

# Cross-input Amortization Captures the Diffuse Adversary

Neal E. Young



## Technical Report PCS-TR96-302

Department of Computer Science  
Dartmouth College  
Hanover, NH 03755-3510  
ney@cs.dartmouth.edu

### Abstract

Koutsoupias and Papadimitriou recently raised the question of how well deterministic on-line paging algorithms can do against a certain class of *adversarially biased* random inputs [3]. Such an input is given in an on-line fashion; the adversary determines the next request probabilistically, subject to the constraint that no page may be requested with probability more than a fixed  $\epsilon > 0$ .

In this paper, we answer their question by estimating, within a factor of two, the optimal competitive ratio of any deterministic on-line strategy against this adversary. We further analyze randomized on-line strategies, obtaining upper and lower bounds within a factor of two. These estimates reveal the qualitative changes as  $\epsilon$  ranges continuously from 1 (the standard model) towards 0 (a severely handicapped adversary).

The key to our upper bounds is a novel charging scheme that is appropriate for adversarially biased random inputs. The scheme adjusts the costs of each input so that the expected cost of a *random* input is unchanged, but working with adjusted costs, we can obtain worst-case bounds on a per-input basis. This lets us use worst-case analysis techniques while still thinking of some of the costs as *expected* costs.

# 1 Introduction

In the theoretical analysis of algorithms, measuring an algorithm by its worst-case performance is often impractically pessimistic. On the other hand, measuring an algorithm by its average-case on a specific input distribution is impractically presumptuous. A plausible approach falling in between these two standard models is to assume that *something*, but not *everything*, is known about the distribution from which inputs are generated. One model for this is to assume that the inputs are generated by a random source that has been *biased* in a constrained fashion by an adversary that chooses the worst possible bias for the algorithm in question.

Koutsoupias and Papadimitriou [3] recently studied how well any deterministic on-line paging algorithm can do against such an adversary. Their adversary,  $\Delta_\epsilon$ , is allowed to select the next request only probabilistically, with each page being requested with probability at most some  $\epsilon > 0$ . Koutsoupias and Papadimitriou prove that the least-recently-used strategy (LRU) is an optimal deterministic on-line algorithm against this adversary, but they leave open the problem of giving a closed-form estimate of the optimal competitive ratio  $R(\Delta_\epsilon)$ , commenting “It seems difficult to determine ... the exact competitive ratio. ... In fact ... the ratio may not be expressible as a simple closed form expression.” In this paper, we estimate this ratio within a factor of roughly two for all  $k$  and  $\epsilon$ . Specifically, we show

**Theorem 1.1** *Fix any integer  $k > 0$  and any  $\epsilon \in (0, 1]$ . Then for  $\epsilon \leq \frac{1}{k+1}$ ,*

$$1 + \ln\left(\frac{1}{1 - \epsilon(k-1)} - 1\right) \leq R(\Delta_\epsilon) \leq R(\Delta_\epsilon, \text{LRU}) \leq 2\left(1 + \ln\frac{1}{1 - \epsilon k}\right),$$

*while for  $\epsilon \geq \frac{1}{k+1}$ ,*

$$1 + k - \frac{1}{\epsilon} + \ln\frac{1}{2\epsilon} \leq R(\Delta_\epsilon) \leq R(\Delta_\epsilon, \text{LRU}) \leq 2\left(1 + k - \frac{1}{\epsilon} + \ln\frac{1}{\epsilon}\right).$$

*The upper bound holds for first-in-first-out (FIFO) and the marking algorithm (MARK, [2, 4, 5]) as well as for LRU.*

(The upper bound does not hold for flush-when-full (FWF).) We also estimate the optimal ratio for randomized on-line strategies,  $R_r(\Delta_\epsilon)$ , which Koutsoupias and Papadimitriou do not consider.

