On the Influence of Lookahead in Competitive Paging Algorithms

Susanne Albers*

Abstract

We introduce a new model of lookahead for on-line paging algorithms and study several algorithms using this model. A paging algorithm is *on-line with strong lookahead l* if it sees the present request and a sequence of future requests that contains l pairwise distinct pages. We show that strong lookahead has practical as well as theoretical importance and improves the competitive factors of on-line paging algorithms. This is the first model of lookahead having such properties. In addition to lower bounds we present a number of deterministic and randomized on-line paging algorithms with strong lookahead which are optimal or nearly optimal.

Keywords: On-Line Algorithms, Paging, Lookahead, Competitive Analysis.

1 Introduction

In recent years, the competitive analysis of on-line algorithms has received much attention. Among on-line problems, the *paging problem* is of fundamental interest. Consider a two-level memory system which has a fast memory that can store k pages and a slow memory that can manage, basically, an unbounded number of pages. A sequence of requests to pages in the memory system must be served by a paging algorithm. A request is served if the corresponding page is in fast memory. If the requested page is not stored in fast memory, a *page fault* occurs. Then a page must be evicted from fast memory so that the requested page can be loaded into the vacated location. A paging algorithm specifies which page to evict on a fault. The cost incurred by a paging algorithm equals the number of page faults. A paging algorithm is *on-line* if it determines which page to evict on a fault without knowledge of future requests.

We analyze the performance of on-line paging algorithms using *competitive analysis* [14, 10]. In a competitive analysis, the cost incurred by an on-line algorithm is compared to the cost incurred by an *optimal off-line* algorithm. An optimal off-line algorithm knows the entire request sequence in advance and can serve it with minimum cost. Let $C_A(\sigma)$ and $C_{OPT}(\sigma)$ be the cost

*Address: Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Email: albers@mpi-sb.mpg.de. This work was done while the author was a student at the Graduiertenkolleg Informatik, Universität des Saarlandes, and was supported by a graduate fellowship of the Deutsche Forschungsgemeinschaft.

of the on-line algorithm A and the optimal off-line algorithm OPT on request sequence σ. Then the algorithm A is c-competitive, if there exists a constant a such that

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + a$$

for all request sequences σ. The *competitive factor* of A is the infimum of all c such that A is c-competitive. If A is a randomized algorithm, then $C_A(\sigma)$ is the expected cost incurred by A on request sequence σ. In this paper we evaluate the performance of randomized on-line algorithms only against the *oblivious adversary* (see [3] for details). An optimal off-line paging algorithm has been exhibited by Belady [2]. The algorithm is also called the MIN algorithm. On a fault, MIN evicts the page whose next request occurs farthest in the future.

The paging problem (without lookahead) has been studied intensively. Sleator and Tarjan [14] have demonstrated that the well-known replacement algorithms LRU (Least Recently Used) and FIFO (First-In First-Out) are k-competitive. On a fault LRU removes the page that was requested least recently, and FIFO evicts the page that has been in fast memory longest. Sleator and Tarjan have also proved that no on-line paging algorithm can be better than k-competitive; hence LRU and FIFO achieve the best competitive factor. Fiat *et al.* [5] have shown that no randomized on-line paging algorithm can be better than $H(k)$-competitive against an oblivious adversary. Here $H(k) = \sum_{i=1}^{k} 1/i$ denotes the kth harmonic number. They have also given a simple replacement algorithm, called the MARKING algorithm, which is $2H(k)$-competitive. McGeoch and Sleator [11] have proposed a more complicated randomized paging algorithm which achieves a competitive factor of $H(k)$.

In this paper we study the problem of *lookahead* in on-line paging algorithms. An important question is, what improvement can be achieved in terms of competitiveness, if an on-line algorithm knows not only the present request to be served, but also some future requests. This issue is fundamental from the practical as well as the theoretical point of view. In paging systems some requests usually wait in line to be processed by a paging algorithm. One reason is that requests do not necessarily arrive one after the other, but rather in blocks of possibly variable size. Furthermore, if several processes run on a computer, it is likely that some of them incur page faults which then wait for service. Many memory systems are also equipped with prefetching mechanisms, i.e., on a request not only the currently accessed page but also some related pages which are expected to be asked next are demanded to be in fast memory. Thus each request generates a number of additional requests. In fact, some paging algorithms used in practice make use of lookahead [15]. In the theoretical context a natural question is: What is it worth to know a part of the future?

Previous research on lookahead in on-line algorithms has mostly addressed dynamic location problems and on-line graph problems [4, 9, 7, 6, 8]; only very little is known in the area of competitive paging with lookahead. Consider the intuitive model of lookahead, which we call *weak lookahead.* Let $l \geq 1$ be an integer. We say that an on-line paging algorithm has a *weak lookahead of size l* if it sees the present request to be served and the next l future requests. It is well known that this model cannot improve the competitive factors of on-line paging algorithms. If an on-line paging algorithm has a weak lookahead of size l, then an adversary that constructs a request sequence can simply replicate each request l times in order to make the lookahead useless. The only other result known on competitive paging with lookahead has been developed by Young [17]. According to Young, a paging algorithm is *on-line with a resource-bounded lookahead of size l* if it sees the present request and the maximal sequence of future requests for which it will incur l faults. Young presents deterministic and randomized on-line paging algorithms with resource-bounded lookahead l which are $\max\{2k/l, 2\}$-competitive and $2(\ln(k/l) + 1)$-competitive, respectively. However, the model of resource-bounded lookahead is unrealistic in practice. A subsequence on which an algorithm incurs l faults can be very long. Moreover, the sequence of future requests to be seen by an on-line algorithm depends on the algorithm's behavior on past requests.

We now introduce a new model of lookahead which has practical as well as theoretical importance. As we shall see, this model can improve the competitive factors of on-line paging algorithms. Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence of length m. $\sigma(t)$ denotes the request at time t. For a given set S, $card(S)$ denotes the cardinality of S. Let $l \geq 1$ be an integer.

Strong lookahead of size l: The on-line algorithm sees the present request and a sequence of future requests. This sequence contains l pairwise distinct pages which also differ from the page requested by the present request. More precisely, when serving request $\sigma(t)$, the algorithm knows requests $\sigma(t+1), \sigma(t+2), \ldots, \sigma(t')$, where $t' = \min\{s > t | card(\{\sigma(t), \sigma(t+1), \ldots, \sigma(s)\}) = l+1\}$. The requests $\sigma(s)$, with $s \geq t' + 1$, are not seen by the on-line algorithm at time t.

Strong lookahead is motivated by an analysis of request sequences that occur in practice: Subsequences of consecutive requests generally contain a number of distinct pages. This observation is supported by simulations that we have done using the ATM address traces by Agarwal *et al.* [1]. From a theoretical point of view we require an adversary to reveal some really significant information on future requests. Although an adversary may replicate requests in the lookahead, an on-line algorithm is provided with relevant information about the future.

In the following, we always assume that an on-line algorithm has a strong lookahead of fixed size $l \geq 1$. If a request sequence $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ is given, then for all $t \geq 1$ we define a value $\lambda(t)$. If $card(\{\sigma(t), \sigma(t+1), \ldots, \sigma(m)\}) < l + 1$ then let $\lambda(t) = m$; otherwise let $\lambda(t) = \min\{t' > t | card(\{\sigma(t), \sigma(t+1), \ldots, \sigma(t')\}) = l + 1\}$. The *lookahead $L(t)$ at time t* is defined as

$$L(t) = \{\sigma(s) | s = t, t+1, \ldots, \lambda(t)\}.$$

We say that *a page x is in the lookahead at time t* if $x \in L(t)$.

The remainder of this paper is an in-depth study of paging with strong lookahead. Strong lookahead is the first realistic model of lookahead that also reduces the competitive factors of on-line paging algorithms. In Section 2 we consider deterministic on-line algorithms and present a variant of the algorithm LRU that, given a strong lookahead of size l, where $l \leq k - 2$, achieves a competitive factor of $(k-l)$. We also show that no deterministic on-line paging algorithm with strong lookahead l, $l \leq k-2$, can be better than $(k-l)$-competitive. Thus our proposed algorithm is optimal. Furthermore, we give another variant of the algorithm LRU with strong lookahead l which is $(k-l+1)$-competitive and hence almost optimal. Interestingly, this algorithm does not exploit full lookahead but rather serves the request sequence in a series of blocks. Thus the algorithm takes into account that in practice, requests often arrive in blocks. Section 3 addresses randomized on-line paging algorithms with strong lookahead. We prove that a modification of the MARKING algorithm with strong lookahead l, $l \leq k-2$, is $2H(k-l)$-competitive. This competitiveness is within a factor of 2 of optimal. In particular, we show that no randomized on-line paging algorithm with strong lookahead l, $l \leq k-2$, can be better than $H(k-l)$-competitive. Furthermore we present an extremely simple randomized on-line paging algorithm with strong lookahead l, which is $(k-l+1)$-competitive.

2 Deterministic paging with strong lookahead

Unless otherwise stated, we assume in the following that all our paging algorithms are *lazy* algorithms, i.e., they only evict a page on a fault.

Let $k \geq 3$. We consider the important case that an on-line paging algorithm has a strong lookahead of size $l \leq k-2$. The on-line paging algorithms we present are extensions of the algorithm LRU to our model of strong lookahead.

Algorithm LRU(l): On a fault execute the following steps. Among the pages in fast memory which are not contained in the present lookahead, determine the page whose last request occurred least recently. Evict this page and load the requested page.

