Randomised Online Algorithms

Shawn Tan

November 16, 2011

Table of Contents

What is an Online Algorithm?

1. Receives inputs in parts or *requests*
2. Services or answers each request before going to the next one
3. Does not have overall view of entire request sequence

What is an Online Algorithm?

1. Receives inputs in parts or *requests*
2. Services or answers each request before going to the next one
3. Does not have overall view of entire request sequence
4. Examples of online algorithms:
    1. Memory paging
    2. Data structures
    3. Resource allocation

How do we analyse them?

Same method used for analysing offline algorithms cannot be used here!

Competitve analysis is used:

- Difficult to have absolute performance measure for online algorithms
- Compare against an optimal algorithm

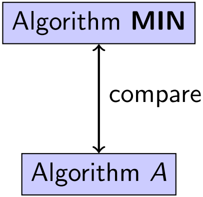
How do we analyse them?

Same method used for analysing offline algorithms cannot be used here!

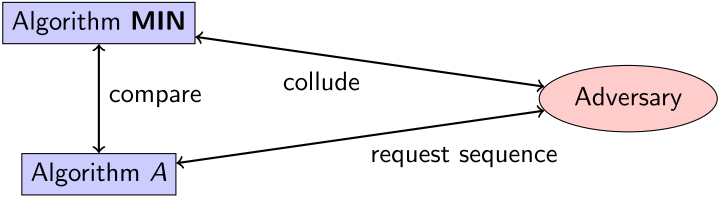
Competitve analysis is used:

- Difficult to have absolute performance measure for online algorithms
- Compare against an optimal algorithm
- Imagine comparing how fast you run compared to Usain Bolt!

Adversarial Models



Adversarial Models



Types of adversaries:

Oblivious Online Adversary Knows about the algorithm used to perform task, but not results of randomisation.

Adaptive Online Adversary Knows past answers to requests, but not results of randomisation.

Adaptive Offline Adversary Knows **everything**, including randomisation results.

Table of Contents

Deterministic Online Algorithms for Paging

Some examples of algorithms with fixed rules for paging are:

- **LRU** Least Recently Used
- **FIFO** First in, First out
- **LFU** Least Frequently Used

Some Notations

We use the symbol $\mathcal{C}$ when we compare the ratio of the cost of an algorithm with the optimal. As in:

$$\textsc{Cost}_A(\boldsymbol{\rho}) \leq \mathcal{C}_A^{ADV} \times \textsc{Cost}_{MIN}(\boldsymbol{\rho}) + b$$

Worst-case analysis

Lemma

Worst-case number of misses for any deterministic online algorithm is N, where N is the length of the request sequence.

Consider an Adaptive Offline Adversary who knows at any moment, which of the $k+1$ pages is not in the cache, and simply makes that the next request. This results in the algorithm doing a page swap at every request.

Worst case analysis

Lemma

For the offline paging algorithm **MIN**, *worst-case number of misses is* $\frac{N}{k}$

Partition some request sequence into *rounds* such that there are only k distinct requests per round.

$$\underbrace{a, \dots, b, \dots, c, \dots, d}_{\text{one round}}, e, \dots, b, \dots, a, \dots, d$$

Then for every round, **MIN** only misses once.

Lower-bound for Deterministic Online Algorithms

Theorem

Let A be a deterministic online algorithm for paging. Then $C_A \geq k$

Proof.

From the first result, we know that we can construct a series of requests that causes A to miss on every request. Then A misses more than k times per round.
From the second result, we know that the **MIN** only misses once a round. The result follows since A misses at least k more times a round compared to **MIN**. □

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $\mathcal{C}_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

The Yao's Minimax theorem tells us,

$$\inf_R \mathcal{C}_R^{obl} = \sup_{\mathcal{P}} \inf_A \mathcal{C}_A^{\mathcal{P}}$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $\mathcal{C}_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

The Yao's Minimax theorem tells us,

$$\inf_R \mathcal{C}_R^{obl} = \text{best deterministic algorithm under worst case request sequence}$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

Construct a request sequence such that each request is uniformly chosen at random from the set of pages such that the current page is not the same as the previous (k choices).
We know that **MIN** faults once in a round.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}, e, \ldots$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

Construct a request sequence such that each request is uniformly chosen at random from the set of pages such that the current page is not the same as the previous (k choices).
We know that **MIN** faults once in a round.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{\text{one round}}, e, \ldots$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

Construct a request sequence such that each request is uniformly chosen at random from the set of pages such that the current page is not the same as the previous (k choices).
We know that **MIN** faults once in a round.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{\text{length?}}, e, \ldots$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

Construct a request sequence such that each request is uniformly chosen at random from the set of pages such that the current page is not the same as the previous (k choices).
We know that **MIN** faults once in a round.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{kH_k}, e, \ldots$$

