15-850: Advanced Algorithms **CMU, Fall 2018**
Lecture #24: Online Algorithms October 24, 2018
Lecturer: Anupam Gupta Scribe: Thomas Lavastida

In this lecture we introduce the field of online algorithms by studying two classic online problems. While the models we consider will be reminiscent of those we saw when studying regret minimization in the context of the experts framework and online convex optimization, there are some important differences. These differences will lend a different flavor to the results that are obtained.

1 Online Algorithms

In the standard setting of online algorithms there is a sequence of requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_t, \ldots)$ that is received online. An online algorithm must serve request σ_i before it is allowed to see σ_{i+1}. Moreover, some cost may be incurred by the online algorithm when it serves request σ_i. We will measure the performance of an algorithm by considering the competitive ratio that we will soon define. Before we do this it will be illustrative to see an example of an online problem.

1.1 The Paging Problem

The paging problem arises in computer memory systems. Often times a memory system consists of a slow but large memory as well as a small but fast memory called a cache. The CPU typically communicates directly with the cache, so in order to access an item that is not contained in the cache, the memory system will have to send the item from the slow memory to the cache. Moreover, if the cache is full then some other item contained in the cache will have to be evicted. We say that a cache miss occurs whenever there is a request to an item that is not currently contained within the cache. The goal is to come up with an eviction strategy that minimizes the number of cache misses. Typically we do not know the future requests that the CPU will make so it is sensible to model this as an online problem.

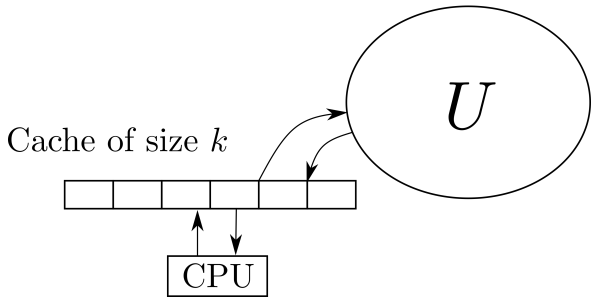

Figure 24.1: Illustration of the Paging Problem

We let U be a universe of n items or pages. The cache is a memory containing at most k pages. The requests are pages $\sigma_i \in U$ and the online algorithm is an eviction policy. Now we return back to defining the performance of an online algorithm.

1.2 The Competitive Ratio

As we said before, the online algorithm incurs some cost as it serves each request. If the complete request sequence is σ, then we let $ALG(\sigma)$ be the total cost incurred by the online algorithm in serving σ. Similarly, we let $OPT(\sigma)$ be the optimal cost in hindsight of serving σ. Note that

$OPT(\sigma)$ represents the cost of an optimal *offline* algorithm that knows the full sequence of requests. Now we define the competitive ratio of an algorithm to be:

$$\max_{\sigma} \frac{ALG(\sigma)}{OPT(\sigma)}$$

In some sense this is an "apples to oranges" comparison, since the online algorithm does not know the full sequence of requests.

Now for some more jargon. We say an online algorithm is (strictly) α-competitive if for all request sequences σ, $ALG(\sigma) \leq \alpha \cdot OPT(\sigma)$. Sometimes it may be useful to give some extra slack to the online algorithm, in which case we say that an algorithm is α-weakly competitive if there exists a constant c such that for all request sequences σ, $ALG(\sigma) \leq \alpha \cdot OPT(\sigma) + c$.

1.2.1 What About Randomized Algorithms?

The above definitions generally hold for deterministic algorithms, so how should we characterize randomized algorithms. For the deterministic case we generally think about some adversary choosing the worst possible request sequence for our algorithm. For randomized algorithms we could consider either oblivious or adaptive adversaries. Oblivious adversaries fix the input sequence up front and then let the randomized algorithm process it. An adaptive adversary is allowed to see the results of the coin flips the online algorithm makes and thus adapt its request sequence. We focus on oblivious adversaries in these notes.

To define the performance of a randomized online algorithm we just consider the expected cost of the algorithm. Against an oblivious adversary, we say that a randomized online algorithm is α-competitive if for all request sequences σ, $\mathbf{E}[ALG(\sigma)] \leq \alpha \cdot OPT(\sigma)$.