Theorem 1 *Let $l \leq k-2$. The algorithm LRU(l) with strong lookahead l is $(k-l)$-competitive.*

Now we prove this theorem. Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence of length m. We assume without loss of generality that LRU(l) and OPT start with an empty fast memory and that on the first k faults, both LRU(l) and OPT load the requested page into the fast memory. Furthermore we assume that σ contains at least $l+1$ distinct pages. The following proof consists of three main parts. First, we introduce the potential function we use to analyze LRU(l). In the second part, we partition the request sequence σ into a series of phases and then, in the third part, we bound LRU(l)'s amortized cost using that partition.

1. The potential function

We introduce some basic notations. For $t = 1, 2, \ldots, \lambda(1)-1$, let $\mu(t) = 1$ and for $t = \lambda(1), \lambda(1)+1, \ldots, m$, let

$$\mu(t) = \max\{t' < t | card(\{\sigma(t'), \sigma(t'+1), \ldots, \sigma(t)\}) = l+1\}.$$

For $t \geq \lambda(1)$, $\mu(t)$ is the most recent point of time such that the subsequence $\sigma(\mu(t)), \sigma(\mu(t)+1), \ldots, \sigma(t)$ contains $l+1$ distinct pages. Define

$$M(t) = \{\sigma(s) | s = \mu(t), \mu(t)+1, \ldots, t\}.$$

For a given time t, the set $M(t)$ contains the last $l+1$ requested pages.

For $t = 1, 2, \ldots, m$, let $S_{LRU(l)}(t)$ be the set of pages contained in LRU(l)'s fast memory after request t, and let $S_{OPT}(t)$ be the set of pages contained in OPT's fast memory after request t. $S_{LRU(l)}(0)$ and $S_{OPT}(0)$ denote the sets of pages which are initially in fast memory, i.e., $S_{LRU(l)}(0) = S_{OPT}(0) = \emptyset$. For the analysis of the algorithm we assign weights to all pages. These weights are updated after each request. Let $w(x,t)$ denote the weight of page x after request t, $1 \leq t \leq m$. The weights are set as follows. If $x \notin S_{LRU(l)}(t)$ or $x \in L(t)$, then

$$w(x,t) = 0.$$

Let $j = card(S_{LRU(l)}(t) \setminus L(t))$. Assign integer weights from the range $[1,j]$ to the pages in $S_{LRU(l)}(t) \setminus L(t)$ such that any two pages $x, y \in S_{LRU(l)}(t) \setminus L(t)$ satisfy

$$w(x,t) < w(y,t)$$

iff the last request to x occurred earlier than the last request to y. For $t = 1, 2, \ldots, m$, let

$$S(t) = S_{LRU(l)}(t) \setminus \{M(t) \cup L(t) \cup S_{OPT}(t)\}.$$

We now define the potential function:

$$\Phi(t) = \sum_{x \in S(t)} w(x,t).$$

Intuitively, $S_{LRU(l)}(t) \setminus S_{OPT}(t)$ contains those pages which cause LRU(l) to have a higher cost than OPT. Instead of the pages $x \in S_{LRU(l)}(t) \setminus S_{OPT}(t)$, OPT can store pages in its fast memory which are not contained in $S_{LRU(l)}(t)$ but are requested in the future. Pages in $S_{LRU(l)}(t) \setminus S_{OPT}(t)$ which are contained in $L(t) \cup M(t)$ do not contribute to $\Phi(t)$. A page x in $S_{LRU(l)}(t) \setminus S_{OPT}(t)$ with $x \in L(t)$ cannot increase LRU(l)'s cost because x is requested in the near future. By neglecting the pages in $M(t)$ we can establish the property that each page can only cause an increase in potential of at most $k-l-1$, cf. Lemma 2. The weight $w(x,t)$ of a page $x \in S(t)$ equals the number of faults that LRU(l) must incur before it can evict x.

2. The partitioning of the request sequence

We will partition the request sequence σ into phases, numbered from 0 to p for some p, such that phase 0 contains at most $l+1$ distinct pages and phase i, $i = 1, 2, \ldots, p$, has the following two properties. Let t_i^b and t_i^e denote the beginning and the end of phase i, respectively.

Property 1: Phase i contains exactly $l+1$ distinct pages, i.e.,

$$card(\{\sigma(t_i^b), \sigma(t_i^b+1), \ldots, \sigma(t_i^e)\}) = l+1.$$

Property 2: For all $x \in S_{LRU(l)}(t_{i-1}^e) \setminus \{L(t_i^b) \cup S_{OPT}(t_{i-1}^e)\}$,

$$w(x, t_i^e) \leq k-l-2.$$

Property 2 will be crucial when bounding LRU(l)'s amortized cost.

In the following, we describe how to decompose σ. We assume that LRU(l) and OPT have already served σ. We partition the request sequence starting at the end of σ. Suppose that we have already constructed phases $P(i+1), P(i+2), \ldots, P(p)$. We show how to generate phase $P(i)$. Let $t_i^e = t_{i+1}^b - 1$. (We let $t_p^e = m$ at the beginning of the decomposition.) Now set $t = \mu(t_i^e)$ and compute $S_{LRU(l)}(t-1) \setminus L(t)$. If $S_{LRU(l)}(t-1) \setminus L(t) \neq \emptyset$, then let y be the most recently requested page in $S_{LRU(l)}(t-1) \setminus L(t)$. We consider two cases. If $S_{LRU(l)}(t-1) \setminus L(t) = \emptyset$ or if $S_{LRU(l)}(t-1) \setminus L(t) \neq \emptyset$ and $y \in S_{OPT}(t-1)$, then let $t_i^b = t$ and call the i-th phase $P(i) = \sigma(t_i^b), \sigma(t_i^b+1), \ldots, \sigma(t_i^e)$ a type 1 phase. Otherwise (if $S_{LRU(l)}(t-1) \setminus L(t) \neq \emptyset$ and $y \notin S_{OPT}(t-1)$) let t', $t' < t$, be the time when OPT evicted page y most recently. Let $t_i^b = t'$ and call the i-th phase $P(i) = \sigma(t_i^b), \sigma(t_i^b+1), \ldots, \sigma(t_i^e)$ a type 2 phase. The detailed algorithm is given below.

```
1.  $i := m$;
2.  $t_i^e := m$;
3.  **repeat**
4.     $t := \mu(t_i^e)$;
5.     **if** $S_{LRU(l)}(t-1) \setminus L(t) \neq \emptyset$ **then**
6.        Let $y$ be the most recently requested page in $S_{LRU(l)}(t-1) \setminus L(t)$;
7.     **endif**;
8.     **if** $(S_{LRU(l)}(t-1) \setminus L(t) = \emptyset)$ or $(S_{LRU(l)}(t-1) \setminus L(t) \neq \emptyset$ and $y \in S_{OPT}(t-1))$ **then**
9.        Let $t_i^b = t$ and let $P(i) = \sigma(t_i^b), \sigma(t_i^b+1), \ldots, \sigma(t_i^e)$ be the $i$-th phase;
10.       Call $P(i)$ a type 1 phase;
11.    **else**
12.       Determine the largest $t' < t$, such that OPT evicts page $y$ at time $t'$;
13.       Let $t_i^b = t'$ and let $P(i) = \sigma(t_i^b), \sigma(t_i^b+1), \ldots, \sigma(t_i^e)$ be the $i$-th phase;
14.       Call $P(i)$ a type 2 phase;
15.    **endif**;
16.    $i := i - 1$;
17.    $t_i^e := t_{i+1}^b - 1$;
18. **until** $t_i^e = 0$;
19. Number the phases from 0 to $p$;
```

Lemma 1 *The partition generated above satisfies the following conditions.*

a) Phase $P(0)$ contains at most $l+1$ distinct pages.

b) Every phase $P(i)$, $1 \leq i \leq p$, has Property 1 and Property 2.

Proof: First we prove part a). We show that $P(0)$ is a type 1 phase. This immediately implies that $P(0)$ contains at most $l+1$ pages. If $P(0)$ was a type 2 phase, then OPT would evict a page on the first request $\sigma(1)$. However, this is impossible because initially the fast memories are empty and on the first k faults both LRU(l) and OPT load the requested page into the fast memory.

Now we prove part b) of the lemma. Consider an arbitrary phase $P(i)$, $1 \le i \le p$. Let $t = \mu(t_i^e)$. If $S_{LRU(l)}(t-1) \setminus L(t) \ne \emptyset$, then let y be the most recently requested page in $S_{LRU(l)}(t-1) \setminus L(t)$ and let t'', $t'' < t$, be the time when y was requested most recently. If $P(i)$ is a type 2 phase, then let t', $t' \le t-1$, be the time when OPT evicted y most recently. (Since $y \notin S_{OPT}(t-1)$, we have $t'' < t' \le t-1$.)

We show that $P(i)$ contains exactly $l+1$ pages. For a type 1 phase there is nothing to show. Suppose $P(i)$ is a type 2 phase. Then $t_i^b = t'$. Let $s \in [t', t-1]$ be arbitrary and let x be the page requested at time s. We need to show that x is requested in the interval $[t, t_i^e]$, i.e., that $x \in L(t)$. So assume $x \notin L(t)$. Then by the definition of y, $x \notin S_{LRU(l)}(t-1)$, i.e., x was evicted by LRU(l) at some time $s' \in [s+1, t-1]$. Since y was not evicted by LRU(l) at time s' and y's most recent request was at time $t'' < s$, we must have $y \in L(s') \subseteq \{\sigma(s'), \ldots, \sigma(t-1), \sigma(t), \ldots, \sigma(t_i^e)\} = \{\sigma(s'), \ldots, \sigma(t-1)\} \cup L(t)$. But $y \notin \{\sigma(s'), \ldots, \sigma(t-1)\}$ and $y \notin L(t)$, by the definition of t'' and y. Thus $x \notin L(t)$ is impossible. We conclude that $P(i)$ contains exactly $l+1$ distinct pages.

