in the formula for $P(\mathbf{s})$ are numbered in LRU-2 order for convenience of discussion. $P(\mathbf{s}|\boldsymbol{\beta})$ is the joint probability function for the observations given the underlying model parameters β_i, and thus is the likelihood function to be maximized. Using a common approach, we note that maximizing $P(\mathbf{s}|\boldsymbol{\beta})$ is the same as maximizing $\log P(\mathbf{s}|\boldsymbol{\beta})$, since the log function is monotonic. For many possible histories, not all n page numbers will show up in the history, so the sum of the β_i could be below 1. But the monotonicity of $P(\mathbf{s}|\boldsymbol{\beta})$ in the individual β_i ensures that a maximal value will have sum 1. Using a Lagrange multiplier λ to handle this sum constraint, we need to maximize $\log(P(\mathbf{s}|\boldsymbol{\beta})) - \lambda \sum \beta_i$. We call this the *maximization condition* in what follows. By taking the partial derivative with respect to each β_i, we obtain a set of nonlinear equations in the β_i variables and λ. These equations have many terms in common, and this makes the set solvable, as we demonstrate in one case below.

Note that there can be runs of equal sets in the sequence of D_k sets in gap positions k as defined in Definition 3.5. In Example 3.3, $D_3(\mathbf{s}) = D_5(\mathbf{s}) = \{1\}$. For this computation, define E_1 to be the first nontrivial D_k set, and E_2 to be the first D_k set different from E_1, E_3 the first different from E_2, etc. Let g_k be the number of gaps with exactly the set of pages E_k occurring twice before each gap counted. In the example above, $g_1 = 2$ because there are 2 gaps, at 3 and 5, with D-sets D_3 and D_5 identical to E_1. We find the following sets of nonlinear equations, among the complete set of partial derivative equations arising from the maximization condition:

$$2\,\beta_i^{-1} + S_k - \lambda = 0 \quad \text{for all } i \text{ in } E_k, \quad \text{for each } k, \tag{MLE.1}$$

$$\left(\text{where } S_k = g_k\left(\sum_{i\varepsilon E_k} \beta_i\right)^{-1} + S_{k+1}, \text{ for each } k\right) \tag{MLE.2}$$

First, we note that S_k, being a sum over all i in E_k, is constant over i in E_k. Then, equation set (MLE.1) requires that all β_i with i in one particular set E_k are equal. We now denote that common value by β_k. The sum over the β_i in E_k becomes $|E_k|\beta_k$, where $|E_k|$ denotes the cardinality of E_k. Now substituting the right side of equation (MLE.2) in equation (MLE.1) for k and subtracting equation (MLE.1) for $k + 1$, we can eliminate the common terms $S_{k+1} - \lambda$, yielding:

$$2\ \beta_k{}^{-1} + \left(\frac{g_k}{|E_k|}\right)\beta_k{}^{-1} = 2\ \beta_{k+1}{}^{-1}$$

Since the term involving g_k is positive, this equation requires that β_{k+1} is smaller than β_k, for all k, following the LRU-2 age order. Similar arguments with the remaining equations confirm this conclusion. □

Although it is encouraging that MLE indicates that the "best" guesses for the β_i are consistent with the LRU-2 ordering, it doesn't evaluate the expected value of the page replacement cost, and of course an expected value weighs many possibilities, not just the most likely. Knowing the most-likely value is most valuable when it is reasonable to suppose that the distribution away from it is narrow, but that is far from true here.

Further, as listed earlier in the model assumptions, we would like to take the viewpoint that the β_i are not completely unknown. It is well known that some pages are more commonly referenced than others, and roughly how much. The thing we really don't know is which pages are which, at least at the buffer manager level. Thus, we turn now to a model that has given β_i but unknown mapping x from observable page numbers to proper page numbers.

3.2.2 A POSTERIORI PAGE IDENTIFICATION. We return now to the question of determining from the observed page history **w** how x (probabilistically) maps observable page numbers to the page numbers on which the β_i values are based; this will show us how the observable page numbers map to the probabilities of $\boldsymbol{\beta}$. The probabilistic information on x is $P(x|\mathbf{w})$, and this will be determined from the result of Theorem 3.2 about $P(\mathbf{s}) = P(\mathbf{w}|x)$ and the use of Bayes' Theorem.

