Online Algorithms

Lecture scribed by Tarun Agarwal

Introduction

Online Algorithms receive input and process them in partial amounts. This is different from *offline algorithms* where the entire input is available at one time. Online algorithms typically receive *service* requests and the current request must be processed before the next one is received. Servicing of a request incurs a cost depending on the way it is serviced. The objective is to optimize the total cost incurred in servicing a series of requests.

Some Examples

Cab Service Problem

Suppose there are k taxis. When a call is received requesting for a taxi, one of the taxis must be assigned to service the request. If the taxi, say t_i, is assigned to service the request r_i it has to travel a distance $d(t_i, r_i)$ to reach the point of request. The objective is to minimize $\sum_i d(t_i, r_i)$ for a sequence of requests $\{r_i\}$.

Page Replacement Problem

There is a cache of size K and let $r_1, r_2, \ldots, r_N$ be pages requested ($K << N$). If a page that is requested is already present in the cache we say that a *hit* has occured. Otherwise it must be fetched and must be put in the cache where it replaces a page based on the page replacement policy. This is called a *miss*. The cost measure for paging is the number of misses on a sequence of request. The objective is to minimize this cost.

The Online Paging Problem

Performance Measure

Traditional performance analysis like the worst case cost (number of misses) over all possible request sequences is not useful for an online paging algorithm. This is because in most cases any algorithm for processing requests can be forced to incur an unbounded cost by appropriately choosing the input sequence. Thus to compare different algorithms we define a competitive measure of performance.

Definition

Let A be a deterministic online paging algorithm. A is said to be $C-competitive$ if there exits a constant D such that on every sequence of requests $r_1, r_2, \ldots, r_N$,

$$cost_A(r_1, r_2, \ldots, r_N) - C * OPT(r_1, r_2, \ldots, r_N) \leq D \tag{1}$$

where the constant C must be independent of N but may depend on k. The *competitiveness coefficient* of A, denoted C_A is the infimum of C such that A is C-competitive.

Claim : Any deterministic online algorithm for the paging problem has $C_A \geq k$

Proof : Let us assume that there are $k+1$ distinct pages. First consider an offline algorithm which divides page requests into rounds. A round ends when the $(k+1)^{th}$ unique page is requested. It is easy to see that the offline algorithm incurs atmost one miss per round (potentially on the first request). Now let A be a deterministic online paging algorithm. Since the cache size is k exactly one page will be out of the cache at any time. Let the i^{th} request be for a page which is not present in the cache at that time. Thus A misses on every request. Since the round contains atleast k requests and A misses on every one of them, it misses atleast k times during the round. Thus $C_A \geq k$.

Claim : Any randomized online algorithm has $C_A = \Omega(H_K)$

Proof: We will use Yao's Minimax Principle for the proof. Let us consider a set of $k+1$ memory pages to prove the lower bound. Since the cache has k location exactly one of them remains out of the cache. Consider the following distribution on the page requests. The first page request is chosen uniformly at random from the set I. Let the page requested in i_{th} round be r_i The request r_{i+1} is chosen uniformly at random from the set $I - r_i$. For analysis we will divide the sequence of requests into rounds where a round ends when $(k+1)^{th}$ unique page is requested. It is easy to see that the offline algorithm incurs atmost one miss per round (atmost on the first request in a round). Let us now calculate the number of request per round. Suppose we have already encountered requests for i distinct pages. The probability that a distinct page is requested in the next request is $\frac{k+1-i}{k}$. Thus the expected number of requests made before a new unique page is requested is $\frac{k}{k+1-i}$. By linearity of expectation the number of requests per round is H_K. No consider any online deterministic algorithm A. At any point A must leave one of the $k+1$ pages out of the cache. The probability that this page is requested is $1/k$. Thus the expected number of misses by A per round is $H_K(= kH_K * 1/k)$. So we see by the arguments above that $C_A = H_K$ for the request distribution that we have considered. By the Yao's principle any randomized online algorithm cannot do better and hence the claim is proved.

Marker Algorithm

We describe a randomized online algorithm which achieves a competitive coefficient close to the lower bound given above. The algorithm proceeds in a series of rounds. Each of the k cache locations has a bit (0/1) associated with it. Initially all bits are set to 0. When a page request is made, the algorithms processes them as follows. If the page is present in the cache the bit associated with that location is set to 1 (it is marked) and the page is fetched. If however the page is not present in the cache, a cache location marked 0 is selected uniformly at random and the page at that location is replaced by the requested page. Finally the bit at that cache location is set to 1. As more requests are processed potentially all cache locations are marked. A round is considered complete when all locations are marked and a page is requested which is not already in the cache. When a round is completed all locations are unmarked again (bits set to 0) and the same procedure is followed.

Claim : The **Marker** algorithm has $C_A \leq 2H_K$.

Proof : Let OPT be the optimal offline algorithm for paging. Let C_I^M, C_I^O be the cache contents of Marker and OPT algorithms respectively at the beginning of a round and C_F^M, C_F^O be the contents at the end. A page is called *stale* if it was brought into cache in last round but not requested in present round. A page is called clean if was not brought into the cache in the last round and not requested in the present round. Let l be the number of distinct clean pages and $k - l$ stale pages requested in the round being considered. Also let

$$d_I = C_I^O - C_I^M, \quad d_F = C_F^O - C_F^M \tag{2}$$

The offline algorithm misses on atleast $l - d_I$ clean pages requested since atleast $l - d_I$ of the clean pages requested in the round are not in the offline algorithm's cache at the beginning of the round. At the end of the round all the k pages which are present in the cache of the Marker algorithm have been requested in the round. Since OPT's cache contains d_F different pages it must have incurred atleast d_F misses in this round. Thus in a round,

$$Number\ of\ misses\ by\ OPT \geq max\{l - d_I, d_F\} \geq \frac{1}{2}(l - d_I + d_F) \tag{3}$$

On summing this lower bound over all rounds the amortized number of misses by OPT per round is $\frac{l}{2}$.

Now consider the Marker algorithm. It misses on all l clean pages. Of the $k - l$ requests to stale pages, the expected cost of each is the probability that the item requested is not in the cache. This is maximized when all l clean pages' requests are made before the $k-l$ stale pages' requests. For $1 \leq i \leq k-l$, the probability that the i^{th} stale page request is a miss is $\frac{l}{k-i+1}$. Thus summing over all requests the expected number of misses by Marker algorithm per round is :

$$l + l(H_K - H_l) \leq lH_K, \tag{4}$$

Thus for the Marker algorithm $C_A \leq \frac{lH_K}{\frac{l}{2}} = 2H_K$. Hence proved.