It remains to prove that $P(i)$ has Property 2. Consider an arbitrary page $x \in S_{LRU(l)}(t_{i-1}^e) \setminus \{L(t_i^b) \cup S_{OPT}(t_{i-1}^e)\}$. If $w(x, t_i^e) = 0$, then the property clearly holds. Therefore assume $w(x, t_i^e) \ge 1$. By Property 1, $L(t_i^b)$ contains all pages which are requested in $P(i)$. Since $w(x, t_i^e) \ge 1$, we have $x \in S_{LRU(l)}(t_i^e) \setminus L(t_i^e)$ and hence $x \notin L(t_i^b) \cup L(t_i^e) \supseteq L(s)$ for all $s \in [t_i^b, t_i^e]$. Thus, x was a candidate for eviction by LRU(l) throughout $P(i)$, but was not evicted. This implies immediately that all pages requested in $P(i)$, i.e. all pages in $L(t_i^b)$, also belong to $S_{LRU(l)}(t_i^e)$.

We next show that $y \in S_{LRU(l)}(t_i^e)$. If $P(i)$ has type 1, then $y \in S_{OPT}(t_{i-1}^e)$ and hence $y \ne x$. Furthermore, y was requested more recently than x. Since x was not evicted by LRU(l) during $P(i)$, we conclude $y \in S_{LRU(l)}(t_i^e)$. If $P(i)$ has type 2, then y was evicted by OPT at t_i^b and hence $y \in S_{OPT}(t_{i-1}^e)$. This implies $y \ne x$. Also, by the definition of y, $y \notin L(t) = L(t_i^b)$. Since y was requested more recently than x, it follows, as above, $y \in S_{LRU(l)}(t_i^e)$. We conclude $L(t_i^b) \cup \{y\} \subseteq S_{LRU(l)}(t_i^e)$ and hence we have identified $l+2$ pages in $S_{LRU(l)}(t_i^e)$ which, at time t_i^e, were requested later than x. At time t_i^e, each of these pages has a weight of 0 or a weight which is greater than that of x. Thus, $w(x, t_i^e) \le k - l - 2$. □

In the remainder of this proof it is not important how the partition of σ is constructed. We only use the fact that we have a partition satisfying Property 1 and Property 2.

3. Bounding LRU(l)'s amortized cost

Using the partition of σ generated above, we will evaluate LRU(l)'s amortized cost on σ. First we will bound the increase in potential $\sum_{t=1}^{m} \Phi(t) - \Phi(t-1)$. Then we will estimate LRU(l)'s actual cost in each phase of σ. For $t = 1, 2, \ldots, m$, let

$$N(t) = S(t) \setminus S(t-1).$$

We set $M(0) = L(0) = \emptyset$ and $S(0) = S_{LRU(l)}(0) \setminus \{M(0) \cup L(0) \cup S_{OPT}(0)\}$, which is used in the definition of $N(1) = S(1) \setminus S(0)$.

We present two lemmas which are crucial in analyzing the change in potential $\Phi(t)-\Phi(t-1)$, $1 \le t \le m$. Note that

$$\begin{aligned}\Phi(t)-\Phi(t-1) &= \sum_{x\in S(t)} w(x,t) - \sum_{x\in S(t-1)} w(x,t-1)\\ &= \sum_{x\in N(t)} w(x,t) + \sum_{x\in S(t-1)\cap S(t)} (w(x,t)-w(x,t-1)) - \sum_{x\in S(t-1)\setminus S(t)} w(x,t-1).\end{aligned}$$

Lemma 2 *Let $1 \le t \le m$. If $x \in N(t)$, then $w(x,t) \le k-l-1$.*

Proof: By the definition of $N(t)$, we have $x \in S_{LRU(l)}(t) \setminus \{M(t) \cup L(t) \cup S_{OPT}(t)\}$. Since $x \notin M(t)$, page x is not requested in the interval $[\mu(t), t]$ and hence $x \in S_{LRU(l)}(\mu(t)-1)$. We have $x \notin M(t) \cup L(t)$ which implies $x \notin L(s)$ for all s with $\mu(t) \le s \le t$. Thus, x has been a candidate for eviction by LRU(l) throughout the interval $[\mu(t), t]$, but was not evicted. It follows that all pages in $M(t)$ must be in $S_{LRU(l)}(t)$. Note that $M(t)$ contains $l+1$ pages because OPT does not evict a page before the $(k+1)$-st fault. At time t, all pages in $M(t)$ have a weight of 0 or a weight which is greater than $w(x,t)$. Thus $w(x,t) \le k-l-1$. □

Lemma 3 *Let $1 \le t \le m$ and $x \in S(t-1)\cap S(t)$. Then x's weight satisfies $w(x,t-1) \ge w(x,t)$. In particular, if LRU(l) incurs a fault at time t, then $w(x,t-1) > w(x,t)$.*

Proof: First we show $w(x,t-1) \ge w(x,t)$. Note that by the definition of $S(t-1)$ and $S(t)$, we have $x \in S_{LRU(l)}(t-1) \setminus L(t-1)$ and $x \in S_{LRU(l)}(t) \setminus L(t)$. Hence $w(x,t-1) \ge 1$ and $w(x,t) \ge 1$. In order to show $w(x,t-1) \ge w(x,t)$, it suffices to prove the following statements.

1) Let y, $y \ne x$, be a page which satisfies $w(y,t-1) = 0$ and $w(y,t) > 0$. Then $w(x,t) < w(y,t)$.

2) Let y, $y \ne x$, be a page which satisfies $w(y,t-1) > 0$ and $w(x,t-1) < w(y,t-1)$. Then $w(y,t) = 0$ or $w(x,t) < w(y,t)$.

We prove these statements. If a page y satisfies $w(y,t-1) = 0$ and $w(y,t) > 0$, then y must be requested at time $t-1$ (and evicted by OPT at time t). Thus, at time t, y has the highest weight among all pages in $S_{LRU(l)}(t) \setminus L(t)$. Hence $w(x,t) < w(y,t)$. Suppose a page y satisfies $w(y,t-1) > 0$ and $1 \le w(x,t-1) < w(y,t-1)$. This implies that at time $t-1$, x's most recent request is longer ago than y's most recent request. We conclude that this statement must also hold at time t because x is not requested at time t. Thus, if $w(y,t) > 0$, then $w(x,t) < w(y,t)$. This completes the proof that $w(x,t-1) \ge w(x,t)$.

Now suppose that LRU(l) incurs a fault at time t. Then, at time t, LRU(l) evicts a page z, $z \ne x$, whose last request occurred earlier than x's last request. Hence $1 \le w(z,t-1) < w(x,t-1)$. Since the statements 1) and 2) hold, x's weight must decrease after z is evicted, i.e., $w(x,t-1) > w(x,t)$. □

Lemma 2 implies that at any time t, $1 \le t \le m$, a page $x \in N(t)$ can cause an increase in potential of at most $k-l-1$. Thus, for every t, $1 \le t \le m$, we have

$$\Phi(t)-\Phi(t-1) = (k-l-1)card(N(t)) - W(t), \tag{1}$$

where $W(t) = W^1(t) + W^2(t) + W^3(t)$ and

$$\begin{aligned} W^1(t) &= \sum_{x \in N(t)} (k - l - 1 - w(x,t)) \\ W^2(t) &= \sum_{x \in S(t-1) \cap S(t)} (w(x,t-1) - w(x,t)) \\ W^3(t) &= \sum_{x \in S(t-1) \setminus S(t)} w(x,t-1). \end{aligned}$$

For all $t = 1, 2, \ldots, m$, we have

$$W^1(t) \geq 0,\ W^2(t) \geq 0,\ W^3(t) \geq 0. \tag{2}$$

Clearly, $W^3(t) \geq 0$. The inequalities $W^1(t) \geq 0$ and $W^2(t) \geq 0$ follow from Lemma 2 and Lemma 3, respectively.

Next we estimate $\sum_{t=1}^{m} card(N(t))$ and derive a bound on $\sum_{t=1}^{m} \Phi(t) - \Phi(t-1)$. To each element $x \in N(t)$ we assign the most recent eviction of x by OPT. More formally, let

$$X = \{(x,t) \in (\bigcup_{t=1}^{m} N(t)) \times [1,m] | x \in N(t)\}.$$

We define a function $f : X \longrightarrow [1,m]$. For $(x,t) \in X$ we define

$$f(x,t) = \max\{s \leq t | \text{OPT evicts page } x \text{ at time } s\}.$$

Note that f is well-defined.

We prove two properties of the function f. Part b) of the following lemma will be useful when bounding LRU(l)'s actual cost in each phase of σ.

Lemma 4 *a) The function f is injective.*

b) Let $(x,t) \in X$ and $f(x,t) = t'$. Let $t \in [t_i^b, t_i^e]$, $0 \leq i \leq p$. If $i = 0$, then $t' \in [t_0^b, t_0^e]$. If $i \geq 1$, then $t' \in [t_{i-1}^b, t_i^e]$.

