COMP 7720 - Online Algorithms

Paging and $k$-Server Problem

Shahin Kamali

Lecture 11 - Oct. 11, 2018

University of Manitoba
Review & Plan
Today’s objectives

- $k$-server problem
  - Paths & trees
  - Balancing algorithms
  - The case of $k = 2$
  - Randomized algorithm for cycles
$k$-Server Problem
We have a metric space of size $m$

- $k < m$ servers in the graph

A sequence of $n$ requests to the vertices of the graph

- Each request should be served by a server

Minimize the total distance moved by servers

$\sigma = < S \ M \ K \ A \ D \ B \ D \ B \ D >$

costs $= 2 \ 0 \ 2 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1$
Major Results

**Theorem**

For any metric $G$, no deterministic $k$-server algorithm $\text{Alg}$ can have a competitive ratio smaller than $k$.

**Conjecture**

Conjecture: for any metric space, there is a deterministic algorithm with competitive ratio of $k$.

- $k$-server conjecture is one of the big open problems in the context of online algorithms.
  - Verified for $k = 2$, $m = k + 1$, $m = k + 2$, paths and trees.
Double Coverage Algorithm (DCA) for Paths

- On a request to $x$:
  - Move the closest server on left and closest server on right at the same ‘speed’ toward $x$ until one meets $x$.
  - If the closest server is at distance $d$, the algorithm incurs a cost of $2d$.
  - If there is no server on left (or right), just move the closest server!

**Cost:** $4 + 2 + 1 + 2$
The double coverage algorithm (DCA) has a competitive ratio of $k$ for paths. So, it is the optimal deterministic algorithm for paths. For the proof, we used the potential function method 😊.
Lazy Algorithms

- An algorithm is called **lazy** if it moves at most one server to serve each request.

- Is DCA a lazy algorithm?
  - No, it might move two servers.
Lazy Algorithms

Theorem

Any non-lazy algorithm $A$ can be converted to a lazy algorithm $A'$ without increasing its cost.

- In $A'$, for each server, maintain a real position and a virtual position.
- Virtual positions are maintained similar to $A$.
- When $A$ moves $p$ servers for a request to node $x$:
  - Only update the real position of one server that arrives to $x$.
  - We ‘delay’ moving other servers.

$A'$ saved a distance of 2 on moves of server 3!
Double Coverage Algorithm for Trees

- Move servers that have no other serve between them and the request
  - Move servers with equal speed to the requested sequence
  - Stop when any server arrives to the requested vertex

**Theorem**

*Double-Coverage algorithm (DCA) has a competitive ratio of $k$ for trees.*

- Similar potential & proof as in paths!
- The $k$-server conjecture is true (via DCA) for paths & trees
Revisiting Paging

- Recall that $k$-server becomes equal to caching problem when the metric is **uniform**
  - When distance between vertices associated with pages (yellow vertices) is the same.
- We can **embed** a complete graph into a **star tree**
  - So that the distances remain the same between pages (yellow vertices)
- What is the double-coverage algorithm for star? (paging)
  - It will be Flash-When-Full (FWF)
  - Another proof that FWF has competitive ratio $k$.
  - Note that FWF can be implemented in a lazy fashion!
Double Coverage Algorithm for $k = 2$

- When we have $k = 2$, we can use a version of double-coverage algorithm.
- On a request to $x$, consider a ‘red spider’ that embeds shortest distances of servers and request.
- Apply DCA using the red spider (move servers on the star edges).
- In reality, we cannot move on the star (since it is not a part of graph).
  - Use a lazy variant; star positions are virtual positions; in reality only one server is moved.

request to $y = D$
Double Coverage Algorithm (DCA) for $k = 2$ & $k = 3$

- Why DCA has a competitive ratio of $k$ when $k = 2$ and unbounded competitive ratio for $k = 3$? (intuition)
- When $k = 2$, the triangle formed by the two servers & the requested node can be embedded into a tree.
- When $k = 3$, the graph formed by the three vertices & the requested node cannot be necessarily embedded into a tree.
  - E.g., a cycle cannot be embedded into a tree
Double Coverage Algorithm (DCA) Summary

