CS294-1 On-line Computation & Network Algorithms	**Spring 1997**

Lecture 4: January 30

Lecturer: Yair Bartal *Scribe: Tzu-Yi Chen*

In this lecture we begin by reviewing some basic definitions and then go on to examine the random-marking algorithm for the paging problem in more detail, often generalizing to draw conclusions regarding randomized algorithms in general.

4.1 Review

4.1.1 Definition of Competitiveness

We first repeat the definition for deterministic algorithms and then define it for randomized algorithms.

Definition 4.1 (c-competitiveness) *For deterministic algorithms, an algorithm A is c-competitive iff*

$$\exists a \geq 0 \; s.t. \; \forall \sigma, \;\; A(\sigma) \leq c \cdot \; \mathrm{OPT}(\sigma) + a$$

This is equivalent to saying

$$\exists a \geq 0 \; s.t. \; \forall \sigma \forall \mathrm{ADV}, \;\; A(\sigma) \; \leq c \cdot \mathrm{ADV}(\sigma) + a$$

The intuition is that the adversary ADV*plays against A and tries to maximize c.*

The definition can be extended to randomized algorithms by adding expectations as follows. A randomized algorithm A is c-competitive iff

$$\exists a \geq 0 \text{ s.t. } \forall \sigma, \;\; E(A(\sigma)) \leq c \cdot \mathrm{OPT}(\sigma) + a$$

This is equivalent to saying

$$\exists a \geq 0 \text{ s.t. } \forall \sigma \forall \mathrm{ADV}, \;\; E(A(\sigma)) \; \leq c \cdot \mathrm{ADV}(\sigma) + a$$

4.1.2 Definition of an Adversary

ADV stands for an adversary. This is an algorithm which tries to maximize c, whereas we (the on-line player) try to minimize c. The adversary does this by choosing sequences of inputs. For example, with the paging problem, the adversary could consistently choose to request pages not in the cache, forcing our algorithm to page fault on every request.

When dicussing randomized algorithms it is important to define carefully the exact power given to the adversary. There are several kinds of adversaries (formally defined in [BBKTW90]):

- **oblivious adversary:** This adversary must pick the sequence completely in advance, without knowing anything about the coin tosses of the other algorithm. The definition given above applies to this kind of adversary. Randomized algorithms often perform especially well against such adversaries.
- **adaptive adversary:** These are adversaries which can see the coin tosses of the other algorithm and adapt to them. Randomized algorithms do not perform as well here. For adaptive adversaries, the definition of competitiveness must be adjusted as follows:
  - **on-line:** An on-line adaptive adversary must serve the sequence upon generation of requests.

$$E(A(\sigma)) \leq c \cdot E(\mathrm{ADV}(\sigma)) + a. \tag{4.1}$$

  - **off-line:** An off-line adaptive adversary may serve the sequence optimally at the end.

$$E(A(\sigma)) \leq c \cdot E(\mathrm{OPT}(\sigma)) + a.$$

Note that definition 4.1 is the most general and we may use it in all three cases once defining the set of advesaries considered.

Theorem 4.2 *For the paging problem, the competitive ratio of any random on-line algorithm against an adaptive on-line algorithm is $\geq k$.*

This will be proved in the next lecture in the more general context of the k-server problem.

For an algorithm A. Let $\rho_{\mathrm{OBL}}(A), \rho_{\mathrm{ADOL}}(A), \rho_{\mathrm{ADOFF}}(A)$ be its competitive ratios against the three kinds of adversaries then the following relations hold:

Claim 4.3 $\rho_{\mathrm{OBL}}(A) \leq \rho_{\mathrm{ADOL}}(A) \leq \rho_{\mathrm{ADOFF}}(A)$

4.2 Random Marking

In contrast the competitive ratio of randomized algorithms for the paging problem against an oblivious advesary is H_k defined below. (Note that $\ln k < H_k < \ln k + 1$.)

Definition 4.4 (H_i) *H_i is the i-th harmonic number, so $H_i = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$.*

We will prove that the Random Marking algorithm of [FKLMSY91] is $2H_k$-competitive. This analysis has been tightened by [ACN96] to show the competitive ratio of Random Marking is actualy $2H_k - 1$. Optimal H_k-competitive algorithms were first given by [MS91] and then by [ACN96]. Both algorithms are quite complicated and will not be dicussed here.