Proof: First we prove part a). Consider two distinct elements $(x,t_1) \in X$ and $(y,t_2) \in X$. We show that $f(x,t_1) \neq f(y,t_2)$. If $x \neq y$, then there is nothing to prove. So assume $x = y$ and let $t_1 < t_2$. We have $x \in N(t_1)$ and $x \in N(t_2)$. Thus

$$x \in S(t_1) = S_{LRU(l)}(t_1) \setminus \{M(t_1) \cup L(t_1) \cup S_{OPT}(t_1)\},$$

$$x \in S(t_2) = S_{LRU(l)}(t_2) \setminus \{M(t_2) \cup L(t_2) \cup S_{OPT}(t_2)\}$$

and

$$x \notin S(t_2 - 1) = S_{LRU(l)}(t_2 - 1) \setminus \{M(t_2 - 1) \cup L(t_2 - 1) \cup S_{OPT}(t_2 - 1)\}.$$

Since $x \in S_{LRU(l)}(t_2) \setminus \{M(t_2) \cup L(t_2)\}$, we have $x \in S_{LRU(l)}(t_2 - 1) \setminus L(t_2 - 1)$. This implies $x \in M(t_2 - 1) \cup S_{OPT}(t_2 - 1)$ because $x \notin S(t_2 - 1)$. Note that $x \notin M(t_1) \cup S_{OPT}(t_1)$. We conclude that page x must be requested at some time $t \in [t_1+1, t_2-1]$. Hence, OPT must evict x at some time $t' \in [t_1+2, t_2]$. Thus $f(x,t_1) < f(x,t_2)$.

Now we show part b) of the lemma. Note that $t' = f(x,t) \le t \le t_i^e$. If $i = 0$ or $i = 1$, then $t' \in [t_0^b, t_0^e]$ or $t' \in [t_0^b, t_1^e]$, respectively, and part b) is proved. So suppose $i \ge 2$. We assume $t' < t_{i-1}^b$ and show that this assumption implies $x \in S(t-1)$. This is a contradiction because $x \in N(t) = S(t) \setminus S(t-1)$. By the definition of t', the page x is not requested in the interval $[t', t]$ and $x \notin S_{OPT}(t')$. It follows $x \notin S_{OPT}(s)$ for all $s \in [t', t]$. Since $x \in S(t)$, we have $x \notin L(t)$. Thus, for all $s \in [t', t]$, x is not contained in $\{\sigma(s), \sigma(s+1), \ldots, \sigma(t)\} \cup L(t) \supseteq L(s)$. By Property 1, phase $P(i-1)$ contains $l+1$ distinct pages. Since x is not requested in the interval $[t', t]$ and $t' < t_{i-1}^b < t_{i-1}^e < t$, it follows $x \notin M(s)$ for all $s \in [t_{i-1}^e, t]$. Note that $x \in S_{LRU(l)}(t-1)$ because $x \in S(t) \subseteq S_{LRU(l)}(t)$ and x is not requested at time t. We conclude that

$$x \in S_{LRU(l)}(t-1) \setminus \{M(t-1) \cup L(t-1) \cup S_{OPT}(t-1)\} = S(t-1).$$

We obtain a contradiction because $x \in N(t)$. Thus $t' \ge t_{i-1}^b$. The proof of the lemma is complete. $\Box$

Let T_{OPT} be the set of all $t \in [1, m]$ such that OPT evicts a page at time t. Note that $C_{OPT}(\sigma) = card(T_{OPT})$. Let $T_{OPT}^1 = \{f(x,t) | (x,t) \in X\}$. By Lemma 4, f is injective and hence

$$\sum_{t=1}^{m} card(N(t)) = card(X) = card(T_{OPT}^1).$$

Thus, by equation (1), we obtain

$$\sum_{t=1}^{m} \Phi(t) - \Phi(t-1) = (k-l-1) card(T_{OPT}^1) - \sum_{t=1}^{m} W(t). \qquad (3)$$

Now we bound LRU(l)'s actual cost in each phase of σ. For $i = 0, 1, \ldots, p$, let $C_{LRU(l)}(i)$ be the actual cost LRU(l) incurs in serving phase $P(i)$, and let $C_{OPT}(i)$ be the cost OPT incurs in serving $P(i)$. Furthermore, let

$$T_{OPT}^2 = T_{OPT} \setminus T_{OPT}^1$$

and, for $i = 0, 1, \ldots, p$, let

$$T_{OPT}^2(i) = \{t \in T_{OPT}^2 | t_i^b \le t \le t_i^e\}.$$

Lemma 5 *a)* $C_{LRU(l)}(0) = C_{OPT}(0)$

b) For $i = 1, 2, \ldots, p$,

$$C_{LRU(l)}(i) \le C_{OPT}(i) + card(T_{OPT}^2(i-1)) + \sum_{t=t_i^b}^{t_i^e} W(t).$$

Proof: First we prove part a) of the lemma. Phase $P(0)$ contains at most $l+1$ distinct pages. On the first $k \geq l+1$ faults, both LRU(l) and OPT load the requested page into fast memory. Thus, during $P(0)$, LRU(l) and OPT incur the same cost, i.e.,

$$C_{LRU(l)}(0) = C_{OPT}(0).$$

In the proof of part b), we consider a fixed $i \in [1,p]$. If $C_{LRU(l)}(i) = 0$, then the inequality clearly holds because, by line (2), $W(t) \geq 0$ for all $t \in [t_i^b, t_i^e]$. So suppose $C_{LRU(l)}(i) \geq 1$. Let

$$\tilde{C}(i) = card(S_{LRU(l)}(t_{i-1}^e) \setminus \{L(t_i^b) \cup S_{OPT}(t_{i-1}^e)\}).$$

In the following we prove that the inequalities

$$C_{LRU(l)}(i) \leq C_{OPT}(i) + \tilde{C}(i) \tag{4}$$

and

$$\tilde{C}(i) \leq card(T_{OPT}^2(i-1)) + \sum_{t=t_i^b}^{t_i^e} W(t) \tag{5}$$

hold. These two inequalities imply part b).

First we prove inequality (4). At the beginning of phase $P(i)$, there are $card(S_{LRU(l)}(t_{i-1}^e) \cap S_{OPT}(t_{i-1}^e))$ pages which are contained in LRU(l)'s as well as by OPT's fast memory. Hence, at the beginning of $P(i)$, OPT's fast memory contains at most $card(S_{OPT}(t_{i-1}^e) \setminus S_{LRU(l)}(t_{i-1}^e)) = card(S_{LRU(l)}(t_{i-1}^e) \setminus S_{OPT}(t_{i-1}^e))$ pages which are in $L(t_i^b)$ but which are not contained in LRU(l)'s fast memory. Thus, during phase $P(i)$, LRU(l) incurs at most

$$card(S_{LRU(l)}(t_{i-1}^e) \setminus S_{OPT}(t_{i-1}^e)) - card((S_{LRU(l)}(t_{i-1}^e) \setminus S_{OPT}(t_{i-1}^e)) \cap L(t_i^b)) =$$
$$card(S_{LRU(l)}(t_{i-1}^e) \setminus \{L(t_i^b) \cup S_{OPT}(t_{i-1}^e)\})$$

faults more than OPT, i.e.,

$$C_{LRU(l)}(i) \leq C_{OPT}(i) + \tilde{C}(i).$$

Next we prove inequality (5). We introduce some notations. Let $t \in [t_i^b, t_i^e]$. For $x \in N(t)$ let

$$W^1(x,t) = k - l - 1 - w(x,t).$$

For $x \in S(t-1) \cap S(t)$ let

$$W^2(x,t) = w(x,t-1) - w(x,t)$$

and for $x \in S(t-1) \setminus S(t)$ let

$$W^3(x,t) = w(x,t-1).$$

Note that

$$\begin{aligned} W^1(t) &= \sum_{x \in N(t)} W^1(x,t) \\ W^2(t) &= \sum_{x \in S(t-1) \cap S(t)} W^2(x,t) \\ W^3(t) &= \sum_{x \in S(t-1) \setminus S(t)} W^3(x,t). \end{aligned}$$

For any $x \in N(t)$ ($x \in S(t-1) \cap S(t)$, $x \in S(t-1) \setminus S(t)$) we have

$$W^1(x,t) \geq 0 \quad (W^2(x,t) \geq 0,\ W^3(x,t) \geq 1). \tag{6}$$

The inequality $W^1(x,t) \geq 0$ follows from Lemma 2. Lemma 3 implies $W^2(x,t) \geq 0$. If $x \in S(t-1) \setminus S(t)$, then $x \in S_{LRU(l)}(t-1) \setminus L(t-1)$ and hence $1 \leq w(x,t-1) = W^3(x,t)$.

We sketch the main idea of the proof of inequality (5). We show that for each page $x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{L(t^b_i) \cup S_{OPT}(t^e_{i-1})\}$ one of the following two statements holds.

1) There exists a $t' \in T^2_{OPT}(i-1)$ such that OPT evicts page x at time t'.

2) There exists a time $t' \in [t^b_i, t^e_i]$ and a $j \in \{1,2,3\}$ such that $W^j(x,t') \geq 1$.

These statements, together with line (6), imply the correctness of inequality (5).

Consider a page $x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{L(t^b_i) \cup S_{OPT}(t^e_{i-1})\}$. We distinguish between two main cases.