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

$$\ldots, b, \quad \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{\text{probability of missing a page?}} \quad , e, \ldots$$

□

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{\frac{1}{k} \text{ per request}}, e, \ldots$$

We know that for each request, there are k possibilities, and that there is only 1 item **not** in the cache, so the probability for missing is $\frac{1}{k}$ □

Lower bound for Randomised Paging Algorithms

Theorem

Let R be a randomised algorithm for paging. Then $C_R^{obl} \geq H_k$ where H_k is the kth Harmonic number

Proof.

$$\ldots, b, \underbrace{a, \ldots, b, \ldots, c, \ldots, d}_{E(\text{misses}) = kH_k/k = H_k}, e, \ldots$$

We know that for each request, there are k possibilities, and that there is only 1 item **not** in the cache, so the probability for missing is $\frac{1}{k}$
Since **MIN** only misses once per round, we have the result. □

The **Marker** Algorithm

a	b	c	d
0	0	0	0

The **Marker** Algorithm

a	b	c	d
1	0	0	0

a

The **Marker** Algorithm

a	b	c	d
1	0	1	0

a, c

The **Marker** Algorithm

a	b	c	e
1	0	1	1

a, c, e

Randomly selects page to evict, marks the location, and brings in new page.

The **Marker** Algorithm

a	b	c	e
1	1	1	1

$$a, c, e, b$$

Randomly selects page to evict, marks the location, and brings in new page.

The **Marker** Algorithm

a	b	c	e
0	0	0	0

$$a, c, e, b$$

Randomly selects page to evict, marks the location, and brings in new page.
Resets just after a miss, before bringing in new page.

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

- A page is considered marked if its location was marked.
- A *clean* page is an unmarked page that was unmarked in the previous round.
- A *stale* page is a currently unmarked page that was marked in the previous round.
- $d_I = |S_{\mathbf{OPT}} - S_M|$ at the beginning of the phase
- $d_F = |S_{\mathbf{OPT}} - S_M|$ at the end of the phase
- Let the number of requests to clean items be c

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

- Number of misses made by **OPT** is at least $c - d_l$

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

- Number of misses made by **OPT** is at least $c - d_I$
- Number of misses made by **OPT** is at least d_F

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

- Number of misses made by **OPT** is at least $c - d_I$
- Number of misses made by **OPT** is at least d_F

So we have,

$$\text{No. of misses} \geq \max\{c - d_I, d_F\} \geq \frac{c - d_I + d_F}{2}$$

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

Summing over all rounds, we have,

$$\ldots + \frac{c - d_I + d_F}{2} + \frac{c - d_I + d_F}{2} + \ldots$$

Analysis of **Marker** Algorithm

Theorem

*The **Marker** algorithm is $2H_k$-competitive.*

Proof.

Summing over all rounds, we have,

$$\ldots + \frac{c}{2} + \frac{c}{2} + \ldots$$

and the terms d_F and d_I telescope for consecutive rounds. So we have the number of misses a round for the offline algorithm is at least $\frac{c}{2}$

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

- There are c to clean items and $k - c$ requests to stale items.
- To maximise number of misses, let requests to clean items come first.
- Then expected number of requests to stale pages at time i of the round given by

$$0 * \frac{s_i - c_i}{s_i} + 1 * \frac{c_i}{s_i} = \frac{c_i}{s_i} \leq \frac{c}{k - i + 1}$$

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

Then the expected cost is given by,

$$\underbrace{c}_{\text{clean}} + \underbrace{\sum_{i=1}^{k-c} \frac{c}{k-i+1}}_{\text{stale}}$$

□

Analysis of **Marker** Algorithm

Theorem

The **Marker** *algorithm is* $2H_k$*-competitive.*

Proof.

Then the expected cost is given by,

$$\underbrace{c}_{\text{clean}} + \underbrace{\sum_{i=1}^{k-c} \frac{c}{k-i+1}}_{\text{stale}} = c + c(H_k - H_c) \leq cH_k$$

OPT incurs at least $\frac{c}{2}$, while **Marker** incurs at most cH_k. From this, we obtain the result. $\square$

The **Reciprocal** algorithm

The **Reciprocal** algorithm evicts a page from the cache with probability

$$p_i = \frac{1/w(x_i)}{\sum_{x \in S_i^R} 1/w(x)}$$

where S_i^R is the pages in the algorithm R's cache at the time i. $w(x)$ is the weight incurred when a page is brought *into* the cache.