LEMMA 3.3. A POSTERIORI PAGE IDENTIFICATION. *We are given $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, with t fixed and $K = 2$. There is some permutation mapping, x: $N \rightarrow N$ unknown to us, such that for each observable page number $p \in N$, the probability of reference, $P(p)$, is given by $\beta_{x(p)}$. A priori, we assume that all possible permutation mappings x have equal probabilities. Then for fixed* **w**, *there is a constant c such that for all x*:

$$P(x|\mathbf{w}) = c \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)}.$$

PROOF. By the preceding discussion, $P(\mathbf{s}) = P(\mathbf{w}|x)$. By Theorem 3.2:

$$P(\mathbf{s}) = P(\mathbf{w}|x) = \prod_{i \in N(\mathbf{s})} \beta_i \prod_{k \in G(\mathbf{s})} \sum_{i \in D_k(\mathbf{s})} \beta_i. \tag{3.2}$$

Note that the mapping x is known in this formula; gaps of **s** are at the same positions as the gaps of **w** and similarly for the non-gaps. Thus, for given **w**, we have:

$$\prod_{i \in N(\mathbf{s})} \beta_i = \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \text{ and } \prod_{k \in G(\mathbf{s})} \sum_{i \in D_k(\mathbf{s})} \beta_i = \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)}.$$

Thus, (3.2) reduces to:

$$P(\mathbf{s}) = P(\mathbf{w}|x) = \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)}. \tag{3.3}$$

By Bayes' formula (see, for example, Hodges and Lehmann [1970]), assuming M is the set of all possible mappings x: $N \to N$, we have the following result:

$$P(x|\mathbf{w}) = \frac{P(\mathbf{w}|x)P(x)}{\sum_{y \in M} P(\mathbf{w}|y)P(y)} \tag{3.4}$$

Since we are assuming that all possible permutation mappings $x \in M$ have equal *a priori* probabilities, we can cancel out the $P(x)$ against $P(y)$ in (3.4), and use the result in (3.3) to replace $P(\mathbf{w}|x)$ and $P(\mathbf{w}|y)$. The denominator sums over all mappings in M, so the result is constant with respect to fixed $\mathbf{w}$, say c_1, and 3.4 reduces to:

$$P(x|\mathbf{w}) = \left(\frac{1}{c_1}\right) \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)},$$

which is the result desired, taking the constant c to be $(1/c_1)$. □

3.3 Algorithmic Cost Formula. In Coffman and Denning [1973], the independent page reference model is defined with a cost based on the behavior of the whole reference string. Coffman and Denning then find that "demand paging" algorithms outperform the other algorithms, so that we need only study these algorithms. Demand paging means that pages are read into memory as late as possible in time consistent with the requirement to satisfy the page reference. At each reference, either 0 or 1 page is read in, depending on whether or not the page was in the buffer. The only thing that the algorithm decides on is which page to replace. Thus in this case the Coffman/Denning cost is exactly the time-average page fault probability. In any steady-state case, this will equal the expected page-fault probability over the universe of page-reference strings.

Definition 3.6. *Algorithmic Cost Given the Mapping x.* As above, we are given $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, with t fixed and $K = 2$. Assume that some buffering algorithm A uses the information present in the vector $\mathbf{w}$ at time t to determine the set B of observable disk pages, $|B| = m$, that are buffer resident at time t. (We assume A does not use an oracle.) We assume that as t increases, the vector $\mathbf{w}$ changes, and we are able to determine the set B of buffer resident pages from the algorithm A, the time t, and the vector $\mathbf{w}$. If we also know the mapping x from observable page numbers to proper page numbers, we can define the cost of the algorithm A:

$$C(A, \mathbf{w}, t, x) = 1 - \sum_{j \in B} \beta_{x(j)} \tag{3.5}$$

This is basically the probability that the next page referenced is not buffer resident, and thus represents the expected number of disk I/Os on the next reference (clearly less than one). It is more commonly called the *page fault probability.*

Formula 3.5 assumes knowledge of the mapping x. If we don't know this mapping but still know the vector **w**, it is still possible to estimate the probabilities of the various mappings x, $P(x|\mathbf{w})$, as shown in Lemma 3.3.

Definition 3.7 *Algorithmic Cost given only the vector* **w.** We define the cost of the algorithm A given only **w** and t (no x) to be the expected probability of a page fault, ranging over all possible x mappings in M.

$$C(A, \mathbf{w}, t) = 1 - \sum_{j \in B} \sum_{x \in M} \beta_{x(j)} P(x|\mathbf{w})$$

Plugging the result of Lemma 3.3 into the formula of Definition 3.7, we have the following:

THEOREM 3.4. *Given the assumptions of Definition* 3.6:

$$C(A, \mathbf{w}, t) = 1 - c \sum_{j \in B} \sum_{x \in M} \beta_{x(j)} \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)}, \tag{3.6}$$

where c is a constant for any fixed **w**.