**Theorem 1.2** *Fix any integer  $k > 0$  and any  $\epsilon \in (0, 1]$ . For  $\epsilon \leq \frac{1}{k+1}$ ,*

$$1 + \ln\left(\frac{1}{1 - \epsilon(k-1)} - 1\right) \leq R_r(\Delta_\epsilon) \leq R(\Delta_\epsilon, \text{MARK}) \leq 2\left(1 + \ln\frac{1}{1 - \epsilon k}\right),$$

*while for  $\epsilon \geq \frac{1}{k+1}$ ,*

$$1 + \ln\frac{k}{2} \leq R_r(\Delta_\epsilon) \leq R(\Delta_\epsilon, \text{MARK}) \leq 2(1 + \ln k).$$

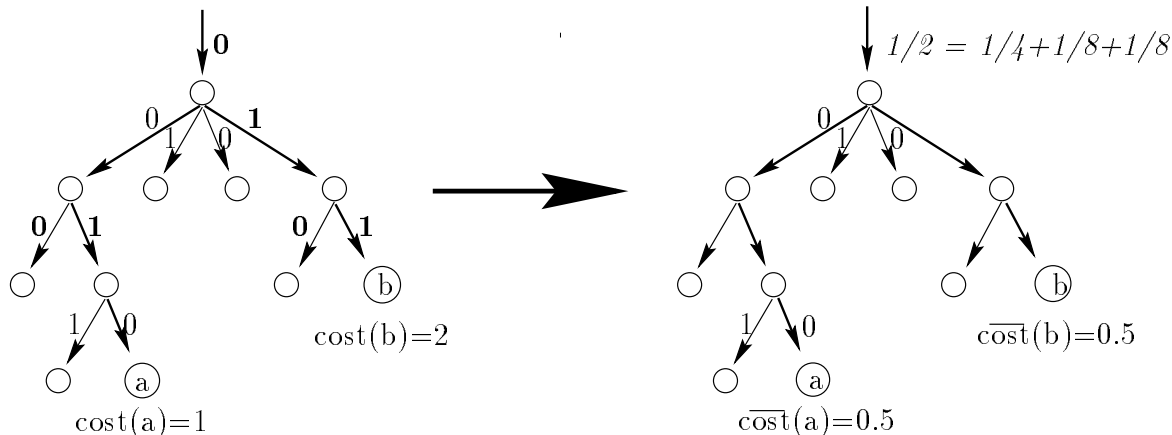


Figure 1: To bound the expected cost against an adversarially biased random input, the costs are adjusted so that (1) the expected cost of a *random* input stays the same, but (2) the adjusted cost of each input can now be bounded *individually*. A random input sequence corresponds to a random walk down the tree. The increased cost at the top edge represents “insurance” against edges below whose costs (in bold on the left) no longer have to be paid.

As  $\epsilon$  varies from 1 to 0, the diffuse adversary becomes more and more restricted. With  $\epsilon = 1$ , the diffuse adversary is the standard adversary. When  $\epsilon$  is above the threshold  $1/k$ , the ratio for deterministic strategies ranges from  $k$  down to about  $\ln k$ . In this range randomized strategies can do no better than against the standard adversary. For  $\epsilon$  below the threshold  $1/k$ , the ratio for deterministic strategies ranges from  $\ln k$  down to constant (as  $\epsilon k$  grows away from 1). In this range, randomized strategies do essentially no better than deterministic strategies.

The key to our upper bounds is a novel charging scheme for analyzing adversarially biased random inputs (see Figure 1). The charging scheme adjusts the costs of each input in such a way that the expected cost of a random input is unchanged, but so that, working with the adjusted costs, we can show *worst-case* bounds on each input. This allows us to reduce the analysis to a worst-case analysis, while still thinking of some of the costs incurred on a given input as average-case.

## 2 Definitions

The paging problem, given an integer  $k > 0$ , is to dynamically maintain a cache (set) of at most  $k$  pages in response to a sequence of requests for pages so as to minimize the number of *page faults*. A page fault occurs when the requested page is not in the cache, at which point the page must be brought into the cache. If there are  $k$  pages in the cache already, one must be evicted (removed) before bringing in the requested item. An algorithm for the problem must specify which page to evict when a fault occurs. Given an algorithm  $A$  and a sequence  $x$ , we let  $A(x)$  denote the cost (number of faults) incurred by  $A$  in servicing  $x$ . The optimal algorithm, OPT [1], evicts the page that will be next requested latest. An algorithm

is *on-line* if the choice of which page to evict is independent of subsequent requests.