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

$$S_i^R = \text{ items in cache of } \textbf{Reciprocal}$$

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

$$S_i^R = \text{ items in cache of } \textbf{Reciprocal}$$
$$S_i^{ADV} = \text{items in cache of Adaptive Online Adversary}$$

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

$$S_i^R = \text{ items in cache of } \textbf{Reciprocal}$$

$$S_i^{ADV} = \text{items in cache of Adaptive Online Adversary}$$

$$\Phi_i = \sum_{x \in S_i^R} w(x) - k \sum_{x \in S_i^R - S_i^{ADV}} w(x)$$

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

$$S_i^R = \text{ items in cache of } \textbf{Reciprocal}$$
$$S_i^{ADV} = \text{items in cache of Adaptive Online Adversary}$$
$$\Phi_i = \sum_{x \in S_i^R} w(x) - k \sum_{x \in S_i^R - S_i^{ADV}} w(x)$$
$$\Delta\Phi_i = \Phi_i - \Phi_{i-1}$$

Competitive Analysis of **Reciprocal** algorithm

Making use of a potential function,

$$S_i^R = \text{ items in cache of } \textbf{Reciprocal}$$

$$S_i^{ADV} = \text{items in cache of Adaptive Online Adversary}$$

$$\Phi_i = \sum_{x \in S_i^R} w(x) - k \sum_{x \in S_i^R - S_i^{ADV}} w(x)$$

$$\Delta\Phi_i = \Phi_i - \Phi_{i-1}$$

$$X_i = \underbrace{f_i^R}_{\text{brought in item cost}} - k \underbrace{f_i^{ADV}}_{\text{evicted item cost}} - \Delta\Phi_i$$

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Adversary

- Brings x into the cache, and evicts x'

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Adversary

- Brings x into the cache, and evicts x'
- $f_i^{ADV} = w(x')$

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Adversary

- Brings x into the cache, and evicts x'
- $f_i^{ADV} = w(x')$
- $\Delta\Phi \geq -kw(x') = -kf_j^{ADV}$, ADV only deducts from the 'bank'

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Adversary

- Brings x into the cache, and evicts x'
- $f_i^{ADV} = w(x')$
- $\Delta\Phi \geq -kw(x') = -kf_j^{ADV}$, ADV only deducts from the 'bank'

Reciprocal

Just before **Reciprocal** does anything, $|S_i^R - S_i^{ADV}| \geq 1$. Substituting,

$$\mathbf{E}[\Delta\Phi] = w(x) - \frac{k}{\sum_{y \in S_i^R} 1/w(y)} + k\frac{|S_i^R - S_i^{ADV}|}{\sum_{y \in S_i^R} 1/w(y)}$$

Looking at,

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

Adversary

- Brings x into the cache, and evicts x'
- $f_i^{ADV} = w(x')$
- $\Delta\Phi \geq -kw(x') = -kf_j^{ADV}$, ADV only deducts from the 'bank'

Reciprocal

Just before **Reciprocal** does anything, $|S_i^R - S_i^{ADV}| \geq 1$. Substituting,

$$\mathbf{E}[\Delta\Phi] = w(x) - \frac{k}{\sum_{y \in S_i^R} 1/w(y)} + k\frac{|S_i^R - S_i^{ADV}|}{\sum_{y \in S_i^R} 1/w(y)}$$

Since $f_i^R = w(x)$Then we have that **Reciprocal** also only deducts from the 'bank'

Theorem

The **Reciprocal** *algorithm is k-competitive against any adaptive online adversary.*

Proof.

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

□

Theorem

The **Reciprocal** *algorithm is k-competitive against any adaptive online adversary.*

Proof.

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

- From the contributions of the adversary and **Reciprocal**: $\mathbf{E}[\sum X_i] \leq 0$

Theorem

The **Reciprocal** *algorithm is k-competitive against any adaptive online adversary.*

Proof.

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

- From the contributions of the adversary and **Reciprocal**: $\mathbf{E}[\sum X_i] \leq 0$
- Terms of $\Delta\Phi$ telescope

□

Theorem

The **Reciprocal** *algorithm is k-competitive against any adaptive online adversary.*

Proof.

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

- From the contributions of the adversary and **Reciprocal**: $\mathbf{E}[\sum X_i] \leq 0$
- Terms of $\Delta\Phi$ telescope
- Φ_0 and Φ_n are bounded

□

Theorem

The **Reciprocal** *algorithm is k-competitive against any adaptive online adversary.*

Proof.

$$\sum_{j=1}^{i} X_j = \sum_{j=1}^{i} f_j^R - kf_j^{ADV} - \Delta\Phi_j$$

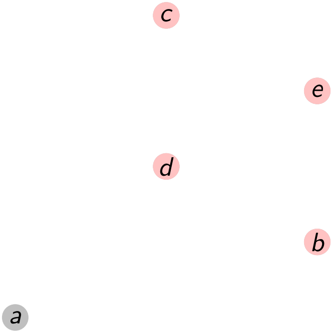
- From the contributions of the adversary and **Reciprocal**: $\mathbf{E}[\sum X_i] \leq 0$
- Terms of $\Delta\Phi$ telescope
- Φ_0 and Φ_n are bounded

$$\sum_i \left(\mathbf{E}[f_i^R] - k\mathbf{E}[f_i^{ADV}]\right) \leq \text{some constant } b$$

