JOURNAL OF ALGORITHMS 12, 685–699 (1991)

Competitive Paging Algorithms

AMOS FIAT,*^,1 RICHARD M. KARP,†,2 MICHAEL LUBY,‡,3
LYLE A. MCGEOCH,§ DANIEL D. SLEATOR,″,4 AND NEAL E. YOUNG#,5

*Department of Computer Science, Tel Aviv University, Tel Aviv, Israel 69978;
†Department of Electrical Engineering and Computer Science, Computer Science Division, University of California, Berkeley, CA 94720;
‡Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M6C 3B7;
§Department of Mathematics and Computer Science, Amherst College, Amherst, Massachusetts 01002;
″School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213;
and
#Computer Science Department, Princeton University, Princeton, New Jersey 08544.

Received November 1988; revised October 1990

The *paging problem* is that of deciding which pages to keep in a memory of k pages in order to minimize the number of page faults. We develop the *marking algorithm*, a randomized on-line algorithm for the paging problem. We prove that its expected cost on any sequence of requests is within a factor of $2H_k$ of optimum. (Where H_k is the kth harmonic number, which is roughly $\ln k$.) The best such factor that can be achieved is H_k. This is in contrast to deterministic algorithms, which cannot be guaranteed to be within a factor smaller than k of optimum. An alternative to comparing an on-line algorithm with the optimum off-line algorithm is the idea of comparing it to several other on-line algorithms. We have obtained results along these lines for the paging problem. Given a set of on-line algorithms

---

[1] Support was provided by a Weizmann fellowship.

[2] Partial support was provided by the International Computer Science Institute, Berkeley, CA, and by NSF Grant CCR-8411954.

[3] Support was provided by the International Computer Science Institute and operating grant A8092 of the Natural Sciences and Engineering Research Council of Canada. Current address: International Computer Science Institute, Berkeley, CA 94704.

[4] Partial support was provided by DARPA, ARPA order 4976, Amendment 20, monitored by the Air Force Avionics Laboratory under Contract F33615-87-C-1499, and by the National Science Foundation under Grant CCR-8658139.

[5] Part of this work was performed while the author was at the Digital Equipment Corp. Systems Research Center, Palo Alto, CA.

0196-6774/91 $3.00

and a set of appropriate constants, we describe a way of constructing another on-line algorithm whose performance is within the appropriate constant factor of each algorithm in the set.

1. INTRODUCTION

Consider a memory system with k pages of fast memory (a cache) and $n - k$ pages of slow memory. A sequence of requests to pages is to be satisfied, and in order to satisfy a request to a page that page must be in fast memory. If a requested page is not in fast memory a *page fault* occurs. In this case the requested page must be moved into fast memory, and (usually) a page must be moved from fast memory to slow memory to make room for the requested page. The *paging problem* is that of deciding which page to eject from fast memory. The cost to be minimized is the number of page faults.

A paging algorithm is said to be *on-line* if its decision of which page to eject from fast memory is made without knowledge of future requests. Sleator and Tarjan [15] analyzed on-line paging algorithms by comparing their performance on any sequence of requests to that of the optimum *off-line* algorithm (that is, one that has knowledge of the entire sequence of requests in advance). They showed that two strategies for paging (ejecting the least recently used page, or LRU, and first-in-first-out, or FIFO) could be worse than the optimum off-line algorithm by a factor of k, but not more, and that no on-line algorithm could achieve a factor less than k.

Karlin *et al.* [9] introduced the term *competitive* to describe an on-line algorithm whose cost is within a constant factor (independent of the request sequence) of the optimum off-line algorithm, and they used the term *strongly competitive* to describe an algorithm whose cost is within the smallest possible constant factor of optimum. These authors proposed another paging strategy, *flush-when-full* or FWF, and showed that it is also strongly k-competitive.

Manasse *et al.* [10] extended the definition of competitiveness to include randomized on-line algorithms (on-line algorithms which are allowed to use randomness in deciding what to do). Let A be a randomized on-line algorithm, let σ be a sequence of requests, and let $\overline{C_A}(\sigma)$ be the cost of algorithm A on sequence σ averaged over all the random choices that A makes while processing σ. Let $C_B(\sigma)$ be the cost of deterministic algorithm B on sequence σ. Algorithm A is said to be *c-competitive* if there is a constant a such that for every request sequence σ and every algorithm B:

$$\overline{C_A}(\sigma) \leq c \cdot C_B(\sigma) + a.$$

The constant c is known as the *competitive factor*. This definition has the desirable feature of ensuring that A's average performance on every individual sequence is close to that of the optimum off-line algorithm.