Following Koutsoupias and Papadimitriou [3], given a known class of distributions  $\Delta$  of the input sequences and an algorithm  $A$ , define

$$R(\Delta, A) = \max_{D \in \Delta} \frac{\mathbb{E}_D[A(x)]}{\mathbb{E}_D[\text{OPT}(x)]}$$

and

$$R(\Delta) = \min_A R(\Delta, A),$$

where  $A$  ranges over the class of deterministic on-line algorithms, and

$$R_r(\Delta) = \min_A R(\Delta, A),$$

where  $A$  ranges over the class of randomized on-line algorithms. (Note that  $A(x)$  denotes the number of faults made by  $A$  on input  $x$ ;  $k$  is an implicit parameter to these definitions.) Koutsoupias and Papadimitriou call this the *diffuse adversary* model.

Any distribution  $D$  specifies, for each page  $p$  and sequence of page requests  $x$ , the probability  $\Pr_D(p|x)$  that the next request is  $p$  given that the sequence so far is  $x$ . Define  $\Delta_\epsilon$  to be the distributions  $D$  such that, for any request sequence  $x$  and page  $p$ ,  $\Pr_D(p|x) \leq \epsilon$ . We are interested in determining  $R(\Delta_\epsilon)$  and  $R_r(\Delta_\epsilon)$ .

### 3 Upper Bound on Deterministic Strategies

Our strategy is to “adjust” the costs incurred by LRU on any distribution  $D \in \Delta_\epsilon$  so that the expected cost of a random input from  $D$  is not decreased, but so that, working with the adjusted costs, we can show that on *any* sequence  $x$ , the adjusted cost  $\overline{\text{LRU}}(x)$  is bounded by the appropriate factor  $c$  times  $\text{OPT}(x)$ . This will give the upper bound via

$$\frac{\mathbb{E}[\text{LRU}(x)]}{\mathbb{E}[\text{OPT}(x)]} \leq \frac{\mathbb{E}[\overline{\text{LRU}}(x)]}{\mathbb{E}[\text{OPT}(x)]} \leq \max_x \frac{\overline{\text{LRU}}(x)}{\text{OPT}(x)} \leq c.$$

#### 3.1 The Amortized Cost $\overline{\text{LRU}}(x)$ .

We now motivate and define the adjusted cost  $\overline{\text{LRU}}(x)$  of any sequence  $x$ . Partition  $x$  into consecutive subsequences, called *k-phases*, or just *phases*, such that each phase (except possibly the last) contains exactly  $k$  distinct pages and each phase (other than the first) begins with a page not occurring in the previous phase. Classify each request according to the structure of the phase in which it occurs as follows:

- redundant — the page requested was previously requested during the phase (these requests play an important role in the analysis);
- new — the page requested was not requested during this or the previous phase;
- worrisome — the request is not new, but nonetheless causes LRU to fault.

It was previously observed [2, 4, 5] that in a phase with  $m$  new requests, OPT incurs at least  $m/2$  faults in the traditional amortized sense. (This is because in two consecutive phases, if the second has  $m$  new requests, then  $k + m$  distinct pages are requested; since OPT has a cache of size  $k$ , it must incur at least  $m$  faults during the two phases.) On the other hand, OPT incurs at most  $m$  faults in the phase. (This is because OPT has the option of starting the phase with the  $k$  pages from last phase and then evicting  $m$  of these pages that won't be requested this phase.)

Note that LRU doesn't evict a page during a phase once it has been requested and that LRU starts a phase with all of the pages from the previous phase in its cache. (In fact, these are the only two properties of LRU that we use. Since these properties are shared by first-in-first out and MARK, these upper bounds apply to them as well.) Thus, each new request definitely causes a fault, and each *worrisome* request requests a page that was previously requested during the current phase but subsequently evicted by LRU. The new requests are not a problem since OPT is also paying for those. The worrisome requests are the problem — we need to argue they don't cost too much.

To do this we need to use the limitations on the adversary. These limitations do not hold on a per-sequence basis. In fact, if the random sequence is conditioned on any future event the limitation on the adversary does not hold. Thus, for instance, we cannot even partition the sequences into groups such that each sequence in a group shares a similar phase structure; if the random input is conditioned on membership in such a group, we lose the guarantee that at any given point, the adversary can request any given page with probability at most  $\epsilon$ .