The Random Marking paging algorithm works in phases as follows:

- initially (at the beginning of a phase), all the pages in the cache are unmarked.
- upon a request for a page p:

  case 1 p is in the cache $\rightarrow$ mark p.

  case 2 p is not in the cache $\rightarrow$ if all the pages in the cache are marked, then unmark all the pages and start a new phase. Evict a uniformly chosen random unmarked page. Fetch p and mark p.

Here is a simple illustration of how the random marking algorithm works on a cache of size 4.

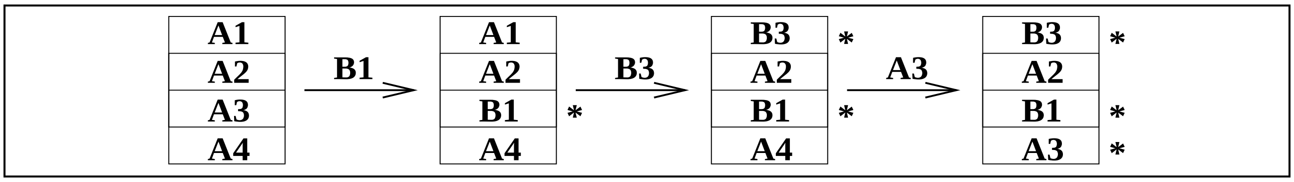

Figure 4.1: Example of how random marking works

We will show that random marking is $2H_k$ competitive.

Next we define a few new terms by first considering the decomposition into phases $P_1, P_2, P_3, \ldots$, where the ith phase P_i is defined as the longest subsequence after P_{i-1} that contains requests for at most k-distinct pages.

We'll label a page:

- OLD for phase P_i if it was in the cache at the beginning of the phase.
- NEW for phase P_i if it was requested during phase P_i but wasn't OLD.

Now let n_i be the number of new pages in P_i.

Claim 4.5 $\mathrm{OPT}(\sigma) \geq \frac{1}{2}\sum_{i=1}^{m} n_i$

Proof: Let q_i be the number of page faults to OPT during P_i. Then:

$$\begin{aligned} q_{i-1} + q_i &\geq n_i \\ \sum_{i=2}^{m}(q_{i-1} + q_i) &\geq \sum_{i=2}^{m} n_i \\ 2\sum_{i=1}^{m} q_i - q_1 &\geq \sum_{i=2}^{m} n_i \\ 2\,\mathrm{OPT}(\sigma) &\geq q_1 + \sum_{i=2}^{m} n_i \\ 2\,\mathrm{OPT}(\sigma) &\geq \sum_{i=1}^{m} n_i \end{aligned}$$

■

Claim 4.6 $\mathrm{RMARK}(\sigma) \leq H_k \sum_{i=1}^{m} n_i$

Proof: There have been $k - n_i$ OLD pages requested in P_i. Let's label these pages $B_1, B_2, \ldots, B_{k-n_i}$ and consider the requests for them.

In particular, given a request for B_j, what is the probability that this page is in the cache? We want to know this because

$$E(\mathrm{RMARK}(\sigma)) = n_i + \sum_{j=1}^{k-n_i} (\text{probability that } B_j \text{ is not in the cache when it is first requested})$$

Let $\tilde{n}_j$ be the number of NEW pages requested prior to the first request for B_j. Assume there are $(j-1)$ OLD pages already marked. The number of OLD pages previously evicted and not yet requested is equal to $\tilde{n}_j$. Therefore there are $k - \tilde{n}_j - (j-1)$ pages which are OLD and unmarked.

$$\begin{aligned}
E(\mathrm{RMARK}(\sigma)) &= n_i + \sum_{j=1}^{k-n_i} \frac{\tilde{n}_j}{k-(j-1)} \\
\text{since } \tilde{n}_j &\leq n_i \text{ we get} \\
E(\mathrm{RMARK}(\sigma)) &= n_i(1 + \sum_{j=n_i+1}^{k} \frac{1}{j}) \\
&= n_i(1 + H_k - H_{n_i}) \\
&\leq H_k n_i
\end{aligned}$$

■

Theorem 4.7 *Random Marking is $2H_k$-competitive against an oblivious adversary. (side note: It can be shown to be $2H_k - 1$ competitive, but we won't do that here.)*