Case 1: For $t = t^e_{i-1}, t^b_i, t^b_i + 1, \ldots, t^e_i$, $x \notin S(t)$

We prove that statement 1) holds. Since $x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{L(t^b_i) \cup S_{OPT}(t^e_{i-1})\}$, we have $x \notin S_{OPT}(t^e_{i-1})$. Let $t' = \max\{s \leq t^e_{i-1} | \text{OPT evicts page } x \text{ at time } s\}$. In the following we show that $t' \geq t^b_{i-1}$ and $t' \notin T^1_{OPT}$, which implies $t' \in T^2_{OPT}(i-1)$. If $i = 1$, then $t' \geq 1 = t^b_0 = t^b_{i-1}$. So let $i \geq 2$ and suppose $t' < t^b_{i-1}$. By the definition of t', x is not requested in the interval $[t', t^e_{i-1}]$. Thus, x is not contained in phase $P(i-1)$. By Property 1, phase $P(i-1)$ contains $l+1$ distinct pages, and hence $x \notin M(t^e_{i-1})$. We have $x \neq \sigma(t^e_{i-1})$ and $x \notin L(t^b_i)$, which implies $x \notin L(t^e_{i-1})$. Since $x \in S_{LRU(l)}(t^e_{i-1}) \setminus S_{OPT}(t^e_{i-1})$, we conclude

$$x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{M(t^e_{i-1}) \cup L(t^e_{i-1}) \cup S_{OPT}(t^e_{i-1})\} = S(t^e_{i-1}).$$

We obtain a contradiction because $x \notin S(t)$ for all $t \in [t^e_{i-1}, t^e_i]$. Thus $t' \geq t^b_{i-1}$.

Next we show $t' \notin T^1_{OPT}$. We have to prove that there exists no pair $(x,s) \in X$ satisfying $t' \leq s \leq m$ and $f(x,s) = t'$. Assume that there is such a pair. Since $t' \leq t^e_{i-1}$, part b) of Lemma 4 implies $s \leq t^e_i$. We have $x \notin S(t)$ for all $t \in [t^e_{i-1}, t^e_i]$ and hence $x \notin N(t)$ for all $t \in [t^e_{i-1}, t^e_i]$. Thus $t' \leq s < t^e_{i-1}$. The page x is contained in $N(s)$, i.e., it is contained in $S(s)$. This implies $x \notin M(s)$. By the definition of t', x is not requested in $[t', t^e_{i-1}]$ and hence $x \notin M(t^e_{i-1})$. Since $x \neq \sigma(t^e_{i-1})$ and $x \notin L(t^b_i)$, we have $x \notin L(t^e_{i-1})$. Thus

$$x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{M(t^e_{i-1}) \cup L(t^e_{i-1}) \cup S_{OPT}(t^e_{i-1})\} = S(t^e_{i-1})$$

because $x \in S_{LRU(l)}(t^e_{i-1}) \setminus S_{OPT}(t^e_{i-1})$. As above, we obtain a contradiction.

Case 2: There exists a t, $t^e_{i-1} \leq t \leq t^e_i$, such that $x \in S(t)$
In this case we show that the above statement 2) holds. Let $t_{\min}$ be the smallest $t \in [t^e_{i-1}, t^e_i]$ such that $x \in S(t)$.

Case 2.1: $t_{\min} = t^e_{i-1}$
Let t'' be the time when LRU(l) incurs the first fault during phase $P(i)$. We consider $w(x, t'')$. If $w(x, t'') = 0$, then $x \notin S(t'')$. Hence there must exist a t', $t^b_i \leq t' \leq t''$, such that $x \in S(t'-1) \setminus S(t')$. Thus $W^3(x, t') \geq 1$. Now suppose $w(x, t'') \geq 1$. Then $x \in S_{LRU(l)}(t'') \setminus L(t'')$. We have $x \in S(t^e_{i-1})$. Since $x \notin L(t^b_i)$, the page is not requested in the interval $[t^b_i, t'']$. We easily verify that for all $s \in [t^e_{i-1}, t'']$, $x \in S_{LRU(l)}(s) \setminus S_{OPT}(s)$ and $x \notin M(s) \cup L(s)$. Thus $x \in S(t''-1) \cap S(t'')$. Now Lemma 3 implies $W^2(x, t'') \geq 1$.

Case 2.2: $t_{\min} > t^e_{i-1}$
If $w(x, t_{\min}) < k - l - 1$, then $W^1(x, t_{\min}) \geq 1$. Suppose $w(x, t_{\min}) = k - l - 1$. By Property 2, $w(x, t^e_i) \leq k - l - 2$. This implies that if $x \in S(s)$ for all $s \in [t_{\min}, t^e_i]$, then there must exist a time t', $t_{\min} < t' \leq t^e_i$, such that $W^2(x, t') \geq 1$. If $x \notin S(s)$ for some $s \in [t_{\min}, t^e_i]$, then there must exist a time $t' \in [t_{\min} + 1, t^e_i]$ with $x \in S(t'-1) \setminus S(t')$ and hence $W^3(x, t') \geq 1$.

The proof of Lemma 5 is complete. □

Using equation (3) and Lemma 5, it is easy to finish the proof of Theorem 1. We estimate LRU(l)'s amortized cost. By equation (3) we have

$$\begin{aligned} C_{LRU(l)}(\sigma) + \Phi(m) - \Phi(0) &= C_{LRU(l)}(\sigma) + \sum_{t=1}^{m} \Phi(t) - \Phi(t-1) \\ &= C_{LRU(l)}(\sigma) + (k-l-1)card(T^1_{OPT}) - \sum_{t=1}^{m} W(t). \end{aligned}$$

Lemma 5 implies that

$$\begin{aligned} C_{LRU(l)}(\sigma) + \Phi(m) - \Phi(0) &= \sum_{i=0}^{p} C_{LRU(l)}(i) - \sum_{t=1}^{m} W(t) + (k-l-1)card(T^1_{OPT}) \\ &\leq \sum_{i=0}^{p} C_{OPT}(i) + \sum_{i=0}^{p-1} card(T^2_{OPT}(i)) + \sum_{t=t^b_1}^{m} W(t) \\ &\quad - \sum_{t=1}^{m} W(t) + (k-l-1)card(T^1_{OPT}). \end{aligned}$$

Line (2) implies that $W(t) \geq 0$ for all $t \in [t^b_0, t^e_0]$. Hence

$$\begin{aligned} C_{LRU(l)}(\sigma) + \Phi(m) - \Phi(0) &\leq C_{OPT}(\sigma) + card(T^2_{OPT}) + (k-l-1)card(T^1_{OPT}) \\ &\leq C_{OPT}(\sigma) + (k-l-1)card(T_{OPT}) \\ &= (k-l)C_{OPT}(\sigma). \end{aligned}$$

The proof of Theorem 1 is complete.

Next we present another on-line algorithm with strong lookahead. This algorithm does not use full lookahead but rather serves the request sequence in a series of blocks.

Algorithm LRU(l)-blocked: Serve the request sequence in a series of blocks $B(1), B(2), \ldots$, where $B(1) = \sigma(1), \sigma(2), \ldots, \sigma(\lambda(1))$ and $B(i) = \sigma(t^e_{i-1}+1), \sigma(t^e_{i-1}+2), \ldots, \sigma(\lambda(t^e_{i-1}+1))$ for $i \geq 2$. Here t^e_{i-1} denotes the end of block $B(i-1)$. If there occurs a fault while $B(i)$ is processed, then the following rule applies. Among the pages in fast memory which are not contained in $B(i)$, determine the page whose last request occurred least recently. Evict that page.

LRU(l)-blocked has the advantage that it updates its information on future requests only once during each block. Thus it can respond to requests faster that LRU(l). Furthermore, LRU(l)-blocked takes into account that in practice requests often arrive in blocks. Interestingly, this simpler algorithm is only slightly weaker than LRU(l).

Theorem 2 *Let $l \leq k-2$. The algorithm LRU(l)-blocked with strong lookahead l is $(k-l+1)$-competitive.*

Proof: We assume that the request sequence consists of b blocks $B(1), B(2), \ldots, B(b)$. For $i = 1, 2, \ldots, b$, let t^b_i and t^e_i denote the beginning and the end of block $B(i)$, respectively. Again we assume that LRU(l)-blocked and OPT start with an empty fast memory. On the first k faults, both LRU(l)-blocked and OPT load the requested page into fast memory. We assume that σ contains at least $l+1$ distinct requests. The following proof is very similar to the proof of Theorem 1.

The potential function we use to analyze LRU(l)-blocked resembles the function we introduced in the proof of Theorem 1. For $t = 1, 2, \ldots, m$, the values $\mu(t)$ and the sets $M(t)$ are defined as in the previous proof. Let $S_{LRU(l)}(t)$ be the set of pages contained in LRU(l)-blocked's fast memory after request t, and let $S_{OPT}(t)$ be the set of pages contained in OPT's fast memory after request t, $1 \leq t \leq m$. $S_{LRU(l)}(0)$ and $S_{OPT}(0)$ denote the sets of pages which are initially in fast memory, i.e., $S_{LRU(l)}(0) = S_{OPT}(0) = \emptyset$. For $t = 1, 2, \ldots, m$, we define values $\gamma(t)$. Set $\gamma(t) = i$ iff $t^b_i \leq t \leq t^e_i$. Let $S_B(t)$ be the set of pages that are requested during block $B(\gamma(t))$.