Instead, we adjust the costs of the worrisome requests. As the sequence is given and processed by LRU in an on-line fashion, LRU is not charged (directly) for worrisome requests. Instead, after each non-redundant request, LRU pays an “insurance premium” equal to the *probability* that the next non-redundant request would be a worrisome request *if the remaining requests were generated randomly by the adversary*. Having paid this insurance premium, LRU only pays for the next non-redundant request if it turns out to be a new request, not if it is a worrisome request. On any *given* sequence, the cost may be reduced (if there are many worrisome requests) or increased (if there are few worrisome requests). On the other hand, over all random inputs, the average cost paid by LRU is unchanged, because on average the insurance premiums exactly pay for the worrisome requests (see Figure 1). Thus we have

**Lemma 3.1** *For any  $D \in \Delta_\epsilon$ ,  $E_D[\text{LRU}(x)] \leq E_D[\overline{\text{LRU}}(x)]$ .*

### 3.2 The Worst-Case Bound on $\overline{\text{LRU}}(x)/\text{OPT}(x)$ .

Next we give a worst-case bound on  $\overline{\text{LRU}}(x)/\text{OPT}(x)$  for any  $x$ . We do this on a per-phase basis. The main task is bounding the cost of the insurance premiums paid during the phase (the only other cost charged to LRU is for new requests). We bound the cost of the premiums in terms of the number of new requests in the phase. As described previously, we know the number of new requests is at most twice the cost incurred by OPT in the phase.

Insurance premiums are paid only following non-redundant requests, so let  $p$  be any non-redundant request in the phase. Let  $m$  be the number of new requests so far (i.e., in the phase, up to and including  $p$ ). Let  $i$  be the number of non-redundant requests so far.

There are at most  $m$  pages that, if requested next, would result in a worrisome request (of the  $k$  pages requested in the previous phase, all but  $m$  are in the cache). There are at most  $i$  pages that, if requested next, would result in a redundant request (the  $i$  distinct pages requested so far). If any other page were requested next, it would either be a new request or a non-worrisome, non-redundant request (to a page in LRU's cache). Thus, the only way that the next non-redundant request could be a worrisome request is if the upcoming sequence of requests were to consist of some sequence of the  $i$  possible redundant requests followed by a request to one of the  $m$  possible worrisome requests.

Since the adversary can assign a probability of at most  $\epsilon$  to any page, the probability that the next non-redundant request would be worrisome is bounded by

$$\sum_{\ell} (\epsilon i)^{\ell} \epsilon m = \frac{\epsilon m}{1 - \epsilon i}$$

(or 1 if the quantity on the right-hand side is negative or more than 1). This is our upper bound on the insurance payment paid by LRU after request  $p$ . Over the course of an entire phase with  $m$  new requests, the number of new requests plus the sum of these upper bounds is at most

$$m + \sum_{i=1}^{k-1} \overline{\min\left\{\frac{\epsilon m}{1 - \epsilon i}, 1\right\}} \quad (1)$$

where by  $\overline{\min\{a, 1\}}$  we mean 1 if  $a$  is negative or more than 1 and  $a$  otherwise. (Note that after the  $k$ th non-redundant request, the insurance payment is 0 because the next non-redundant request is necessarily a new request.)

Using calculus to estimate the sum in (1), using the fact that  $\overline{\text{LRU}}(x)$  is the sum of the number of new requests and the insurance payments, and using the fact that the number of new requests in  $x$  is at most  $2\text{OPT}(x)$ , we obtain

**Lemma 3.2** *For any request sequence  $x$ ,*

$$\frac{\overline{\text{LRU}}(x)}{\text{OPT}(x)} \leq \begin{cases} 2\left(1 + \ln \frac{1}{1 - \epsilon k}\right) & \text{when } \epsilon \leq \frac{1}{k+1}, \\ 2\left(1 + k - \frac{1}{\epsilon} + \ln \frac{1}{\epsilon}\right) & \text{when } \epsilon \geq \frac{1}{k+1}. \end{cases}$$

Combining this with Lemma 3.1 gives the upper bounds stated in Theorem 1.1.

## 4 Lower Bound on Deterministic Strategies

In this section we show that for any deterministic on-line strategy  $A$ , there is a distribution  $D$  in  $\Delta_{\epsilon}$  such that the expected cost of the strategy divided by the expected cost of OPT is at least roughly half the upper bound proved for LRU.

We describe  $D$  by describing an adversary that requests pages probabilistically subject to the limitations of  $\Delta_{\epsilon}$ . Fix  $\epsilon > 0$  and  $k > 0$ . For simplicity we assume  $\epsilon > 1/2k$  (otherwise the upper bound already established shows the ratio is at most a small constant).