PROOF. By substitution. □

Theorem 3.4 gives the expected cost of algorithm A observed history w through time t. We will show that LRU-*K*, among all algorithms without an oracle, minimizes this cost formula for each t and each possible vector **w**. Therefore, LRU-*K* has optimal steady-state behavior.

3.4. STATEMENT OF THE FUNDAMENTAL THEOREM 3.5. We are now in a position to state the Fundamental Theorem. For each specific vector **w**, we wish to determine the j values in B that minimize the cost in (3.6), which is a sum over all possible mappings x and therefore independent of x. We intend to show that the optimal memory resident pages in B are those m pages that appear earliest in $D_k(\mathbf{w})$. These are the pages of least LRU-2 age. Recall that, for gap positions k_i, the $D_{k_1}(\mathbf{w})$ form a nested sequence of sets, each one a superset of the last, including pages with a greater and greater range of LRU-2 ages. The following theorem shows that the page ordering by LRU-2 age is optimal.

THEOREM 3.5. INDIVIDUAL PAGE CONTRIBUTIONS TO ALGORITHMIC COST. *Define the individual page cost-savings function* $f(j)$ *and* $F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(n)})$ *as follows*:

$$\begin{aligned} f(j) &= \sum_{x \in M} \beta_{x(j)} \prod_{i \in N(\mathbf{w})} \beta_{x(i)} \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} \beta_{x(i)} \\ &= \sum_{x \in M} \beta_{x(j)} F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(n)}). \end{aligned} \tag{3.7}$$

Note that $f(j)$ is dependent on **w**, but to simplify notation below we do not make this dependency explicit. When $f(j)$ is summed over all j in B and subtracted from 1, it determines $C(A, \mathbf{w}, t)$ in formula (3.6) of Theorem 3.4. Thus, $f(j)$ represents an expected cost-savings contribution for observable page j. We will show that $f(j)$ is maximized by choosing j to be any element in $D_{k_1}(\mathbf{w})$, where k_1 is the first gap position, the next largest value comes when j is in $D_{k_2}(\mathbf{w})$ but not in $D_{k_1}(\mathbf{w})$, where k_2 is the next gap position with a strictly larger double-page set than $D_{k_1}(\mathbf{w})$, and so on through all the gap positions in **w**. All other pages in **w** that are represented twice (those that are doubled after the last gap) have one common $f(j)$ value that is smaller than the previously mentioned values. All other pages in **w** are represented once each, and have one common $f(j)$ value smaller yet than any $f(j)$ value for a doubly-represented page. Finally, the pages that never appear in **w** have a common, globally minimal $f(j)$ value.

3.5 Proof of the Fundamental Theorem 3.5

Definition 3.8 *Observable Page Numbering Rule for Proof.* In what follows, we will be working with a specific observable page history vector **w**, and since the page number order of observable pages is arbitrary, with a relationship between the observable pages and the underlying probabilities β_i to be determined, it is no loss of generality to assume that the first page number in **w** to occur twice (more generally, K times, but we are assuming $K = 2$) is 1, the second page number to occur twice is 2, and so on through all pages that occur twice in **w**. After that, all pages that occur once in **w** are numbered in order of their appearance, and finally all pages that never occur in **w** are numbered in any order.

Example 3.4 Consider a system with eight pages, six of which are referenced up to time $t = 16$. Below, we describe a set of significant positions in a vector **w**:

```
w =  (2, 1, 3, 1, 4, 2,| 0,  5,  0,| 3, 6, 4,|  0, 0, 0,   |      5)
     ------------------ |-----------|---------|------------|-----------------
     Before first gap   |  2 gap    | 3 and 4 |  3 gap     | 5 is doubled
   1 and 2 occur twice  | positions |  occur  | positions, | after last gap,
                        | with 1, 2 |  twice  | gaps in w  | 6 still single
                        |  doubled  |         | end        |
```

Note that pages 7 and 8 are not referenced. For this **w**, we have:

$$N(\mathbf{w}) = \{1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 16\}$$

$$G(\mathbf{w}) = \{7, 9, 13, 14, 15\}$$

$$D_7 = D_9 = \{1, 2\};\ D_{13} = D_{14} = D_{15} = \{1, 2, 3, 4\}$$

$$f(j) = \sum_{x \in M} \beta_{x(j)}\beta_{x(1)}{}^2\beta_{x(2)}{}^2\beta_{x(3)}{}^2\beta_{x(4)}{}^2\beta_{x(5)}{}^2\beta_{x(6)}(\beta_{x(1)} + \beta_{x(2)})^2 (\beta_{x(1)} + \beta_{x(2)} + \beta_{x(3)} + \beta_{x(4)})^3$$

$$= \sum_{x \in M} z_j F(z_1, z_2, \ldots, z_n), \text{ where } z_i = \beta_{x(i)} \text{ for all } i \text{ any page in } \mathbf{w},$$

$$\text{and } F(z_1, z_2, \ldots, z_n) = z_1{}^2 z_2{}^2 z_3{}^2 z_4{}^2 z_5{}^2 z_6 (z_1 + z_2)^2 (z_1 + z_2 + z_3 + z_4)^3$$