Again, we assign weights to all pages. Let $w(x,t)$ denote the weight of page x after request t, $1 \leq t \leq m$. If $x \notin S_{LRU(l)}(t)$ or if $x \in S_B(t)$ then $w(x,t) = 0$. Let $j = card(S_{LRU(l)}(t) \setminus S_B(t))$. Assign integer weights from the range $[1, j]$ to the pages in $S_{LRU(l)}(t) \setminus S_B(t)$ such that

$$w(x,t) < w(y,t)$$

iff the last request to x occurred earlier than the last request to y.

For $t = 1, 2, \ldots, m$, let

$$S(t) = S_{LRU(l)}(t) \setminus \{M(t) \cup S_B(t) \cup S_{OPT}(t)\}.$$

The potential function is defined as

$$\Phi(t) = \sum_{x \in S(t)} w(x,t).$$

In the following, we evaluate LRU(l)-blocked's amortized cost on the request sequence σ. First we derive a bound on the increase in potential $\sum_{t=1}^{m} \Phi(t) - \Phi(t-1)$. Then we determine LRU($l$)-blocked's actual cost in each block of σ.

For $t = 1, 2, \ldots, m$ let

$$N(t) = S(t) \setminus S(t-1).$$

Again, we set $M(0) = S_B(0) = \emptyset$ and $S(0) = S_{LRU(l)}(0) \setminus \{M(0) \cup S_B(0) \cup S_{OPT}(0)\}$, which we need in the definition of $N(1) = S(1) \setminus S(0)$.

Lemma 6 *Let $1 \le t \le m$. If $x \in N(t)$, then $w(x,t) \le k - l - 1$.*

Proof: The definition of $N(t)$ implies $x \in S_{LRU(l)}(t) \setminus \{M(t) \cup S_B(t) \cup S_{OPT}(t)\}$ and hence $x \notin M(t)$. We show that all pages $y \in M(t)$ satisfy $y \in S_{LRU(l)}(t)$. Suppose $t \in [t_i^b, t_i^e]$, $1 \le i \le b$. Consider an arbitrary $y \in M(t)$. If y is requested in the interval $[t_i^b, t]$, then the definition of the algorithm LRU(l)-blocked implies $y \in S_{LRU(l)}(t)$. Suppose y is not requested in the interval $[t_i^b, t]$. Then $i \ge 2$ and y must be requested at some time $t' \in [\mu(t), t_{i-1}^e]$. Since block $B(i-1)$ contains $l+1$ distinct pages, we have $t_{i-1}^b \le \mu(t)$. Thus, by the definition of the algorithm LRU(l)-blocked, $y \in S_{LRU(l)}(t_{i-1}^e)$. We have $x \notin M(t)$, and hence x is not requested in the interval $[\mu(t), t]$. It follows that $x \in S_{LRU(l)}(\mu(t)-1)$ (because $x \in S_{LRU(l)}(t)$) and that at any time s, $t_i^b \le s \le t$, x's most recent request is longer ago than y's most recent request. Since $x \notin S_B(t)$, x is a candidate for eviction throughout the interval $[t_i^b, t]$, but is not evicted. Hence, y cannot be evicted in the interval $[t_i^b, t]$ and must be in $S_{LRU(l)}(t)$. The set $M(t)$ contains $l+1$ distinct pages, and each of these pages was requested more recently than x. Thus $w(x,t) \le k - l - 1$. □

Lemma 7 *Let $1 \le t \le m$ and $x \in S(t-1) \cap S(t)$. Then x's weight satisfies $w(x,t-1) \ge w(x,t)$. In particular, if LRU(l)-blocked incurs a fault at time t, then $w(x,t-1) > w(x,t)$.*

Proof: The lemma can be shown in the same way as Lemma 3. Only the proof of the statement 1) is slightly different: Consider a page y, $y \ne x$, which satisfies $w(y,t-1) = 0$ and $w(y,t) > 0$. Suppose $t \in [t_i^b, t_i^e]$, $1 \le i \le b$. The inequality $w(y,t) > 0$ implies that y is not requested in block $B(i)$. We have $y \in S_{LRU(l)}(t)$. Thus, $y \in S_{LRU(l)}(t-1)$ because y is not requested at time t. Hence, equation $w(y,t-1) = 0$ implies $y \in S_B(t-1)$, i.e., y is requested in the block which contains time $t-1$. It follows $i \ge 2$, $t = t_i^b$ and $t - 1 = t_{i-1}^e$. Note that x is neither requested in block $B(i-1)$ nor in block $B(i)$ because $w(x,t-1) \ge 1$ and $w(x,t) \ge 1$. We conclude that at time t, x's most recent request is longer ago than y's last request. Thus $w(x,t) < w(y,t)$. □

Lemma 6 implies that for every t, $1 \le t \le m$,

$$\Phi(t) - \Phi(t-1) = (k-l)card(N(t)) - W(t), \tag{7}$$

where

$$W(t) = W^1(t) + W^2(t) + W^3(t)$$

and

$$\begin{aligned} W^1(t) &= \sum_{x \in N(t)} (k - l - w(x,t)) \\ W^2(t) &= \sum_{x \in S(t-1) \cap S(t)} (w(x,t-1) - w(x,t)) \\ W^3(t) &= \sum_{x \in S(t-1) \setminus S(t)} w(x,t-1). \end{aligned}$$

For $t = 1, 2, \ldots, m$ we have

$$W^1(t) \geq 0, \; W^2(t) \geq 0, \; W^3(t) \geq 0. \tag{8}$$

Obviously, $W^3(t) \geq 0$. The inequalities $W^1(t) \geq 0$ and $W^2(t) \geq 0$ follow from Lemma 6 and Lemma 7, respectively.

Our next goal is to determine $\sum_{t=1}^m card(N(t))$ and to bound $\sum_{t=1}^m \Phi(t) - \Phi(t-1)$. We define a set X and a function f in exactly the same way as in the proof of Theorem 1.

Using a similar analysis as in the proof of Lemma 4 we are able to show

Lemma 8 *a) The function f is injective.*

b) Let $(x,t) \in X$ and $f(x,t) = t'$. Let $t \in [t_i^b, t_i^e]$, $1 \leq i \leq b$. If $i = 1$, then $t' \in [t_1^b, t_1^e]$. If $i \geq 2$, then $t' \in [t_{i-1}^b, t_i^e]$.

As in the proof of Theorem 1, let T_{OPT} be the set of all $t \in [1,m]$ such that OPT evicts a page at time t. Again, we have $C_{OPT}(\sigma) = card(T_{OPT})$. Let $T_{OPT}^1 = \{f(x,t) | (x,t) \in X\}$. Since the function f is injective (see Lemma 8), we have $\sum_{t=1}^m card(N(t)) = card(X) = card(T_{OPT}^1)$. Hence, equation (7) implies

$$\sum_{t=1}^m \Phi(t) - \Phi(t-1) = (k-l) card(T_{OPT}^1) - \sum_{t=1}^m W(t). \tag{9}$$

Next we bound LRU(l)'s actual cost in each block of σ. For $i = 1, 2, \ldots, b$, let $C_{LRU(l)}(i)$ be the actual cost LRU(l)-blocked incurs in serving block $B(i)$, and let $C_{OPT}(i)$ be the cost OPT incurs in serving $B(i)$. Let

$$T_{OPT}^2 = T_{OPT} \setminus T_{OPT}^1$$

and, for $i = 1, 2, \ldots, b$, let

$$T_{OPT}^2(i) = \{t \in T_{OPT}^2 | t_i^b \leq t \leq t_i^e\}.$$

Lemma 9 *a)* $C_{LRU(l)}(1) = C_{OPT}(1)$

b) For $i = 2, 3, \ldots, b$,

$$C_{LRU(l)}(i) \leq C_{OPT}(i) + card(T^2_{OPT}(i-1)) + \sum_{t=t^b_i}^{t^e_i} W(t).$$

Proof: Part a) of the lemma can be shown in the same way as the corresponding statement of Lemma 5. We prove part b). Fix an $i \in [2, b]$. If $C_{LRU(l)}(i) = 0$, then there is nothing to show because $W(t) \geq 0$ for all $t \in [t^b_i, t^e_i]$ (see line (8)). So assume $C_{LRU(l)}(i) \geq 1$. Let

$$\tilde{C}(i) = card(S_{LRU(l)}(t^e_{i-1}) \setminus \{S_B(t^b_i) \cup S_{OPT}(t^e_{i-1})\}).$$

During block $B(i)$, LRU(l)-blocked incurs at most $\tilde{C}(i)$ faults more than OPT, i.e.,

$$C_{LRU(l)}(i) \leq C_{OPT}(i) + \tilde{C}(i). \tag{10}$$

We show that the inequality

$$\tilde{C}(i) \leq card(T^2_{OPT}(i-1)) + \sum_{t=t^b_i}^{t^e_i} W(t) \tag{11}$$

holds. Inequalities (10) and (11) imply part b) of Lemma 9.

We need some more notations. Let $t \in [t^b_i, t^e_i]$. For $x \in N(t)$ let $W^1(x,t) = k - l - w(x,t)$. For $x \in S(t-1) \cap S(t)$ let $W^2(x,t) = w(x,t-1) - w(x,t)$ and for $x \in S(t-1) \setminus S(t)$ let $W^3(x,t) = w(x,t-1)$. We have

$$W^1(t) = \sum_{x \in N(t)} W^1(x,t), \quad W^2(t) = \sum_{x \in S(t-1) \cap S(t)} W^2(x,t), \quad W^3(t) = \sum_{x \in S(t-1) \setminus S(t)} W^3(x,t).$$

Furthermore, for a page $x \in N(t)$ ($x \in S(t-1) \cap S(t)$, $x \in S(t-1) \setminus S(t)$), we have

$$W^1(x,t) \geq 0 \;\; (W^2(x,t) \geq 0,\; W^3(x,t) \geq 1). \tag{12}$$

We prove inequality (11). Consider a page $x \in S_{LRU(l)}(t^e_{i-1}) \setminus \{S_B(t^b_i) \cup S_{OPT}(t^e_{i-1})\}$.

