Algorithm Theory

Chapter 9
Online Algorithms

Part II:
Randomized Paging

Fabian Kuhn

Randomized Algorithms

- We have seen that deterministic paging algorithms cannot be better than k-competitive

- Does it help to use randomization?

Competitive Ratio: A randomized online algorithm has competitive ratio $c \geq 1$ if for all inputs I,

$$\mathbb{E}[\mathbf{ALG}(\boldsymbol{I})] \leq \boldsymbol{c} \cdot \mathbf{OPT}(\boldsymbol{I}) + \boldsymbol{\alpha}.$$

- If $\alpha \leq 0$, we say that ALG is strictly c-competitive.

Adversaries

- For randomized algorithm, we need to distinguish between different kinds of adversaries (providing the input)

Oblivious Adversary:

- Has to determine the complete input sequence before the algorithm starts
  - The adversary cannot adapt to random decisions of the algorithm

Adaptive Adversary:

- The input sequence is constructed during the execution
- When determining the next input, the adversary knows how the algorithm reacted to the previous inputs
- Input sequence depends on the random behavior of the alg.
- Sometimes, two adaptive adversaries are distinguished
  - offline, online : different way of measuring the adversary cost

Lower Bound

The adversaries can be ordered according to their strength

$$\text{oblivious} < \text{adaptive online} < \text{adaptive offline}$$

- An algorithm that achieves a given comp. ratio with an adaptive adversary is at least as good with an oblivious one
- A lower bound that holds against an oblivious adversary also holds for the adaptive adversaries

Theorem: No randomized paging algorithm can be better than k-competitive against an adaptive offline adversary.

Proof: The same proof as for deterministic algorithms works.

- For an adaptive online algorithm, a similar lower bound holds.

Are there better algorithms with an **oblivious adversary**?

The Randomized Marking Algorithm

- Every entry in fast memory has a marked flag

- Initially, all entries are unmarked.
- If a page in fast memory is accessed, it gets marked
- When a page fault occurs:
  - If all k pages in fast memory are marked, all marked bits are set to 0
  - The page to be evicted is chosen uniformly at random among the unmarked pages
  - The marked bit of the new page in fast memory is set to 1

Example

Input Sequence (k=6):

$$\underbrace{3,5,3,9,6,8,2,9,5}_{\text{phase 1}},\underbrace{7,1,2,5,2,3,7,4}_{\text{phase 2}},\underbrace{8,1,2,7,5,3}_{\text{phase 3}},\underbrace{6,9,6,10,4,1,2}_{\text{phase 4}}\ldots$$

Fast Memory:

10	**6**	**1**	**9**	**4**	**2**

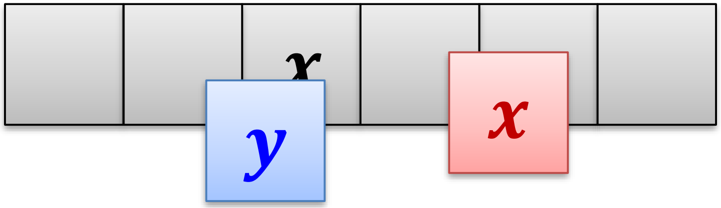
Observations:

- At the end of a phase, the fast memory entries are exactly the k pages of that phase
- At the beginning of a phase, all entries get unmarked
- #page faults depends on #new pages in a phase

Page Faults per Phase

Consider a fixed phase i:

- Assume that of the k pages of phase i, m_i are new and $k - m_i$ are old (i.e., they already appear in phase $i - 1$)
- All m_i new pages lead to page faults (when they are requested for the first time)
- When requested for the first time, an old page leads to a page fault, if the page was evicted in one of the previous page faults



- We need to count the number of page faults for old pages

Page Faults per Phase

Phase $\boldsymbol{i}$, $\boldsymbol{j}^{\text{th}}$ old page that is requested (for the first time):

- There is a page fault if the page has been evicted
- There have been at most $m_i + j - 1$ distinct requests before
- The old places of the $j - 1$ first old pages are occupied (marked)
- The other $\leq m_i$ pages are at uniformly random places among the remaining $k - (j - 1)$ places (oblivious adv.)
- Probability that the old place of the j^{th} old page is taken:

$$\leq \frac{m_i}{k - (j - 1)}$$

Page Faults per Phase

Phase $i > 1$, j^{th} old page that is requested (for the first time):

- Probability that there is a page fault:

$$\leq \frac{m_i}{k-(j-1)}$$

Number of page faults for old pages in phase i: F_i

$F_{ij} = 1 \Leftrightarrow j^{\text{th}}$ old page incurs page fault $\Leftrightarrow \mathbb{E}[F_{ij}] = \mathbb{P}(F_{ij} = 1)$

$$\mathbb{E}[F_i] = \sum_{j=1}^{k-m_i} \mathbb{P}\big(j^{\text{th}} \text{ old page incurs page fault}\big)$$

$$\leq \sum_{j=1}^{k-m_i} \frac{m_i}{k-(j-1)} = m_i \cdot \sum_{\ell=m_i+1}^{k} \frac{1}{\ell}$$

$$= m_i \cdot \big(H(k) - H(m_i)\big) \leq m_i \cdot (H(k) - 1)$$

Competitive Ratio

Theorem: Against an oblivious adversary, the randomized marking algorithm has a competitive ratio of at most $2H(k) \leq 2\ln(k) + 2$.