We see that F is a polynomial in the $z_j = \beta_{x(i)}$'s that involves their sums and products in an almost-symmetric fashion. We wish to show that $f(j)$ is nonincreasing, so we are interested in $f(j)$ vs. $f(j + 1)$ for each j.

For the case $j = 1$ vs. $j = 2$, we see that z_1 and z_2 show up only as their product and sum in F. We can then write:

$$F = S_{12}(z_1, z_2, \ldots, z_n),$$

where $S_{12}(z_1, z_2, \ldots, z_n)$ is symmetric in z_1 and z_2. We can write a similar expression to express the symmetry in z_3 and z_4, and more trivially, in z_7 and z_8. (All other adjacent pairs display asymmetry.) Below, we will show that this symmetry implies that $f(1) = f(2)$, that is, both pages that are doubled before the first gap are equally good in terms of expected cost.

For the case $j = 2$ vs. $j = 3$, we see that asymmetry comes into play through the termination of the first sum-of-z_i's at z_2, falling short of the z_3 needed to make a symmetric sum. Only one sum-of-z_i's will be affected this way. We see that the rest of the sum-of-z_i's are symmetric in 2 vs. 3, and so is the product of simple factors, so we can write:

$$F = S_{23}(z_1, z_2, \ldots, z_n)(z_1 + z_2)^2,$$

where $S_{23}(z_1, z_2, \ldots, z_n)$ is symmetric in z_2 and z_3. We can write a similar expression, with a sum of four z_i's, to express the deviation from symmetry in z_4 and z_5. This controlled asymmetry will be shown to lead to $f(3) < f(2)$, that is, the page doubled after the first gap has a smaller cost savings effect than the first two did.

The deviation from symmetry in the case 5 vs. 6 comes from $z_5{}^2\, z_6 = (z_5\, z_6)z_5$. We can write:

$$F = S_{56}(z_1, z_2, \ldots, z_n)z_5,$$

and similarly for case 6 vs. 7. These results will lead to $f(6) < f(5)$ and $f(7) < f(6)$.

We see from Example 3.4 that for some values of j, F is completely symmetric in z_j and z_{j+1}, while for other values, it is partially symmetric. In general, we can express the difference $f(j + 1) - f(j)$ in terms of the asymmetric part of F as follows:

LEMMA 3.6. $f(j) - f(j + 1)$ IN TERMS OF THE ASYMMETRIC PART OF F. *Recall from formula* (3.7) *the definition for $f(j)$ in terms of F, which we repeat here:*

$$f(j) = \sum_{x \in M} \beta_{x(j)} F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(n)}). \tag{3.7}$$

For any fixed j, let τ be the transposition permutation interchanging the integers j and $j + 1$. By definition, the set $\{x | x \in M\}$ is the full permutation group on integers 1 to n, and it is well known that if we consider the set of composed permutations $\{x \cdot \tau\}$ this also ranges over all permutations in M. Thus, we can rewrite (3.7) *as:*

$$f(j) = \sum_{x \in M} \beta_{x\cdot\tau(j)} F(\beta_{x\cdot\tau(1)}, \beta_{x\cdot\tau(2)}, \ldots, \beta_{x\cdot\tau(j)}, \beta_{x\cdot\tau(j+1)}, \ldots, \beta_{x\cdot\tau(n)}).$$

Since τ leaves all integers the same except j and $j + 1$, which are exchanged, this becomes:

$$f(j) = \sum_{x \in M} \beta_{x(j+1)} F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots, \beta_{x(n)}). \tag{3.8}$$

Next we write the formula for $f(j + 1)$ using the same form as that of (3.7):

$$f(j + 1) = \sum_{x \in M} \beta_{x(j+1)} F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(j)}, \beta_{x(j+1)}, \ldots, \beta_{x(n)}). \tag{3.9}$$

By the same trick of composing the transposition τ, we get:

$$f(j + 1) = \sum_{x \in M} \beta_{x(j)} F(\beta_{x(1)}, \beta_{x(2)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots, \beta_{x(n)}). \tag{3.10}$$