Proof:

$$\begin{aligned}
E(\mathrm{RMARK}(\sigma)) &\leq H_k \sum_{i=1}^{m} n_i \\
\frac{1}{2}\sum_{i=1}^{m} n_i &\leq \mathrm{OPT}(\sigma) \\
E(\mathrm{RMARK}(\sigma)) &\leq 2H_k \mathrm{OPT}(\sigma)
\end{aligned}$$

Therefore RMARK is $2H_k$ competitive. ■

4.3 Lower Bounds for Randomized Algorithms

Because it is easier to bound deterministic algorithms than randomized algorithms, we begin by stating Yao's principle [Yao77], which relates deterministic and randomized algorithms. Yao's principle is an application of the Von-Neumann—Morgenstein Minimax Theorem from basic game theory. This section considers an application of Yao's principle for finding lower bounds for randomized paging algorithms.

Principle 4.8 (Yao's Principle) *If we define the following:*

$$\begin{array}{rcl} \mathcal{R} & = & \textit{set of random algorithms} \\ \mathcal{S} & = & \textit{set of all possible sequences} \\ \mathcal{D} & = & \textit{set of deterministic algorithms} \\ \mathcal{P} & = & \textit{set of all probability distributions over sequences} \end{array}$$

and let $f : \mathcal{D} \times \mathcal{S} \to \Re$.

then:

$$\inf_{A \in \mathcal{R}} \sup_{\sigma \in \mathcal{S}} E_A(f(A, \sigma)) = \sup_{d \in \mathcal{P}} \inf_{A \in \mathcal{D}} E_d(f(A, \sigma_d))$$

Proof: Let

$$\sup_{d \in \mathcal{P}} \inf_{A \in \mathcal{D}} E_d(f(A, \sigma_d)) = C.$$

We'll show

$$\inf_{A \in \mathcal{R}} \sup_{\sigma \in \mathcal{S}} E_A(f(A, \sigma)) \geq C$$

(This is the useful direction. The other direction can be proven similarly.)

Assume by contradiction that

$$\begin{array}{rll} \exists A \in \mathcal{R} & \text{s.t.} & \sup_{\sigma \in \mathcal{S}} E_A(f(A, \sigma)) < C \\ \forall d \in \mathcal{P} & \text{s.t.} & E_d(E_A(f(A, \sigma_d))) < C \\ & & E_A(E_d(f(A, \sigma_d))) < C \\ \Rightarrow \exists A \in \mathcal{D} & \text{s.t.} & E_d(f(A, \sigma_d)) < C \end{array}$$

That is a contradiction, which concludes the sketch of our proof. ■

4.3.1 The Paging Problem

We now turn to proving a lower bound for the random marking paging algorithm.

Theorem 4.9 *The competitive ratio of any randomized algorithm for the paging problem against an oblivious advsversary is at least H_k.*

The H_k lower bound was orginially proved by [FKLMSY91]. However we will present a somewhat more elegant proof, similar to the one of [BLS87] for uniform Metrical Task Systems, which is based on Yao's principle.

We let $n = k + 1$, and define the probability distribution over the request sequences as follows: every request is chosen uniformly among all pages with probability $\frac{1}{k+1}$.

Consider a sequence σ of length l. For any on-line algorithm, $E(A(\sigma)) = \frac{l}{k+1}$. We now examine the value of $E(\text{OPT}(\sigma))$.

As before we define a phase to be the longest following subsequence such that most k distinct pages are requested.

If $X_1 X_2 X_3 \ldots X_m$ defines the sequence of requests, where X_i is the subsequence of requests in phase i, then we know that $\text{OPT}(\sigma) \leq m$, since it would fault exactly once per phase. We now want to find the expected value for m.

Claim 4.10

$$\begin{aligned} E(|X_i|) &\leq H_k \cdot (k+1) \\ \Rightarrow E(m) &\leq \frac{l}{H_k \cdot (k+1)} \end{aligned}$$

Proof: The first equality is equivalent to the *coupon collector problem*, in which you have n boxes into which you repeatedly toss balls. Each ball has an equal probability for falling into each box. The question is how many balls need to be thrown before all the boxes have at least one ball. The coupon collector problem is well known and a simple analysis shows that the number of balls b needed to be thrown until all the boxes are filled is:

$$\begin{aligned} b &= 1 + \frac{1}{(\frac{n-1}{n})} + \frac{1}{(\frac{n-2}{n})} + \ldots + \frac{1}{(\frac{1}{n})} \\ &= n[1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}] \\ &= nH_n \end{aligned}$$

where H_n is the n-th harmonic number.

