6.854 Advanced Algorithms

Lecture 18: 11/05/2004 Lecturer: David Karger

Scribes: Chun-Chieh Lin

18.1 Lower Bounds for Competitive Ratios of Randomized Online Algorithms

Designing an online algorithm can be viewed as a game between the algorithm designer and the adversary. The algorithm designer chooses an algorithm A_i, and the adversary chooses an input σ_j. The payoff matrix contains the cost of the algorithm on the input $C_{A_i}(\sigma_j)$. The algorithm designer wants to minimize the cost, while the adversary wants to maximize the cost. A randomized online algorithm is a probability distribution over the deterministic algorithms, so it corresponds to a mixed strategy for the algorithm designer.

18.1.1 Game Theory Analysis

Von Neumann proved that for any game, there exist equilibrium (mixed) strategies for the players. At the equilibrium, neither side is able to improve (increase or decrease depending on the player) the cost any further by changing the strategy.

However, problem 4a of problem set 6 showed that if one player's mixed strategy is known (and fixed), the other player has a pure strategy as a best response. That means, if one of the players is using the equilibrium (optimal) mixed strategy, the other player has pure strategy as a best response, and the resulting cost is the equilibrium cost. Again, a pure strategy for the algorithm designer corresponds to a deterministic algorithm, and a mixed strategy for the algorithm designer corresponds to a randomized algorithm, so this leads to Yao's Minimax Principle:

Theorem 1 ***Yao's Minimax Princliple:*** *If for some input distribution no deterministic algorithm is k-competitive, then no randomized k-competitive algorithm exists.*

18.1.2 Example: Paging

Suppose there are $k+1$ pages and n accesses, and for each access, the pages all have probability $1/(k+1)$ of being requested. In other words, this is a uniform distribution over inputs with length n of the $k+1$ pages.

Online Algorithm

No matter what the online algorithm does, there are only k pages in the memory at any point in time. So with probability $1/(k+1)$, the requested page is not in the memory. Therefore, the expected number of faults over the n accesses is $n/(k+1)$, and the expected number of requests per fault is $k+1$, which is $\Theta(k)$.

Offline Algorithm

Even though the sequence of requests is still chosen at random, the offline algorithm has access to the whole sequence before it starts running.

As shown in previous lectures, an optimal algorithm for the offline algorithm is the Farthest in Future algorithm, which evicts the page that is requested farthest in the future. This algorithm faults once every $k+1$ distinct pages seen, because after each fault, the evicted page is not requested again until after all other k pages are requested.

The expected number of requests it takes to see all $k+1$ distinct pages can be calculated as follows:

E[No. requests total] $= \Sigma_{i=1}^{k+1}$ E[No. requests between the $i-1^{th}$ distinct request and the i^{th}]

P(each request after the $i-1^{th}$ distinct request is the i^{th} distinct request) $= \frac{k+2-i}{k+1}$

E[No. requests between the $i-1^{th}$ distinct request and the i^{th}] $= \frac{k+1}{k+2-i}$

E[No. requests total] $= \Sigma_{i=1}^{k+1} \frac{k+1}{k+2-i} = (k+1) * (\frac{1}{k+1} + \frac{1}{k} + \frac{1}{k-1} + \frac{1}{k-2} + ... + \frac{1}{1}) = \Theta(klogk)$

Conclusions

The expected number of pages per fault for the online algorithm is $\Theta(k)$. The expected number of pages per fault for the offline algorithm is $\Theta(klogk)$. So the ratio of fault counts is $\Theta(logk)$.

Using Yao's Minimax Principle, this shows that no randomized algorithm can have competitive ratio better than $\Theta(logk)$ for paging.