Proof:

- Assume that there are p phases
- #page faults of rand. marking algorithm in phase i: $F_i + m_i$
- We have seen that

$$\mathbb{E}[F_i] \leq m_i \cdot (H(k) - 1) \leq m_i \cdot \ln(k)$$

- Let F be the total number of page faults of the algorithm:

$$\mathbb{E}[F] \leq \sum_{i=1}^{p} (\mathbb{E}[F_i] + m_i) \leq H(k) \cdot \sum_{i=1}^{p} m_i$$

Competitive Ratio

Theorem: Against an oblivious adversary, the randomized marking algorithm has a competitive ratio of at most $2H(k) \leq 2\ln(k) + 2$.

Proof:

- Let F_i^* be the number of page faults in phase i in an opt. exec.
- Phase 1: m_1 pages have to be replaced → $F_1^* \geq m_1$
- Phase $i > 1$:
  - Number of distinct page requests in phases $i - 1$ and i: $\boldsymbol{k + m_i}$
  - Therefore, $\boldsymbol{F_{i-1}^* + F_i^* \geq m_i}$
- Total number of page faults F^*:

$$F^* = \sum_{i=1}^{p} F_i^* \geq \frac{1}{2} \cdot \left(\underbrace{F_1^*}_{\geq m_1} + \sum_{i=2}^{p} \underbrace{(F_{i-1}^* + F_i^*)}_{\geq m_i} \right) \geq \frac{1}{2} \cdot \sum_{i=1}^{p} m_i$$

Competitive Ratio

Theorem: Against an oblivious adversary, the randomized marking algorithm has a competitive ratio of at most $2H(k) \leq 2\ln(k) + 2$.

Proof:

- Randomized marking algorithm:

$$\mathbb{E}[F] \leq H(k) \cdot \sum_{i=1}^{p} m_i$$

- Optimal algorithm:

$$F^* \geq \frac{1}{2} \cdot \sum_{i=1}^{p} m_i$$

Randomized Lower Bound

Yao's Principle (more precisely Yao's Minimax Principle)**:**

exp. cost of best randomized alg. for worst-case input

$$\geq$$

exp. cost of best deterministic alg. for a given random input distr.

Proving a lower bound using Yao's principle:

- Design a random input distribution
- Show that every deterministic algorithm has a bad expected competitive ratio if the input is chosen at random according to this distribution
- Yao's principle then implies that every randomized algorithm is at least equally bad for a fixed worst-case input
  - worst-case fixed input: holds even for oblivious adversary

Randomized Paging Lower Bound

Input Distribution

- There are $k+1$ different pages in the slow memory
- In each step, a uniformly random page is requested

Deterministic Online Algorithms

- Consider some request i
  - Current state of the fast memory depends on requests $i-1$ and on the algorithm, assume that page p is **not** in fast memory
  - $\mathbb{P}(\text{page fault}) = \mathbb{P}(\text{request for page } p) = \frac{1}{k+1}$
- Expected #page faults after n requests:

$$\frac{n}{k+1}$$

Randomized Paging Lower Bound

Best Offline Algorithm: Longest Forward Distance

- After each page fault, optimal offline algorithm loads the page that will not be used for the longest possible time
- After a page fault, all $k+1$ pages are requested at least once before the next page fault

time between two page faults
$=$
time to request each page at least once -1

Claim: If T = time to request each page once, then

$$\mathbb{E}[\boldsymbol{T}] = (\boldsymbol{k}+\mathbf{1}) \cdot \boldsymbol{H}(\boldsymbol{k}+\mathbf{1})$$

- Probability for req. i^{th} page after requesting $i-1$ diff. pages:

$$p_i = \frac{k+1-(i-1)}{k+1}$$

Randomized Paging Lower Bound

Claim: If T = time to request each page once, then

$$\mathbb{E}[\boldsymbol{T}] = (\boldsymbol{k}+\mathbf{1}) \cdot \boldsymbol{H}(\boldsymbol{k}+\mathbf{1})$$

- Prob. for req. i^{th} page after req. $i-1$ diff. pages: $p_i = \frac{k+1-(i-1)}{k+1}$
- For $i \in \{1, \dots, k+1\}$: T_i time to request i^{th} page after requesting $i-1$ different pages

$$\boldsymbol{T_i} \sim \mathbf{Geom}(\boldsymbol{p_i}) \implies \mathbb{E}[T_i] = \frac{1}{p_i} = \frac{k+1}{k+1-(i-1)}$$

$$\boldsymbol{T} = \boldsymbol{T_1} + \cdots + \boldsymbol{T_{k+1}} : \mathbb{E}[T] = \mathbb{E}[T_1] + \cdots + \mathbb{E}[T_{k+1}]$$

$$\mathbb{E}[T] = \sum_{i=1}^{k+1} \mathbb{E}[T_i] = (k+1) \cdot \sum_{i=1}^{k+1} \frac{1}{k+1-(i-1)}$$

$$= (k+1) \cdot \sum_{j=1}^{k+1} \frac{1}{j} = (k+1) \cdot H(k+1)$$

Randomized Paging Lower Bound

Theorem: Every randomized paging algorithm has competitive ratio at least $H(k)$ even for an oblivious adversary.

- Assume we $k+1$ pages and uniformly random page requests
- Consider the phase partition from before
- Optimal offline algorithm has exactly one page fault per phase.
- Expected length of a phase : $(k+1) \cdot H(k+1) - 1$
- Expected number of page faults of any online algorithm per phase is at least

$$\frac{(k+1) \cdot H(k+1) - 1}{k+1} = H(k+1) - \frac{1}{k+1} = H(k)$$

- Now, the lower bound follows from Yao's principle.