- DCA is $k$-competitive (optimal) for paths, trees, and any metric that can be embedded in trees (e.g., complete graph).
- DCA is $k$-competitive (optimal) for $k = 2$.
- DCA is not useful for $k \geq 3$ even if the metric is a cycle.
Balancing Algorithms

- Move the server which after (potentially) serving the request, has moved less than other servers

- Is it a good algorithm?
  - For \( n \) requests, \( \text{cost}(\text{Balance}) = n \cdot d \)
  - \( \text{cost}(\text{OPT}) = d + n \) (why?)
  - The competitive ratio of the Balance algorithm is at least \( \frac{nd}{n+d} \approx d \), which is much more than the optimal ratio of \( k = 2 \).

- Balance is \( k \)-competitive for metrics with \( k + 1 \) nodes

\[ \sigma = (D \ C \ B \ A)^n \]
Randomized algorithms

- Compare against *oblivious adversary*
  - For any metric space, no algorithm can be better than $\log k$ competitive
- Randomized $k$-server conjecture
  - For any metric space there is a randomized $\log k$-complete algorithm
- Verified for hierarchical binary trees
- For general graphs, there is a $O(\log^3 m \log^2 k)$-competitive graph
  - Better than $2k - 1$ when $m$ is sub-exponential of $k$
Randomized Algorithm CIRC for Cycle

- Select a point $P$, uniformly at random, from the cycle of length $C$.
  - Think of $P$ as a ‘road-block’ and apply DCA for the resulting segment $L$
  - This selection of $P$ is equivalent to deletion of a random edge from the cycle

**Theorem**

*CIRC is a $2k$-competitive algorithm for cycle*

- Observation: $P$ appears in the shortest path between $(A, B)$ with probability $d(A, B)/C$. 

![Diagram of a cycle with points A, B, and P marked, and distance label d(A,B) between P and another point.](image)
Randomized Algorithm CIRC for Cycle

- Let OPT-Line be the optimal offline algorithm when restricted to $L$
- We have $\text{Cost}(CIRC) \leq k \cdot \text{Cost}(\text{OPT-Line})$ (double-coverage algorithm on line)
- $\text{Cost}(\text{OPT-Line}) \leq 2\text{Cost}(\text{OPT})$
  - Assume $\text{OPT}$ makes moves of lengths $d_1, d_2, \ldots, d_n$
    - $\text{cost}(\text{opt}) = d_1 + d_2 + \ldots + d_n$
  - Apply the same moves as $\text{OPT}$; with additional penalty of at most $C$ if a server passes $P$ (the penalty means you go all the way through other side).
  - The chance of passing $P$ on a move of length $d_i$ is $d_i / C$.
    - The whole penalty is expected to be at most $d_1 / C \cdot C + d_2 / C \cdot C + \ldots + d_n / C \cdot C = \text{Cost}(\text{OPT})$.
    - The expected cost of OPT-Line is at most $d_1 + d_2 + \ldots + d_n + \text{Cost}(\text{OPT}) = 2\text{Cost}(\text{OPT})$
Randomized Algorithm CIRC for Cycle

- In summary, we have $\text{cost}(\text{CIRC}) \leq k \cdot \text{cost}(\text{OPT-Line})$ and $\text{cost}(\text{OPT-Line}) \leq 2\text{cost}(\text{OPT})$.

**Theorem**

*CIRC is a $2k$-competitive algorithm for cycle*

- Is it good?
  - Yes (it is the best existing algorithm) and No (we hope to get something around $\log k$).
  - Deterministic $k$-server conjecture is still open for cycles.

- Here, we reduced a cycle to a line segment

- This type of reduction is the main tool for analysis of randomized $k$-server
  - Reduce an arbitrary graph to a ‘hierarchical binary tree’
Announcements
Announcements

- Assignment 2 will be posted shortly
- Solutions for assignment 1 will be posted shortly
- The final exam is scheduled for Tuesday, Dec 18, 2018 from 09:00 AM to 12:00 pm.