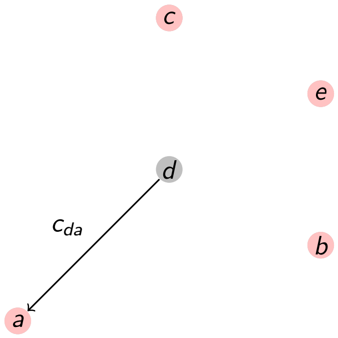
Which gives us our result. $\square$

Table of Contents

The *k*-Server Problem



The k-Server Problem



Why is it important?

Why is it important?

- Generalised version of paging problem

Why is it important?

- Generalised version of paging problem
- Resource allocation problems:
    - Motion of two-headed disks
    - Maintenance of data structures

Why is it important?

- Generalised version of paging problem
- Resource allocation problems:
  - Motion of two-headed disks
  - Maintenance of data structures

However, it is still an open problem.

Lower bound against Adaptive Online Adversary I

Theorem

Let R be a randomised online algorithm that manages k servers in any metric space. Then $C_R^{aon} \geq k$.

Lower bound against Adaptive Online Adversary

R	a	b	c	d
B_1	b	c	d	e
B_2	a	c	d	e
B_3	a	b	d	e
B_4	a	b	c	e

Lower bound against Adaptive Online Adversary

R	a	b	c	e

B_1	b	c	d	e
B_2	a	c	d	e
B_3	a	b	d	e
B_4	a	b	c	e

$e,$

Lower bound against Adaptive Online Adversary

R	d	b	c	e
B_1	b	c	d	e
B_2	a	c	d	e
B_3	a	b	d	e
B_4	a	b	c	d

$e, d,$

Lower bound against Adaptive Online Adversary

R	d	b	a	e
B_1	b	c	a	e
B_2	a	c	d	e
B_3	a	b	d	e
B_4	a	b	c	d

$e, d, a,$

Lower bound against Adaptive Online Adversary

R	d	c	a	e
B_1	b	c	a	e
B_2	a	c	d	e
B_3	c	b	d	e
B_4	a	b	c	d

$$e, d, a, c$$

Lower bound against Adaptive Online Adversary

Theorem

Let R be a randomised online algorithm that manages k servers in any metric space. Then $\mathcal{C}_R^{aon} \geq k$.

Proof.

There are k algorithms $B_1, \ldots, B_k$ such that

$$\textsc{Cost}_R(\boldsymbol{\rho}_{ADV}) = \sum_{j=1}^{k} \textsc{Cost}_{B_j}(\boldsymbol{\rho}_{ADV})$$

□

Lower bound against Adaptive Online Adversary

Theorem

Let R be a randomised online algorithm that manages k servers in any metric space. Then $\mathcal{C}_R^{aon} \geq k$.

Proof.

There are k algorithms $B_1, \ldots, B_k$ such that

$$\begin{aligned} \textsc{Cost}_R(\boldsymbol{\rho}_{ADV}) &= \sum_{j=1}^{k} \textsc{Cost}_{B_j}(\boldsymbol{\rho}_{ADV}) \\ &\geq k \min_j \textsc{Cost}_{B_j}(\boldsymbol{\rho}_{ADV}) \end{aligned}$$

And we have the result. $\square$

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary
- Deterministic lower-bound k-competitive

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary
- Deterministic lower-bound k-competitive
- Randomised lower-bound H_k-competitive

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary
- Deterministic lower-bound k-competitive
- Randomised lower-bound H_k-competitive
  - **Marker** against oblivious adversary $2H_k$-competitive
  - **Reciprocal** against adaptive online adversary k-competitive

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary
- Deterministic lower-bound k-competitive
- Randomised lower-bound H_k-competitive
  - **Marker** against oblivious adversary $2H_k$-competitive
  - **Reciprocal** against adaptive online adversary k-competitive
- k-Server Problem
  - Lower-bound k-competitive against adaptive online adversary

Summary

- Adversary Models
  - Oblivious Adversary
  - Adaptive Online Adversary
  - Adaptive Offline Adversary
- Deterministic lower-bound k-competitive
- Randomised lower-bound H_k-competitive
  - **Marker** against oblivious adversary $2H_k$-competitive
  - **Reciprocal** against adaptive online adversary k-competitive
- k-Server Problem
  - Lower-bound k-competitive against adaptive online adversary