In this paper we consider randomized algorithms for the paging problem from the competitive point of view. We describe a randomized algorithm, called the *marking algorithm*, and show that it is $2H_k$-competitive. (Here H_k denotes that kth harmonic number: $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + 1/k$. This function is closely approximated by the natural logarithms: $\ln(k+1) \le H_k \le \ln(k) + 1$. We also show that no randomized paging algorithm can have a competitive factor less than H_k.

The marking algorithm is strongly competitive (its competitive factor is H_k) if $k = n - 1$, but it is not strongly competitive if $k < n - 1$. We describe another algorithm, EATR, which is strongly competitive for the case $k = 2$.

Borodin, Linial, and Saks [3] gave the first specific problem in which the competitive factor is reduced if the on-line algorithm is allowed to use randomness. The problem they analyzed is the *uniform task system*. They presented a randomized algorithm for uniform task systems whose competitive factor is $2H_n$, where n is the number of states in the task system, and proved that for this problem the competitive factor of any randomized algorithm is at least H_n. The marking algorithm is an adaptation of the randomized algorithm of Borodin *et al.* It was discovered by three groups working independently. These three groups collaborated in the writing of this paper.

The standard definition of competitiveness requires that the on-line algorithm be within a constant factor of any other algorithm, even an off-line one. In the case of deterministic paging algorithms this constraint is so severe that the best possible constant required is rather large. An alternative approach is to require that the on-line algorithm be *efficient* compared to several other on-line algorithms simultaneously. Given deterministic on-line algorithms for the paging problem $B(1), B(2), \ldots, B(m)$, and constants $c(1), c(2), \ldots, c(m)$, we show how to construct a new on-line algorithm whose performance is within a factor of $c(i)$ of $B(i)$ for all $1 \le i \le m$, under the condition that $1/c(1) + \cdots + 1/c(m) \le 1$. For example, we can construct an algorithm whose performance is within a factor of two of the performance of both the LRU algorithm and the FIFO algorithm. We also show how this construction can be applied to randomized algorithms.

This paper is organized as follows. Section 2 defines *server problems* (a generalized form of the paging problem) and introduces the terminology we shall use for the paging problem. Section 3 discusses the marking algorithm, Section 4 describes algorithm EATR, Section 5 proves the H_k lower bound on the competitive factor, and Section 6 contains our results about combining algorithms. Recent extensions to this work are described in Section 7, along with several open problems.

2. SERVER PROBLEMS

To put our work on paging in context it is useful to point out the connection between the paging problem and the *k-server problem*. Let G be an n-vertex graph with positive edge lengths obeying the triangle inequality, and let k mobile servers occupy vertices of G. Given a sequence of requests, each of which specifies a vertex that requires service, the k-server problem is to decide how to move the servers in response to each request. If a requested vertex is unoccupied, then some server must be moved there. The requests must be satisfied in the order of their occurrence in the request sequence. The cost of handling a sequence of requests is equal to the total distance moved by the servers.

Server problems were introduced by Manasse, McGeoch, and Sleator [10, 11]. They showed that no deterministic algorithm for the k-server problem can be better than k-competitive, they gave k-competitive algorithms for the case when $k = 2$ and $k = n - 1$, and they conjectured that there exists a k-competitive k-server algorithm for any graph. This conjecture holds when the graph is uniform [15], a weighted cache system (where the cost of moving to a vertex from anywhere is the same) [4], a line [4], or a tree [5]. Fiat *et al.* [7] showed that there is an algorithm for the k-server problem with a competitive factor that depends only on k. There has also been work on *memoryless* randomized algorithms for server problems [1, 6, 14]. These algorithms keep no information between requests except the server locations. The randomized algorithm of Coppersmith *et al.* [6] is k-competitive for a large class of graphs.

In the *uniform k-server problem* the cost of moving a server from any vertex to any other is one. The paging problem is isomorphic to the uniform k-server problem. The correspondence between the two problems is as follows: the pages of address space correspond to the n vertices of the graph, and the pages in fast memory correspond to those vertices occupied by servers. In the remainder of this paper we shall use the terminology of the uniform k-server problem.

3. THE MARKING ALGORITHM

The marking algorithm is a randomized algorithm for the uniform k-server problem on a graph with n vertices. The algorithm works as follows: The servers are initially on vertices $1, 2, 3, \dots, k$. The algorithm