Now consider the value of $2 \cdot f(j) - 2 \cdot f(j + 1)$. We wish to show that this is positive, which will demonstrate that $f(j + 1) - f(j)$ is negative, as required in our Theorem statement. We evaluate $2 \cdot f(j) - 2 \cdot f(j + 1)$ by substituting the right side of the formulas (3.7 − 3.10) + (3.8 − 3.9), *and thus get:*

$$\begin{aligned}
&\sum_{x \in M} \beta_{x(j)} [F(\beta_{x(1)}, \ldots, \beta_{x(j)}, \beta_{x(j+1)}, \ldots) - F(\beta_{x(1)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots)] \\
&\qquad + \sum_{x \in M} \beta_{x(j+1)} [F(\beta_{x(1)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots) \\
&\qquad\qquad - F(\beta_{x(1)}, \ldots, \beta_{x(j)}, \beta_{x(j+1)}, \ldots)] \\
&\quad = \sum_{x \in M} [\beta_{x(j)} - \beta_{x(j+1)}][F(\beta_{x(1)}, \ldots, \beta_{x(j)}, \beta_{x(j+1)}, \ldots) \\
&\qquad\qquad - F(\beta_{x(1)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots)].
\end{aligned}$$

It immediately follows that if F is symmetric in $\beta_{x(j)}$ and $\beta_{x(j+1)}$, then $f(j) = f(j + 1)$. We see from the example that $f(j)$ is constant across various subintervals in j and drops at certain special j-values. Some of these special j values are determined by the expansion points in the $D_k(\mathbf{w})$ sequence of nested sets: at each point at which $D_k(\mathbf{w})$ expands to a larger set $D_{k+1}(\mathbf{w})$ because of the addition of pages $j + 1, \ldots$, then $f(j + 1) < f(j)$. Specifically, in Example 3.4, D_9 expands to D_{13} by the addition of pages 3 and 4, and $f(3) < f(2)$. Other special j values are determined by the pattern of powers in $N(\mathbf{w})$. In all cases, $f(j)$ is nonincreasing with j. Now we wish to show that this is true in general. By Lemma 3.6, we need to study certain products of differences, called $Ex(j)$ in Lemma 3.7.

The following lemma involves a quantity $Ex(j)$ that measures the contributions to $f(j + 1) - f(j)$ of a certain permutation x and another one x' that is the same as x except that the targets of the mappings for j and $j + 1$ are interchanged, that is, interchanging the pair of proper page identifications of j and $j + 1$. The positivity of $Ex(j)$ means that the lesser LRU-2-aged page, the one with lower j, is the better probabilistic choice for lowering the part of the algorithmic cost related to permutations x and x'.

LEMMA 3.7. *For any specific permutation x and any value j, $1 \leq j < n$, define $Ex(j)$ as follows:*

$$\begin{aligned} Ex(j) &= [\beta_{x(j)} - \beta_{x(j+1)}][F(\beta_{x(1)}, \ldots, \beta_{x(j)}, \beta_{x(j+1)}, \ldots) \\ &\qquad - F(\beta_{x(1)}, \ldots, \beta_{x(j+1)}, \beta_{x(j)}, \ldots)] \\ &= (z_j - z_{j+1})[F(z_1, z_2, \ldots, z_j, z_{j+1}, \ldots, z_n) \\ &\qquad - F(z_1, z_2, \ldots, z_{j+1}, z_j, \ldots, z_n)] \end{aligned}$$

where

$$F(z_1, z_2, \ldots, z_n) = \prod_{i \in N(\mathbf{w})} z_i \prod_{k \in G(\mathbf{w})} \sum_{i \in D_k(\mathbf{w})} z_i.$$

Recall that $D_k(\mathbf{w})$ is a nested sequence of sets. For two adjacent pages j and $j + 1$, we say they have different *membership patterns* in the $D_k(\mathbf{w})$ sequence if there is some k' for which j is in $D_{k'}(\mathbf{w})$ but $j + 1$ is not. We show that the following inequality holds:

$$Ex(j) \geq 0 \tag{3.11}$$

The value $Ex(j)$ is strictly positive exactly when $\beta_{x(j+1)} \neq \beta_{x(j)}$, that is, when $z_{j+1} \neq z_j$ and

(a) j and $j + 1$ have different membership patterns in the $D_k(\mathbf{w})$ sequence, or
(b) j and $j + 1$ have different multiplicities in $\mathbf{w}$, so that they have different powers in $\Pi_{i \in N(\mathbf{w})} z_i$.