Fix  $m = \max\{1, \lceil \frac{1}{\epsilon} \rceil - k\}$ . The adversary requests the pages in an on-line fashion. At the beginning of a phase, the adversary requests  $m$  new pages by assigning probability only

to pages not in LRU's cache. (Here we follow [3] in assuming that there are at least  $\frac{1}{\epsilon} + k$  pages available.)

For the remainder of the phase, at each request the adversary assigns probability to pages with the following priority. First priority is given to pages requested last phase or this phase but not in  $A$ 's cache. Second priority is given to pages requested last phase or this phase that are in  $A$ 's cache. Third priority is given to any other pages in  $A$ 's cache. Probability is assigned according to priority, subject to the constraint that no page gets more than  $\epsilon$  priority. By the choice of  $m$ , these three classes of pages suffice for all the probability to be assigned.

A straightforward argument shows that after  $i \geq m$  distinct pages have been requested, the probability that the next request to a page not yet requested this phase causes a fault is at least

$$\frac{\epsilon m}{1 - \epsilon i}$$

or 1 if this quantity is negative or more than 1. Thus the expected number of faults incurred by  $A$  during the phase is at least

$$m + \sum_{i=m}^{k-1} \overline{\min}\left\{\frac{\epsilon m}{1 - \epsilon i}, 1\right\}$$

where as before  $\overline{\min}\{a, 1\}$  denotes 1 if  $a$  is negative or more than 1 and  $a$  otherwise. As in the upper bound we use calculus to bound the sum, and we use the fact that OPT incurs at most  $m$  faults in the phase to show

**Lemma 4.1** *For any  $\epsilon > 0$  and any deterministic on-line algorithm  $A$ ,*

$$R(\Delta_\epsilon, A) \geq \begin{cases} 1 + \ln\left(\frac{1}{1 - \epsilon(k-1)} - 1\right) & \text{when } \epsilon \leq \frac{1}{k+1} \\ 1 + k - \frac{1}{\epsilon} + \ln\frac{1}{2\epsilon} & \text{when } \epsilon \geq \frac{1}{k+1}. \end{cases}$$

This gives the lower bounds in Theorem 1.1.

## 5 Randomized Strategies

In this section, for any randomized on-line strategy  $A$ , we describe a distribution  $D$  in  $\Delta_\epsilon$  that gives a good lower bound on  $R(\Delta_\epsilon, A)$ . The method is similar to the deterministic lower bound, and in fact the bounds coincide for small  $\epsilon$ .

Fix  $\epsilon > 0$  and  $k > 0$ . Set  $m = \max\{1, 1/\epsilon - k\}$ . To begin a phase, first make  $m$  new requests. For subsequent requests, simply assign probability  $1/(k+m)$  to each of the  $k$  pages requested in the previous phase and each of the  $m$  pages requested at the beginning of this phase. Standard techniques show that the expected number of faults before the phase ends is at least  $m(1 + \ln\frac{k+1}{m+1})$ . For this phase OPT incurs a cost of at most  $m$ , so the ratio is

**Lemma 5.1** *For any  $\epsilon > 0$  and any randomized on-line algorithm  $A$ ,*

$$R(\Delta_\epsilon, A) \geq \begin{cases} 1 + \ln\frac{k}{2} & \text{when } \epsilon \geq \frac{1}{k+1} \\ 1 + \ln\left(\frac{1}{1 - \epsilon(k-1)} - 1\right) & \text{when } \epsilon \leq \frac{1}{k+1}. \end{cases}$$

This gives the lower bounds in Theorem 1.2. For the upper bounds in the theorem, note that the marking algorithm is known to be  $2(1 + \ln(k))$ -competitive against the standard adversary, while we've shown in a previous section that the ratio of the marking algorithm or LRU when  $\epsilon \leq 1/(k + 1)$  is at worst  $2(1 + \ln \frac{1}{1-\epsilon k})$ . Thus these lower bounds are tight within a factor of roughly two.

## References

- [1] L. A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78–101, 1966.
- [2] 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.
- [3] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. In *Proc. of the 35th IEEE Annual Symp. on Foundation of Computer Science*, pages 394–400, 1994.
- [4] Neal Young. Competitive paging and dual-guided algorithms for weighted caching and matching. (Thesis) Tech. Rep. CS-TR-348-91, Computer Science Department, Princeton University, October 1991.
- [5] Neal Young. The  $k$ -server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.