Case 1: For $t = t^e_{i-1}, t^b_i, t^b_i + 1, \ldots, t^e_i$, $x \notin S(t)$
Using the same analysis as in the proof of Lemma 5 we can show that there exists a time $t' \in T^2_{OPT}(i-1)$ such that OPT evicts a page at time t'.

Case 2: There exists a t, $t^e_{i-1}, \leq t \leq t^e_i$, such that $x \in S(t)$
We show that there exists a time $t' \in [t^b_i, t^e_i]$ and a $j \in \{1, 2, 3\}$ such that $W^j(x, t') \geq 1$. Let $t_{\min}$ be the smallest $t \in [t^e_{i-1}, t^e_i]$ such that $x \in S(t)$.

Case 2.1: $t_{\min} = t^e_{i-1}$
Let t'' be the time when LRU(l)-blocked incurs the first fault during block $B(i)$. Again, we can

apply the same analysis as in the proof of Lemma 5. If $w(x,t'')=0$, then we can show that there exists a time t', $t_i^b \leq t' \leq t_i^e$, such that $W^3(x,t') \geq 1$. If $w(x,t'') \geq 1$, then we can prove $W^2(x,t'') \geq 1$.

Case 2.2: $t_{\min} > t_{i-1}^e$
By Lemma 6, $w(x,t_{\min}) \leq k-l-1$. Hence $W^1(x,t_{\min}) \geq 1$.

The above case analysis, together with line (12), implies inequality (11). The proof of Lemma 9 is complete. □

Let $C_{LRU(l)}(\sigma)$ be the actual cost LRU(l) incurs in serving the request sequence σ. Using equation (9) and Lemma 9 we can easily prove

$$C_{LRU(l)}(\sigma) + \Phi(m) - \Phi(0) \leq (k-l+1)C_{OPT}(\sigma).$$

The proof of Theorem 2 is complete. □

The following theorem shows that LRU(l) and LRU(l)-blocked are optimal and nearly optimal, respectively.

Theorem 3 *Let A be a deterministic on-line paging algorithm with strong lookahead l, where $l \leq k-2$. If A is c-competitive, then $c \geq (k-l)$.*

Proof: Let $S = \{x_1, x_2, \ldots, x_{k+1}\}$ be a set of $k+1$ pages. We assume without loss of generality that $A's$ and OPT's fast memories initially contain $x_1, x_2, \ldots, x_k$. Let $SL = \{x_1, x_2, \ldots, x_l\}$. We construct a request sequence σ consisting of a series of phases. Each phase contains $l+1$ requests to $l+1$ distinct pages. The first phase $P(1)$ consists of requests to the pages in SL, followed by a request to the page x_{k+1} which is not in fast memory, i.e., $P(1) = x_1, x_2, \ldots, x_l, x_{k+1}$. Each of the following phases $P(i)$, $i \geq 2$, has the form $P(i) = x_1, x_2, \ldots, x_l, y_i$, where $y_i \in S \setminus SL$ is chosen as follows. Let $z_i \in S$ be the page which is not in A's fast memory after the last request of phase $i-1$. If $z_i \in S \setminus SL$, then set $y_i = z_i$. Otherwise, if $z_i \in SL$, y_i is an arbitrary page in $S \setminus SL$. The algorithm A incurs a cost of 1 in each phase. We show that during $k-l$ successive phases, OPT's cost is at most 1. This proves the theorem. OPT always keeps $x_1, x_2, \ldots, x_l$ in its fast memory. Note that $k-l$ successive phases contain at most k different pages. If OPT incurs a fault on the last request in a given phase, then OPT can evict a page such that all pages in the next $k-l-1$ phases remain in fast memory. □

So far, we have assumed $k \geq 3$ and $l \leq k-2$, which, of course, is the interesting case. Note that if $l = k-1$ and the total number of different pages in the memory system equals $k+1$, then LRU(l) achieves a competitive factor of 1 because it behaves like Belady's optimal paging algorithm MIN [2].

3 Randomized paging with strong lookahead

Suppose a randomized paging algorithm has a strong lookahead of size l. Again, we assume $k \geq 3$ and $l \leq k-2$. The first algorithm we propose is a slight modification of the MARKING algorithm due to Fiat *et al.* [5]. The MARKING algorithm proceeds in a series of phases. During each phase a set of marked pages is maintained. At the beginning of each phase all pages are unmarked. Whenever a page is requested, that page is marked. On a fault, a page is chosen uniformly at random from among the unmarked pages in fast memory, and that page is evicted. A phase ends immediately before a fault, when there are k marked pages in fast memory. At that moment all marks are erased and a new phase is started.

The modified algorithm with strong lookahead l uses lookahead once during each phase.

Algorithm MARKING(l): At the beginning of each phase execute an initial step: Determine the set S of pages which are in the present lookahead but not in fast memory. Choose $card(S)$ pages uniformly at random from among the pages in fast memory which are not contained in the current lookahead. Evict these pages and load the pages in S. After this initial step proceed with the MARKING algorithm.

Theorem 4 *Let $l \leq k-2$. The algorithm MARKING(l) with strong lookahead l is $2H(k-l)$-competitive.*

Proof: The idea of the proof is the same as the idea of the original proof of the MARKING algorithm [5]. We assume without loss of generality that MARKING(l)'s and OPT's fast memories initially contain the same k pages. During each phase we compare the cost incurred by MARKING(l) to the cost incurred by the optimal algorithm OPT. Consider an arbitrary phase. We use the same terminology as Fiat *et al.* A page is called *stale* if it is unmarked but was marked in the previous phase, and *clean* if it is neither stale nor marked.

Let c be the number of clean pages and s be the number of stale pages requested in the phase. Note that $c+s=k$. Fiat *et al.* prove that OPT has an amortized cost of at least $c/2$ during the phase.

We evaluate MARKING(l)'s cost during the phase. Serving c requests to clean pages obviously costs c. It remains to bound the expected cost for serving the stale pages. Let s_1 be the number of stale pages contained in the lookahead at the beginning of the phase and let $s_2 = s - s_1$. Then $s_1 + c \geq l+1$ because every page in the lookahead is either clean or counted in s_1. Thus $s_2 = s - s_1 \leq k - c - (l+1-c) = k-l-1$. Note that serving the first s_1 stale requests does not incur any cost and that we just have to evaluate MARKING(l)'s cost on the following s_2 requests to stale pages. We determine the expected cost of the (s_1+j)-th request to a stale page, $1 \leq j \leq s_2$. Let $\tilde{c}(j)$ be the number of clean pages which are requested in the phase before the (s_1+j)-th request to a stale page. Furthermore, let $\tilde{s}(j)$ be the number of stale pages which remain before that request. When MARKING(l) serves the (s_1+j)-th request to a stale page, exactly $\tilde{s}(j) - \tilde{c}(j)$ of the $\tilde{s}(j)$ stale pages are in fast memory (the $\tilde{s}(j) - \tilde{c}(j)$ pages in fast memory form a random subset of the $\tilde{s}(j)$ stale pages). Thus, the expected cost of the request equals

$$\frac{\tilde{s}(j) - \tilde{c}(j)}{\tilde{s}(j)} \cdot 0 + \frac{\tilde{c}(j)}{\tilde{s}(j)} \cdot 1 = \frac{\tilde{c}(j)}{\tilde{s}(j)}.$$

Note that $\tilde{c}(j) \le c$ and that $\tilde{s}(j) = k - s_1 - j + 1$ for $j = 1, 2, \ldots, s_2$. It follows that the expected cost of the requests to stale pages is bounded by

$$\begin{aligned} & \frac{c}{k-s_1} + \frac{c}{k-s_1-1} + \frac{c}{k-s_1-2} + \ldots + \frac{c}{k-s_1-s_2+1} \\ = \; & \frac{c}{k-s_1} + \frac{c}{k-s_1-1} + \frac{c}{k-s_1-2} + \ldots + \frac{c}{k-s+1}. \end{aligned}$$

The above sum consists of $s_2 \le k - l - 1$ terms and $\frac{c}{1}$ is missing. Hence the sum is bounded by $c(H(k-l) - 1)$, and we conclude that MARKING(l)'s cost during the phase is bounded from above by $cH(k-l)$. This proves the theorem because OPT's amortized cost during the phase is at least $c/2$. □

This following theorem implies that MARKING(l) is nearly optimal.

Theorem 5 *Let $l \le k - 2$ and let A be a randomized on-line paging algorithm with strong lookahead l. If A is c-competitive, then $c \ge H(k-l)$.*

Proof: The proof is similar to Raghavan's proof that no randomized on-line paging algorithm (without lookahead) can be better than $H(k)$-competitive [12]. Let $S = \{x_1, x_2, \ldots, x_{k+1}\}$ be a set of $k+1$ pages and let $SL = \{x_1, x_2, \ldots, x_l\}$. We assume without loss of generality that initially the pages $x_1, x_2, \ldots, x_k$ are in OPT's fast memory and in the fast memory of the on-line paging algorithm A.