PROOF. We start by showing that $Ex(j) = 0$ when these conditions on j fail. $Ex(j)$ is a product of two terms, each term the difference of two quantities which might be equal. Clearly, if $z_j = z_{j+1}$, then $Ex(j)$ is zero. If $z_j \neq z_{j+1}$, we need to show that $Ex(j) = 0$ when neither of (a) and (b) hold. Then, z_j and z_{j+1} appear in the same sum-of-z_i's and with the same powers in the product of simple factors. Thus, they appear in F only in the form of sums or products of z_j and z_{j+1} and the difference in F is 0 in these cases.

Now we show that if j and $j + 1$ have different membership patterns in the $D_k(\mathbf{w})$ sequence, or different multiplicities in $\mathbf{w}$, the value of $Ex(j)$ will be positive. We will factor F into a symmetric part $S_{j,j+1}$ and F_1, the nonsymmetric part of F. There are three cases of factorization, and we will prove $Ex(j) \geq 0$ after the three cases have been covered.

(1) The case (a) and not (b). Pages j and $j + 1$ have different membership patterns in the $D_k(\mathbf{w})$ sequence but the same multiplicities $\mathbf{w}$. This means z_j and z_{j+1} appear as $z_j + z_{j+1}$ everywhere in the product of sums except for the one factor that has z_j as its highest term, a factor with some power $u > 0$. They also both appear in the squared simple terms in the symmetric form $z_j^2 z_{j+1}^2$.

$$F = S_{j,j+1}[z_1 + z_2 + \cdots + z_j]^u$$

where $u > 0$.

(2) The case (b) and not (a). Pages j and $j + 1$ have different multiplicities in $\mathbf{w}$, but the same membership patterns in the $D_k(\mathbf{w})$ sequence. Since j and $j + 1$ have different multiplicities in $\mathbf{w}$, $j + 1$ does not appear twice and thus does not appear anywhere in the $D_k(\mathbf{w})$ sequence. Since pages j and $j + 1$ have the same membership patterns in the $D_k(\mathbf{w})$ sequence, neither one appears anywhere in the $D_k(\mathbf{w})$ sequence. This implies that z_j and z_{j+1} appear nowhere in the form $z_j + z_{j+1}$ in the product of sums, so the asymmetry comes only from the inequality of powers in the simple factors. By the page numbering rule, Definition 3.8, the multiplicity of j is greater than that of $j + 1$ and correspondingly the power of z_j is higher than that of z_{j+1}. We see that:

$$F = S_{j,j+1}(z_j)^a(z_{j+1})^b = S_{j,j+1}(z_jz_{j+1})^b(z_j)^{a-b}$$

where $a > b > 0$.

(3) For the overlapping case (a) and (b), j and $j + 1$ have different membership patterns in the $D_k(\mathbf{w})$ sequence and j and $j + 1$ have different multiplicities in $\mathbf{w}$, we see a combination of the effects of the previous two cases, so that

$$\begin{aligned} F &= S_{j,j+1}[z_1 + z_2 + \cdots + z_j]^u(z_j)^a(z_{j+1})^b \\ &= S_{j,j+1}[z_1 + z_2 + \cdots + z_j]^u(z_jz_{j+1})^b(z_j)^{a-b} \end{aligned}$$

where $a > b > 0$, $u > 0$.

Now we show that $Ex(j) \geq 0$. We note that the factor $(z_j\, z_{j+1})^b$ can be absorbed into $S_{j,j+1}$. Thus, we can write, for all cases, $F = S_{j,j+1}F_1$ where $S_{j,j+1}$ is symmetric in j and $j + 1$ and $F_1 = [z_1 + z_2 + \cdots + z_j]^u(z_j)^v$ where $u \geq 0$ and $v \geq 0$ and at least one of u or v is strictly positive. Note that this means that F_1 is monotonically strictly increasing in z_j. Also, it is constant in z_{j+1}. Thus:

$$\begin{aligned} Ex(j) &= (z_j - z_{j+1})[S_{j,j+1}F_1(z_1, \ldots, z_{j-1}, z_j) - S_{j+1,j}F_1(z_1, \ldots, z_{j-1}, z_{j+1})]. \\ &= S_{j,j+1}(z_j - z_{j+1})[F_1(z_1, \ldots, z_{j-1}, z_j) - F_1(z_1, \ldots, z_{j-1}, z_{j+1})]. \end{aligned}$$

Clearly this is positive if $z_j \neq z_{j+1}$. □

THEOREM 3.5 (RESTATED AND PROVED). *For each fixed* $\mathbf{w}$*, the value* $f(j)$ *is nonincreasing in* j*. If not all the* β_i *values are identical, then* $f(j) < f(j - 1)$ *for each* j *that satisfies* (*a*) *or* (*b*):

(*a*) *j is the smallest-numbered page involved in expanding* $D_k(\mathbf{w})$ *to* $D_{k+1}(\mathbf{w})$*, that is,* $j = min[D_{k+1}(\mathbf{w}) - D_k(\mathbf{w})]$.