The analogy to paging is as follows — each empty box corresponds to an unmarked page, and each full box corresponds to a marked page. The balls correspond to requests for pages. When all the boxes are full, all the pages are marked, and a new phase begins. Since we are interested in the expected number of requests before a new phase begins, we get

$$E(|X_i|) = nH_n - 1 = nH_{n-1} = (k+1)H_k.$$

The second inequality follows from basic Renwal Theory. ■

And, finally, a little graph that shows approximately how competitive all these paging algorithms really are. This is based on experimental results by Young [Young94].

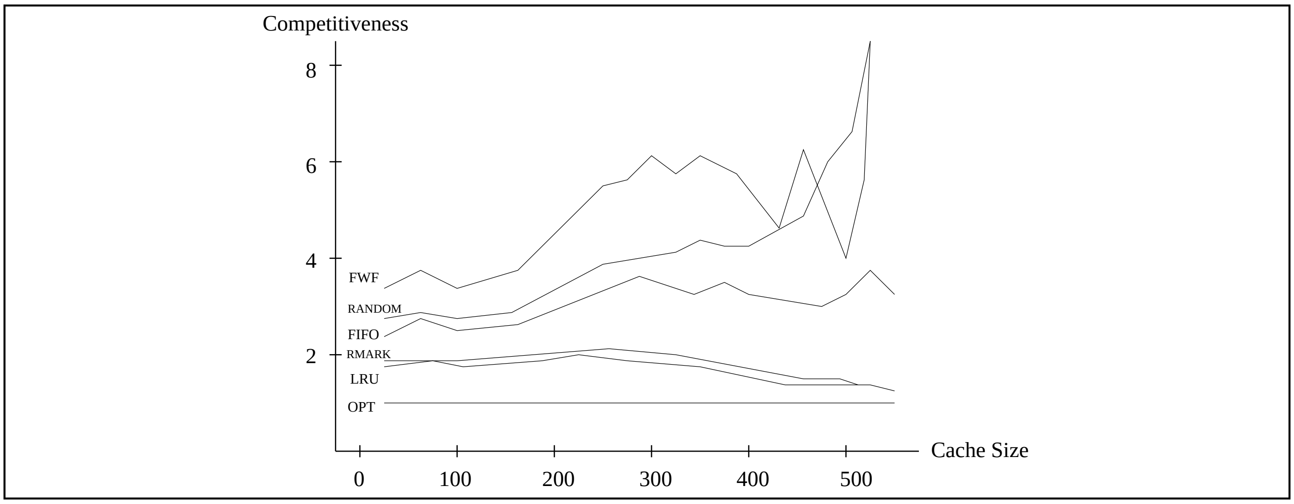

Figure 4.2: Competitive ratios of various paging algorithms as a function of cache size.

References

[ACN96] D. Achlioptas, M. Chrobak and J. Noga. Competitive Analysis of Randomized Paging Algorithms. In *Proc. 4th European Symposium on Algorithms*, Springer, LNCS 1136, pages 419-430, 1996.

[BBKTW90] S. Ben-David, A. Borodin, R.M. Karp, G. Tárdos, and A. Wigderson. On the Power of Randomization in On-line Algorithms. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 379-386, May 1990.

[BLS87] A. Borodin, N. Linial, and M. Saks. An Optimal On-Line Algorithm for Metrical Task Systems. In *Proc. of the 19th Ann. ACM Symp on Theory of Computing*, pages 373-382, May 1987.

[FKLMSY91] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator , and N.E. Young. Competitive Paging Algorithms. Journal of Algorithms, 12(4), pages 685-699, 1991.

[MS91] L.A. McGeoch and D.D. Sleator. A Strongly Competitive Randomized Paging Algorithm, 1991. Submitted to *Algorithmica.*

[Yao77] A.C. Yao. Probabilistic Computations: Towards a Unified Measure of Complexity. In *Proc. of the 17th Annual Symposium on Foundations of Computer Science*, pp. 222-227, 1977.

[Young94] N. Young. The k-Server Dual and Loose Competitiveness for Paging. In *Proc. 2nd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 241-250 (title: "On-line caching as cache size varies"). Also *Algorithmica*, 11(6):525-541, 1994.