The request sequence σ which we will choose consists of a series of phases $P(i)$. The first phase has the form $P(1) = x_1, x_2, \ldots, x_l, y_1$, where $y_1 = x_{k+1}$. Each of the following phases $P(i)$, $i \ge 2$, equals $P(i) = x_1, x_2, \ldots, x_l, y_i$, where y_i is chosen uniformly at random from $S \setminus \{SL \cup \{y_{i-1}\}\}$.

As in Raghavan's original proof, the request sequence σ can be partitioned into a series of rounds $R(1), R(2), R(3), \ldots$, such that during each round, OPT incurs a cost of exactly 1. The first round $R(1)$ consists of the first phase only, i.e., $R(1) = P(1)$. The following rounds $R(i)$, $i \ge 2$, contain requests to all $k+1$ pages in S. More specifically, each round $R(i)$, $i \ge 2$, is finished when, for the first time, every page in S has been requested at least once. Again, for $i = 1, 2, \ldots$, let t_i^e denote the end of round $R(i)$. Then round $R(i)$ comprises $\sigma(t_{i-1}^e + 1), \sigma(t_{i-1}^e + 2), \ldots, \sigma(t_i^e)$, where t_i^e satisfies

$$t_i^e = \min\{s > t_{i-1}^e | card(\{\sigma(t_{i-1}^e + 1), \sigma(t_{i-1}^e + 2), \ldots, \sigma(s)\}) = k+1\}.$$

Note that the end of each round coincides with the end of a phase.

It is easy to see that OPT can serve the request sequence in such a way that its cost in each round equals 1. On a fault, OPT simply evicts the page that will be requested last in the next round.

Now let DA be a deterministic on-line paging algorithm with strong lookahead l. We analyze DA's expected cost on σ. During the first round, DA incurs a cost of 1. We show that in each of the remaining rounds, DA has an expected cost of at least $H(k-l)$. Applying Yao's minimax principle [16] we obtain the theorem. In every phase $P(i)$, $i \geq 2$, DA has an expected cost of at least $\frac{1}{k-l}$. Using a technique presented in Raghavan's original proof, we can easily show that the expected number of phases in each round $R(i)$, $i \geq 2$, equals $(k-l)H(k-l)$. Thus DA's expected cost in each round $R(i)$, $i \geq 2$, is $H(k-l)$. □

We conclude this section by presenting another randomized algorithm, called RANDOM(l)-blocked. As the name suggests this algorithm is a variant of the algorithm RANDOM due to Raghavan and Snir [13]. On a fault RANDOM chooses a page uniformly at random from among the pages in fast memory and evicts that page. In terms of competitiveness RANDOM(l)-blocked represents no improvement upon the previously presented algorithms with strong lookahead. However, RANDOM(l)-blocked, as the original algorithm RANDOM, is very simple and uses no information on previous requests.

Algorithm RANDOM(l)-blocked: Serve the request sequence σ in a series of blocks. These blocks have the same structure as those in the algorithm LRU(l)-blocked. At the beginning of block $B(i)$ determine the set S_i of pages in $B(i)$ which are not in fast memory. Choose $card(S_i)$ pages uniformly at random from among the pages in fast memory which are not contained in $B(i)$. Evict these pages and load the pages in S_i. Then serve the requests in $B(i)$.

Theorem 6 *Let $l \leq k-2$. The algorithm RANDOM(l)-blocked with strong lookahead l is $(k-l+1)$-competitive.*

Proof: The potential function we use to analyze the algorithm is

$$\Phi(t) = (k-l) \cdot card(S_R(t) \setminus S_{OPT}(t)).$$

$S_R(t)$ denotes the set of pages contained in RANDOM(l)-blocked's fast memory after request t and $S_{OPT}(t)$ denotes the set of pages contained in OPT's fast memory after request t. We assume that RANDOM(l)-blocked and OPT start with the same initial fast memory.

Suppose the request sequence σ consists of b blocks $B(1), B(2), \ldots, B(b)$. We assume without loss of generality that the last block $B(b)$ contains $l+1$ distinct requests. The values t_i^b and t_i^e denote the beginning and the end of block $B(i)$, respectively. Define $t_0^e = 0$. Let $E[\Phi(t_i^e) - \Phi(t_{i-1}^e)]$ be the expected change in potential between t_{i-1}^e and t_i^e. Furthermore, let $C_R(i)$ and $C_{OPT}(i)$ denote the cost incurred by RANDOM(l)-blocked and OPT during block $B(i)$. We show that for all $i = 1, 2, \ldots, b$,

$$C_R(i) + E[\Phi(t_i^e) - \Phi(t_{i-1}^e)] \leq (k-l+1)C_{OPT}(i).$$

This inequality proves the theorem.

If $C_R(i) = 0$, then the inequality clearly holds. Each time OPT incurs a fault during block i, it might evict a page which is in $S_R(t^e_{i-1}) = S_R(t^e_i)$. Hence each eviction can increase the potential by $(k-l)$.

Now suppose $C_R(i) \geq 1$ and let

$$\tilde{C}(i) = card(S_R(t^e_{i-1}) \setminus \{L(t^b_i) \cup S_{OPT}(t^e_{i-1})\}).$$

We analyze the effect of the moves by RANDOM(l)-blocked and OPT on the potential function Φ, and assume that our on-line algorithm serves the current block first and OPT serves second.

RANDOM(l)-blocked evicts $C_R(i)$ pages at the beginning of block $B(i)$. The pages to be evicted are chosen from among $k-l-1+C_R(i)$ pages in fast memory which are not in $B(i)$. Among these $k-l-1+C_R(i)$ pages, exactly $\tilde{C}(i)$ pages contribute to $\Phi(t)$. Thus RANDOM(l)-blocked's evictions cause an expected decrease in potential of

$$(k-l)C_R(i)\frac{\tilde{C}(i)}{k-l-1+C_R(i)} \geq (k-l)\tilde{C}(i)\frac{1}{k-l} = \tilde{C}(i).$$

Note that a newly loaded page might not be in $S_{OPT}(t^e_{i-1})$, which can increase the potential by $(k-l)$ per page. Hence, RANDOM(l)-blocked's eviction rule cause an expected increase of potential of at most

$$-\tilde{C}(i) + (k-l)C_R(i).$$

We consider OPT's cost. Each time OPT evicts a page, this can increase the potential by $(k-l)$. Note that a page x which is requested in $B(i)$ is not in $S_{OPT}(t^e_i)$ if and only if it was evicted by OPT during $B(i)$. Hence

$$C_R(i) + E[\Phi(t^e_i) - \Phi(t^e_{i-1})] \leq C_R(i) - \tilde{C}(i) + (k-l)C_{OPT}(i).$$

Since $C_R(i) \leq C_{OPT}(i) + \tilde{C}(i)$, we obtain

$$C_R(i) + E[\Phi(t^e_i) - \Phi(t^e_{i-1})] \leq (k-l+1)C_{OPT}(i).$$

□

4 Open problems

In this paper we have introduced a new model of lookahead, called strong lookahead, and have analyzed several on-line paging algorithms using this model. One open problem is to determine the exact competitiveness of the algorithm LRU(l)-blocked. Is the algorithm $(k-l)$-competitive or can a lower of $(k-l+1)$ on the competitive factor be shown? Another open problem is to extend other k-competitive on-line paging algorithms, such as the algorithm FIFO, to our model of strong lookahead. Intuitively, FIFO(l), where $l \leq k-2$, would work as follows: On a fault it evicts the page that has been in fast memory longest among pages in fast memory that are not contained in the present lookahead. It is worth noting that the techniques which we have used in proving that LRU(l) is $(k-l)$-competitive can not be applied directly to show that FIFO(l) is $(k-l)$-competitive. Finally, an interesting problem is to present other models of lookahead that are of theoretical and practical interest and improve the competitive factors of on-line paging algorithms.

Acknowledgment

The author thanks Kurt Mehlhorn for helpful discussions. She also thanks Volker Priebe and Ronald Rasch for careful reading of the manuscript.

References

[1] A. Agarwal, R.L. Sites and M. Horowitz. ATUM: A new technique for capturing address traces using microcode. In *Proc. 13th Annual Symposium on Computer Architecture*, pages 119–127, 1986.

[2] L.A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, **5**:78–101, 1966.

[3] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, **11**(1):2–14, 1994.

[4] F.K. Chung, R. Graham and M.E. Saks. A dynamic location problem for graphs. *Combinatorica*, **9**(2):111–131, 1989.

[5] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, **12**:685–699, 1991.

[6] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, **11**(1):73–91, 1994.

[7] M.M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. In *Proc. 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 211–216, 1992.

[8] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, **11**(1):53–62, 1994.

[9] M.-Y. Kao and S.R. Tate. Online matching with blocked input. *Information Processing Letters*, **38**:113–116, 1991.

[10] A.R. Karlin, M.S. Manasse, L. Rudolph and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, **3**(1):79–119, 1988.

[11] L.A. McGeoch and D.D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, **6**:816–825, 1991.

[12] P. Raghavan. Lecture notes on randomized algorithms. IBM Research Report No. RC 15340 (# 68237), Yorktown Heights, 1989.

[13] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, Springer Lecture Notes in Computer Science, Vol. 372, pages 687–703, 1989.

[14] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, **28**:202–208, 1985.

[15] J.R. Spirn. *Program Behavior: Models and Measurements*. Elsevier, New York, 1977.

[16] A.C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[17] N. Young. *Competitive Paging and Dual-Guided On-Line Weighted Caching and Matching Algorithms*. Ph.D. thesis, Princeton University, 1991. Available as Computer Science Department Technical Report CS-TR-348-91.