(*b*) *j has a different (smaller) multiplicity in* $\mathbf{w}$ *than* $j - 1$.

Otherwise, $f(j) = f(j - 1)$.

PROOF. Lemma 3.6 demonstrates that $2 \cdot f(j) - 2 \cdot f(j + 1)$ is the value $Ex(j)$ summed over all permutations x. Then, in Lemma 3.7, we showed that $Ex(j)$ was nonnegative for any permutation x, and positive in the cases (a) and (b) above, and $\beta_{x(j+1)} \neq \beta_{x(j)}$. The assumption $\beta_{x(j+1)} \neq \beta_{x(j)}$ was meaningful in Lemma 3.7, where we were speaking of a specific permutation x, but in the sum of (3.7) over all $x \in M$ the values of $\beta_{x(j+1)}$ and $\beta_{x(j)}$ will differ from one permutation to another. However, it is easy to see that if any of the β_i values differ from any others, then some terms in the sum of (3.7) will be positive and the rest will be zero. Thus, $2 \cdot f(j) - 2 \cdot f(j+1)$ is positive and Theorem 3.5 has been demonstrated. □

Theorem 3.4 and 3.5 together show that $C(A, \mathbf{w}, t)$ is minimized by an algorithm that keeps in the m memory buffers those pages with the m smallest backward distances $b_t(j, 2)$. We state without proof that the generalization to $K \geq 2$ can be demonstrated, replacing $b_t(j, 2)$ with $b_t(j, K)$. Let us now consider the actual behavior of the LRU-K algorithm.

LEMMA 3.8. LRU-K STARTUP BEHAVIOR. *At any time t in a reference string ω after at least $m - 1$ pages have been referenced K or more times under this discipline, the LRU-K algorithm will have in memory the $m - 1$ pages with minimum values for $b_t(j, K)$ (which we will refer to as minimum LRU-K age, for short, or simply minimum age, with the K understood).*

PROOF. Starting from a first access at time 1, we proceed to build up a reference string ω, and at some time $t1$ a single page j has been referenced K times under this algorithm, giving it an age $b_{t1}(j, K) \leq t1$; at this time all the other pages have been referenced less than K times and have infinite LRU-K age. As accesses continue and the number of K-referenced pages increase, pages of infinite age are dropped in preference to the K-referenced pages until all but one of these pages have been replaced in buffer. At this time, the $m - 1$ pages of finite LRU-K age are in memory and constitute the set A of pages of the $m - 1$ minimum ages, the starting state for induction.

We now assume that the set A of $m - 1$ pages of least age among all pages N are in memory at time t, and show that this continues to hold after the access at $t + 1$. (This is known as the "steady state" for the LRU-K algorithm.) Assume that page p is referenced at time $t + 1$. All page ages advance by one except for that of page p, whose age decreases (assuming it has been referenced K times). Let A' be the set of $m - 1$ pages of least age at time $t + 1$. We need to show that all the pages of A' will be in memory.

Case 1. If p is in A, it was in memory originally and remains there, and its decreased age ensures its membership in A'. No pages are dropped from memory in this case. Thus, $A' = A$ is in memory.

Case 2. If p was in memory but not in A, it stays in memory and no pages are dropped (the page p must have been the only one of the m pages in memory not to lie in the set A at time t). The $m - 1$ pages in A are still less in age than any pages in N except possibly page p. Thus, the $m - 1$ minimal age pages of the set $(\{p\}$ UNION $A)$ have minimal ages in N, and so constitute A'. They are all in memory.

Case 3. If p was not in memory, it replaces the oldest page in memory, a page outside of A, so the pages in A remain in memory, and remain less in age than all other pages except possibly p. Again A' is just the set of $m - 1$ least-aged pages of $(\{p\}$ UNION $A)$, which are all in memory. □

THEOREM 3.9. OPTIMALITY OF LRU-K. *Under the independent page reference assumption, given knowledge $H(K, \omega, t)$ at time t, the expected cost in the steady-state case resulting from the LRU-K* algorithm acting with *m memory pages in its buffer system is less than that resulting from any other algorithm A acting with $m - 1$ memory pages. In fact, LRU-K acts optimally in all but (perhaps) one of its m page slots, an insignificant cost increment for large m.*

PROOF. By Theorems 3.4 and 3.5, under the model we have developed, the expected cost, given knowledge $H(K, \omega, t)$, is minimized by keeping in memory the $m - 1$ pages with minimum $b_t(i, K)$, and therefore the $m - 1$ pages with maximum expected values for $P(i)$. Clearly, no other algorithm A with $m - 1$ memory pages can improve on this to achieve lesser cost, since the minimum value of formula (3.6) (the cost formula) is achieved for $|B| = m - 1$ with these precise pages. □

In fact, LRU-K is optimal in behavior, not just in $m - 1$ pages, since all algorithms need one buffer to hold the most recently referenced page.