2 The Ski Rental Problem: Rent or Buy?

Now that we have some theory to work with let's apply it to a simple problem. Suppose you are on a ski trip with your friends. On each day you can choose to either rent or buy skis. Renting skis costs \$1, whereas buying skis costs \$$B$ for $B > 1$. However, the benefit of buying skis is that on subsequent days you do not need to rent or buy again, just use the skis you already bought. The problem that arises is that for some mysterious reason we do not know how long the ski trip will last. On each morning we are simply told whether or not the trip will continue that day. The goal of the problem is to find a rent/buy strategy that is competitive with regards to minimizing the cost of the trip.

In the notation that we developed above, the request for the i'th day, σ_i, is either "Y" or "N" indicating whether or not the ski trip continues that day. We also now that once we see a "N" request that the request sequence has ended. For example a possible sequence might be $\sigma = (Y, Y, Y, N)$. This allows us to characterize all instances of the problem as follows. Let I_j be the sequence where the ski trip ends on day j. Suppose we knew ahead of time what instance we received, then we have that $OPT(I_j) = \min\{j, B\}$ since we can choose to either buy skis on day 1 or rent skis every day depending on which is better.

2.1 Deterministic Rent or Buy

We can classify and analyze all possible deterministic algorithms since an algorithm for this problem is simply a rule deciding when to buy skis. Let ALG_i be the algorithm that rents skis until day i, then buys on day i if the trip lasts that long. The cost on instance I_j is then $ALG_i(I_j) =$

$(i-1+B)\cdot \mathbf{1}_{\{i\leq j\}} + j \cdot \mathbf{1}_{\{i>j\}}$. What is the best deterministic algorithm from the point of view of competitive analysis? The following claims answer this question.

Claim 24.1. *The competitive ratio of algorithm ALG_B is $2-1/B$ and this is the best possible ratio for any deterministic algorithm.*

Proof. There are two cases to consider $j < B$ and $j \geq B$. For the first case, $ALG_B(I_j) = j$ and $OPT(I_j) = j$, so $ALG_B(I_j)/OPT(I_j) = 1$. In the second case, $ALG_(I_j) = 2B-1$ and $OPT(I_j) = B$, so $ALG_B(I_j)/OPT(I_j) = 2-1/B$. Thus the competitive ratio of ALG_B is

$$\max_{I_j} \frac{ALG_B(I_j)}{OPT(I_j)} = 2 - 1/B$$

Now to show that this is the best possible competitive ratio for any deterministic algorithm. Consider algorithm ALG_i. We find an instance I_j such that $ALG_i(I_j)/OPT(I_j) \geq 2-1/B$. If $i \geq B$ then we take $j = B$ so that $ALG_i(I_j) = (i-1+B)$ and $OPT(I_j) = B$ so that

$$\frac{ALG_i(I_j)}{OPT(I_j)} = \frac{i-1+B}{B} = \frac{i}{B} + 1 - \frac{1}{B} \geq 2 - \frac{1}{B}$$

Now if $i = 1$, we take $j = 1$ so that

$$\frac{ALG_i(I_j)}{OPT(I_j)} = \frac{B}{1} \geq 2$$

Since B is an integer > 1 by assumption. Now for $1 < i < B$, we take $j = \lfloor (i-1+B)/(2-1/B) \rfloor \geq 1$ so that

$$\frac{ALG_i(I_j)}{OPT(I_j)} \geq 2 - \frac{1}{B}$$

□

2.2 Randomized Rent or Buy

Can randomization improve over deterministic algorithms in terms of expected cost? We will show that this is in fact the case. So how do we design a randomized algorithm for this problem? We use the following general insight about randomized algorithms, notably that *a randomized algorithm is a distribution over deterministic algorithms.*

To keep things simple let's consider the case when $B = 4$. We construct the following table of payoffs where the rows correspond to deterministic algorithms ALG_i and the columns correspond to instances I_j.

	I_1	I_2	I_3	I_4
ALG_1	4/1	4/2	4/3	4/4
ALG_2	1/1	5/2	5/3	5/4
ALG_3	1/1	2/2	6/3	6/4
ALG_4	1/1	2/2	3/3	7/4

We do not need to put columns after $B = 4$ since if the length of the trip is greater than B we can essentially treat it as infinite length. Similarly, we do not need to put rows after $B = 4$ since buying after day B is worse than buying on day B.

