Segueing to Online Algorithms and Competitive Analysis

A Problem in Computational Finance...

- When you join Millisoft, you get a share of stock
- The stock value is guaranteed by Millisoft never to fall below value $m$ nor go above value $M$ ($M > m$) (assume real numbers)
- Each day, you see the share value and must decide whether to sell it or wait for the next day
- At some point, Millisoft will end the program with no warning and pay you $m$ for your share if you haven’t sold it yet!
- **What policy should you use to be “competitive” w.r.t. to an algorithm that knows the future?**

In economics, \( M/m \) is called the “global fluctuation ratio”

Imagine that $M = 100$ and $m = 1$ and Prof. Lai’s policy is “never sell” (unless the value reaches 100)

Imagine that $M = 100$ and $m = 1$ and Prof. Lai’s policy is “sell at 50”
The average of $M$ and $m$ doesn’t seem very good. What’s the optimal threshold?

The Paging Problem

“Fast memory” (e.g., cache) of size $k$ pages

“Slow memory” (e.g., disk) of size $P$ pages ($P >> k$)

- Request sequence
- Algorithms: LIFO, FIFO, LRU, FWF, ...
- Competitive algorithms

Robots on a line

Request 1

Request 2

Assumptions:
- $k$ identical robots. Any robot can service a request.
- Requests arrive one-at-a-time. A request must be serviced before the next request arrives.

Goal: Minimize total travel distance of all robots

Definitions:
- Let $\sigma = \sigma_1 \sigma_2 \ldots \sigma_n$ denote the request sequence
- $\text{OPT}(\sigma)$ denotes the optimal (minimum) cost incurred by an offline algorithm for this sequence
- $\text{ALG}(\sigma)$ denotes the cost paid by our online algorithm
- $\text{ALG}$ is said to be $c$-competitive if $\exists d$ s.t. $\forall \sigma$

$$\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma) + d$$

Greed is Not Competitive!

Greedy Algorithm: Always dispatch the robot closest to the request point.
While Greed is Bad, Laziness is a Virtue!

Assume that robots operate in some metric space (so the triangle inequality holds!)

Back to robots on a line...

The DC Algorithm (Marek Chrobak, UC Riverside)

DC ("Double Coverage") Algorithm: All robots that have an unobstructed view of the request, move towards it at equal speed. (But, if two or more robots are co-located, only one will move!)

(We could optionally "lazify" this algorithm so that only one robot moves on any request – but we won’t do this because it complicates the analysis that we’re about to do!)

Claim: If there are \( k \) servers (robots), DC is \( k \)-competitive!

DC seems to be better than greed!

Initially, DC and OPT have their robots in the same positions...

Plan:
- Define a function \( \Phi_i \) that measures the “difference” in DC and OPT’s configurations immediately after both algorithms have finished servicing request \( i \). \( \Phi_i = 0 \)
- Show that if OPT moves a server a distance of \( d \) then \( \Phi_i \) increases by at most \( kd \).
- Show that if DC moves its servers a total distance \( \delta \) then \( \Phi_i \) decreases by at least \( \delta \).

Why is this useful?
Defining $\Phi_i$

Consider matchings of DC and OPT’s robots...

We can't actually compute the matching in our algorithm - it’s only for the sake of analysis!

... and measure the matching by the length of the horizontal legs

$M_{\text{min}} = \text{total length of horizontal legs in the matching that minimizes that quantity. (An “optimal” matching)}$

An optimal matching has no crossings...

Plan:

- Define a function $\Phi_i$ that measures the “difference” in DC and OPT’s configurations immediately after both algorithms have finished servicing request $i$.

  $$\Phi_i = k \times M_{\text{min}} + \sum_{\text{DC}}$$

- Show that if OPT moves a server a distance of $d$ then $\Phi_i$ increases by at most $kd$.

- Show that if DC moves its servers a total distance $\delta$ then $\Phi_i$ decreases by at least $\delta$.

The sum of all the pairwise distances between DC’s servers

$$\sum_{\text{DC}}$$
Let OPT move first...

Show that if OPT moves a server a distance of $d$ then $\Phi_i$ increases by at most $kd$.

\[ \Phi_i = k \times M_{\text{min}} + \sum_{\text{DC}} \]

$\Phi_{i-1} \rightarrow \Phi_i$

If DC’s move is of distance $\delta$ then $\Phi$ decreases by at least $\delta$

$\Phi_{i-1} \rightarrow \Phi_i$

\[ \Phi_i = k \times M_{\text{min}} + \sum_{\text{DC}} \]

Case 1: DC moves a single server...

Case 2: DC moves two servers...

If DC’s move is of distance $\delta$ then $\Phi$ decreases by at least $\delta$

$\Phi_{i-1} \rightarrow \Phi_i$

\[ \Phi_i = k \times M_{\text{min}} + \sum_{\text{DC}} \]

From robots back to paging...

"Fast memory" (e.g., cache) of size $k$ pages

"Slow memory" (e.g., disk) of size $P$ pages ($P \gg k$)

- Request sequence
- Algorithms: LIFO, FIFO, LRU, FWF, ...
- Competitive algorithms
Related Research

- Is it possible to do better than $k$-competitive on a line or a tree?
- Can we get $k$-competitive algorithms for other metric spaces?