Note that all LRU-K results apply equally to LRU-1. In particular, the LRU algorithm is seen to act optimally (under the independent page reference assumption), given the limited knowledge it has of the most recent reference time.

4. *Concluding Remarks*

We have analyzed the LRU-K algorithm following Coffman and Denning [1973] in nomenclature. The use of Bayesian methods in this analysis is, to our knowledge, a new and entirely appropriate way of handling the real lack of knowledge of page identity at the memory-buffer level of the software.

The assumption of the independent reference model is a poor approximation to reality in many cases, although we believe it is better for database buffer systems than the classical operating systems case where memory buffering applies to both program code and data simultaneously. Code is accessed in a very nonrandom way; in fact, programs must loop to stay on the system long enough to impact it. The code part is removed from the picture in database buffer systems, since they buffer only data pages. Database data has extreme variations in page reference probabilities, from once a month (say) to many times a second, the "hot" data. Much of the hot data is not the actual data of database records, but the "index" data used to help access the records. Even a simple database with only one large (indexed) table will have relatively hot data in the upper reaches of the tree that implements the index. The variations in page reference probabilities are important to the effective operation of the LRU-K method, and further, we believe, to offsetting effects of dependence among the page references.

ACKNOWLEDGMENTS. We would like to acknowledge the help of Catherine O'Neil, who aided our proof of Theorem 3.5 by pointing out that the composition of a transposition with the symmetric group gives the symmetric group again. The authors would also like to thank the reviewer who advised us of the Maximum Likelihood Estimation method.

REFERENCES

AHO, A. V., DENNING, P. J., AND ULLMAN, J. D. 1971. Principles of optimal page replacement. *J. ACM 18*, 1 (Jan.), 80–93.

BELADY, L. A. 1966. A study of replacement algorithms for virtual storage computers. *IBM Syst. J. 5*, 2, 78–101.

BERRY, D. A., AND LINDGREN, B. W. 1990. *Statistics: Theory and Methods.* Brooks/Cole Publishing Company.

COFFMAN, JR., E. G., AND DENNING, P. J. 1973. *Operating Systems Theory.* Prentice-Hall, Englewood Cliffs, N.J.

DENNING, P. J. 1968. The working set model for program behavior. *Commun. ACM 11*, 5 (May), 323–333.

EFFELSBERG, W., AND HAERDER, T. 1984. Principles of database buffer management. *ACM Trans. Datab. Syst. 9*, 4 (Dec.), 560–595.

GRAY, J., AND PUTZOLU, F. 1987. The 5 minute rule for trading memory for disk accesses and the 10 byte rule for trading memory for CPU time. In *Proceedings of the 1987 ACM SIGMOD Conference* (San Francisco, Calif., May 27–29). ACM, New York, pp. 395–398.

HODGES, JR., J. L., AND LEHMANN, E. L. 1970. *Basic Concepts of Probability and Statistics*, 2nd Ed. Holden-Day, San Francisco, Calif.

JOHNSON, T., AND SHASHA, D. 1994. 2Q: A low-overhead high-performance buffer replacement algorithm. In *Proceedings of the 20th VLDB Conference* (Sept.), Morgan-Kaufmann, San Francisco, Calif., pp. 439–449.

KOUTSOUPIAS, E., AND PAPADIMITRIOU, C. H. 1994. Beyond competitive analysis. In *Proceedings of the IEEE Symposium on Foundations of Computer Science.* IEEE, New York, pp. 394–400.

O'NEIL, E. J., O'NEIL, P. E., AND WEIKUM, G. 1993. The LRU-K page replacement algorithm for database disk buffering. In *Proceedings of the 1993 SIGMOD International Conference on Management of Data* (Washington, D.C., May 26–28). ACM, New York, pp 297–306.

SLEATOR, D. D., AND TARJAN, R. E. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM 28*, 2, (Feb.), 202–208.

WEIKUM, G., HASSE, C., MOENKEBERG, A., AND ZABBACK, P. 1994. The comfort automatic tuning project. *Inf. Syst. 19*, 5 (July), 381–432.

RECEIVED JANUARY 1997; REVISED JULY 1997; ACCEPTED AUGUST 1998