We can think of the above table as a 2 player game. The row player chooses an algorithm and the column player chooses an instance, then the number in the corresponding entry indicates the loss of the row player. Thinking along the lines of the Von Neumann minimax theorem, we can consider mixed strategies for the row player to construct a randomized algorithm for the ski rental problem.

Let p_i be the probability of our randomized algorithm choosing row i. What is the expected cost of this algorithm? Suppose that the competitive ratio was at most c in expectation. The expected competitive ratio of our algorithm against each instance should be at most c, so this yields the following linear constraints.

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &\leq c \\ \frac{4p_2 + 5p_2 + 2p_3 + 2p_4}{2} &\leq c \\ \frac{4p_1 + 5p_3 + 6p_3 + 3p_4}{3} &\leq c \\ \frac{4p_1 + 5p_2 + 6p_3 + 7p_4}{4} &\leq c \end{aligned}$$

We would like to minimize c subject to $p_1 + p_2 + p_3 + p_4 = 1$ and $p_i \geq 0$. It turns out that one can do this by solving the following system of equations:

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &= c \\ 4p_2 + 5p_2 + 2p_3 + 2p_4 &= 2c \\ 4p_1 + 5p_3 + 6p_3 + 3p_4 &= 3c \\ 4p_1 + 5p_2 + 6p_3 + 7p_4 &= 4c \\ p_1 + p_2 + p_3 + p_4 &= 1 \end{aligned}$$

Solving this yields $c = \frac{1}{1-(1-1/4)^4}$ and $p_i = (3/4)^{i-1}(c/4)$ for $i = 1, 2, 3, 4$. For general B we get $c = \frac{1}{1-(1-1/B)^B} \leq \frac{e}{e-1}$. I turns out that $\frac{e}{e-1}$ is the best possible competitive ratio for any randomized algorithm for the ski rental problem. How might one prove such a result? We instead consider playing a random instance against a deterministic algorithm. By Von Neumann's minimax theorem the value of this should be the same as what we considered above. We leave it as an exercise to verify this for the case when $B = 4$.

3 The Paging Problem

Now we return to the paging problem that was introduced earlier and start by presenting a disappointing fact.

Claim 24.2. *No deterministic algorithm can be $< k$-competitive for the paging problem.*

Proof. Consider a universe with $k + 1$ pages in all. In each step the adversary requests a page not in the cache (there is always at least 1 such page). Thus the algorithm's cost over n requests is n. The optimal offline algorithm can cut losses by always evicting the item that will be next requested furthest in the future, thus it suffers a cache miss every k steps so the optimal cost will be n/k. Thus the competitive ratio of any deterministic algorithm is at least $n/(n/k) = k$. □

It is also known that many popular eviction strategies are k-competitive such as Least Recently Used (LRU) and FIFO. We will show that a 1-bit variant of LRU is k-competitive and also show that it can be randomized in order to achieve an $O(\log k)$-competitive randomized algorithm for paging.

3.1 The 1-bit LRU/Marking Algorithm

The 1-bit LRU/Marking algorithm works in phases. The algorithm maintains a single bit for each page in the universe. We say that a page is marked/unmarked if its bit is set to 1/0. At the beginning of each phase, all pages are unmarked. When a request for a page not in the cache comes, then we evict an arbitrary unmarked page and put the requested page in the cache, then mark the requested page. If there are no unmarked pages to evict, then we unmark all pages and start a new phase.

Claim 24.3. *The Marking algorithm is k-competitive for the paging problem.*

Proof. Consider the i'th phase of the algorithm. By definition of the algorithm, ALG incurs a cost of at most k during the phase since we can mark at most k different pages and hence we will have at most k cache misses in this time. Now consider the first request after the i'th phase ends. We claim that OPT has incurred at least 1 cache miss by the time of this request. This follows since we have now seen $k+1$ different pages. Now summing over all phases we see that $ALG \leq kOPT$ □

Now suppose that instead of evicting an arbitrary unmarked page, we instead evicted an unmarked page uniformly at random. For this randomized marking algorithm we can prove a much better result.

Claim 24.4. *The Randomized Marking Algorithm is $O(\log k)$-competitive*

Proof. We break up the proof into an upper bound on ALG's cost and a lower bound on OPT's cost. Before doing this we set up some notation. For the i'th phase, let S_i be the set of pages in the cache *at the beginning of the phase.* Now define $\Delta_i = |S_{i+1} \setminus S_i|$. Now let R_i be the set of distinct requests in phase i. For each $\sigma \in R_i$ we say that σ is clean if $\sigma \notin S_i$ and otherwise σ is stale.

We claim that the expected number of cache misses made by the algorithm in phase i is at most $\Delta_i(H_k+1)$. By summing over all phases we see that $\mathbf{E}[ALG] \leq \sum_i \Delta_i(H_k+1)$.

Every cache miss in the i'th phase is caused by either a clean request or a stale request. Clearly the number of cache misses due to clean requests is at most Δ_i since there can be at most Δ_i clean requests in phase i. Now to bound the cache misses due to stale requests. Say that there have been c clean requests and s stale requests so far, and consider the $s+1$'st stale request. The probability this request causes a cache miss is at most $\frac{c}{k-s}$ since we have evicted c random pages out of $k-s$ remaining stale requests. Now since $c \leq \Delta_i$, we have that the expected cost due to stale requests is at most $\sum_{s=0}^{k-1} \frac{c}{k-s} \leq \Delta_i \sum_{s=0} k - 1 \frac{1}{k-s} = \Delta_i H_k$. Now the expected total cost in phase i is at most $\Delta_i H_k + \Delta_i = \Delta_i(H_k+1)$.

Now we claim that $OPT \geq \frac{1}{2}\sum_i \Delta_i$. Let S_i^* be the pages in OPT's cache at the beginning of phase i. Let ϕ_i be the number of pages in S_i but not in OPT's cache at the beginning of phase i i.e. $\phi_i = |S_i \Delta S_i^*|$. Now let OPT_i be the cost that OPT incurs in phase i. We have that $OPT_i \geq \Delta_i - \phi_i$ since this is the number of "clean" requests that OPT sees. Moreover, consider the end of phase i. ALG has the k most recent requests in cache, but OPT does not have ϕ_{i+1} of them by definition of ϕ_{i+1}. Hence $OPT_i \geq \phi_{i+1}$. Now by averaging, $OPT_i \geq \max\{\phi_{i+1}, \Delta_i - \phi_i\} \geq \frac{1}{2}(\phi_{i+1} + \Delta_i - \phi_i)$. So summing over all phases we have $OPT \geq \frac{1}{2}\sum_i \Delta_i + \phi_{final} - \phi_{initial} \geq \frac{1}{2}\sum_i \Delta_i$ since $\phi_{final} \geq 0$ and $\phi_{initial} = 0$.

Combining the upper and lower bound yields $\mathbf{E}[ALG] \leq (2(H_k+1))OPT = O(\log k)OPT$. □

It can also be shown that no randomized algorithm can do better than $\Omega(\log k)$-competitive for the paging problem. For some intuition as to why this might be true, consider the coupon collector problem.

4 Generalizing Paging: The k-Server Problem

Another famous problem in online algorithms is the k-server problem. Let M be a metric space with distance function $d : M \times M \to \mathbb{R}_+$. The distance function for a metric space satisfies 3 properties:

1. For all $x \in M$, $d(x, x) = 0$
2. For all $x, y \in M$, $d(x, y) = d(y, x)$
3. For all $x, y, z \in M$, $d(x, z) \leq d(x, y) + d(y, z)$

where the last property is known as the triangle inequality. In the k-server problem there are k servers that are located at various points of M. Over time we receive requests $\sigma_i \in M$. If there is a server at point σ_i already, then we can server that request for free. Otherwise we move some server at point x to point σ_i and pay a cost equal to $d(x, \sigma_i)$. The goal of the problem is to serve the requests while minimizing the total cost of serving them.

The paging problem can be modelled as a k-server problem as follows. We let U be the points of the metric space and take $d(x, y) = 1$ for all pages x, y where $x \neq y$. This special case shows that every deterministic algorithm is at least k-competitive and every randomized algorithm is $\Omega(\log k)$-competitive by the discussion in the previous section. It is conjectured that there is a k-competitive deterministic algorithm but the best known result is $(2k - 1)$-competitive [KP95].

Acknowledgments

These lecture notes were scribed by Thomas Lavastida, based on previous scribe notes of Anupam Gupta.

References

[KP95] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995. 4