On-Line Algorithms for the k-Server Problem

Richard Borie
University of Alabama
June 27, 2005

Concepts: a quick review

- On-line algorithms are interactive
  - Must process each request before knowing future
- An on-line algorithm $A$ is $c$-competitive if
  - For all sequences $R = (R_1, R_2, R_3, \ldots, R_m)$ of requests,
  \[ \text{cost}_A(R) \leq c \times \text{cost}_{OPT}(R) + b \]
  - OPT is an optimal off-line algorithm
  - $b$ is a constant that is independent of $R$ (usually 0)
- Adversary
  - Knows how our algorithm works, and always issues requests $R_1, R_2, R_3, \ldots, R_m$ that will make our algorithm perform as poorly as it possibly can

Examples: a quick review

- Rent or buy?
  - $(2 - r/p)$-competitive on-line algorithm
  - $r$: cost to rent, $p$: cost to purchase, $r < p$
- Paging
  - Least Recently Used (LRU) is $k$-competitive
  - $k$: number of pages that can fit in the cache
- Self-organizing list
  - Move To Front (MTF) is $4$-competitive
- Bin packing
  - First Fit (FF) is $2$-competitive

Today's example: $k$-server problem

- Complete graph $G$ with edge weights that satisfy the triangle inequality:
  \[ w(x,z) \leq w(x,y) + w(y,z) \]
- Initially $k$ servers reside at $k$ vertices
- Each request occurs at some vertex $v$
- If no server is currently at $v$, one of the servers must move to $v$
- Generalization of paging problem
- Partially accurate model: requesting a plumber

Move Nearest Server heuristic

- Greedy heuristic (Move Nearest Server) is not $c$-competitive for any constant $c$
  - Consider complete graph on vertices $\{a,b,c\}$
  - Let $k=2$, $w(a,b)=2$, $w(a,c)=3$, and $w(b,c)=1$
  - Let $R=(a,b,c,b,c,b,c,\ldots)$ with $|R|=m$
  - $\text{cost}_{\text{Greedy}}(R) \geq m-2$
  - $\text{cost}_{\text{OPT}}(R) \leq 4$
  - Ratio $\geq (m-2)/4$ is unbounded

Not $c$-competitive when $c<k$

- For all $k$, there can be no $c$-competitive on-line algorithm for the $k$-server problem for any $c<k$
- Sketch of proof:
  - Suppose algorithm $A$ is $c$-competitive
  - Let $G$ be a graph with $k+1$ vertices
  - Adversary makes requests to keep $A$ busy
    - Let $R_i$ be vertex in $G$ with no server initially
    - For $2 \leq i \leq m$, let $R_i = \text{vertex that was abandoned to service the previous request } R_{i-1}$
Sketch of proof (continued)

- For each vertex $x \neq R_1$, define algorithm $B_x$
  - $B_x$ starts with servers at each vertex except $x$
  - Upon each request $R_i$, $B_x$ operates as follows:
    - If vertex $R_i$ is vacant, move server from $R_{i-1}$ to $R_i$
  - Claim: if $x \neq z$, then at no time do $B_x$ and $B_z$
    ever have the same vacant vertex
  - Basis: obviously true before request $R_1$
  - Induction: suppose true before $R_i$, so at most
    one $B_x$ has vertex $R_i$ vacant. Hence only this one
    $B_x$ moves in response to request $R_i$ and so only
    this one $B_x$ will have $R_{i-1}$ vacant after request $R_i$

Sketch of proof (continued)

- Hence $\Sigma_{x \neq R_1} \text{cost}_{B_x} (R) \leq \Sigma_{2 \leq i \leq m} w(R_{i-1}, R_i)$
- But $\Sigma_{2 \leq i \leq m} w(R_{i-1}, R_i) \leq \text{cost}_A (R)$
- So $\Sigma_{x \neq R_1} \text{cost}_{B_x} (R) \leq \text{cost}_A (R)$
- Note: $\min_{x \neq R_1} \text{cost}_{B_x} (R) \leq \text{cost}_A (R)/k$
- Finally $\text{cost}_A (R) \geq k*\text{cost}_{B_x} (R) \geq k*\text{cost}_{\text{OPT}} (R)$
- This proof is non-constructive (existential)
  - We don’t know which $B_x$ yields the minimum cost,
    and we didn’t state how to construct $\text{OPT}$

Sketch of proof (continued)

- What if we are not permitted to change the
  initial location of the $k$ servers?
  - $\text{cost}_{B_x} (R) \leq \text{cost}_A (R)/k + w(R_1, x)$
  - $\text{cost}_A (R) \geq k*\text{cost}_{B_x} (R) - k*w(R_1, x)$
  - So only the additive constant is affected

[ End of proof ]

k-Server Conjecture

- For all $k$, there exists a $k$-competitive
  algorithm for the $k$-server problem???
- Question was first posed in about 1988,
  and its status remains unresolved
- Probably the most important open problem
  in on-line algorithms
- In the remainder we discuss some progress
  made toward solving this problem

2-competitive for 2 servers

- There exists a 2-competitive algorithm for
  the 2-server problem
  - Result originally obtained by
    Manasse/McGeoch/Sleator in 1988
  - We describe a simpler and more general idea
    used by Chrobak/Larmore in 1990
    - Extend the graph to a metric space
      - Add virtual points that are not exactly at any vertex
      - Each such point is defined by its distance to each vertex
    - Maintain both an actual location and a virtual
      location for each server

2-server algorithm (continued)

- Suppose two servers are at virtual locations $(a,b)$,
  and the next request occurs at vertex $c$
  - If $w(a,b) + w(b,c) = w(a,c)$ then
    - Server $b$ is between $a$ and $c$
    - Move server at $b$ to $c$, and do not move server at $a$
  - If $w(a,b) + w(a,c) = w(b,c)$ then
    - Vertex $a$ is between $b$ and $c$
    - Move server at $a$ to $c$, and do not move server at $b$
  - Otherwise...
2-server algorithm (continued)

- Recall two servers are at virtual locations \(\{a, b\}\), and the next request occurs at vertex \(c\).
- Add a new virtual point \(d\) such that:
  - \(w(d, a) = \frac{w(a, b) + w(a, c) - w(b, c)}{2}\)
  - \(w(d, b) = \frac{w(a, b) + w(b, c) - w(a, c)}{2}\)
  - \(w(d, c) = \frac{w(a, c) + w(b, c) - w(a, b)}{2}\)
- For all \(x \notin \{a, b, c\}\), define:
  - \(w(d, x) = \min\{w(d, a) + w(a, x), w(d, b) + w(b, x), w(d, c) + w(c, x)\}\)
- This still obeys the triangle inequality.

k-competitive for \(k\) servers if
graph has exactly \(k+1\) vertices

- There exists a \(k\)-competitive algorithm for the \(k\)-server problem when the graph \(G\) has exactly \(k+1\) vertices.
  - Result obtained by Manasse/McGeoch/Sleator in 1988.
(2k-1)-competitive for k servers

- For all k, there exists a (2k-1)-competitive algorithm for the k-server problem
  - Result obtained by Koutsoupias/Papadimitriou in 1994
  - Idea is called the Work Function Algorithm
    - Remember all previous requests (up to the current request)
    - Dynamic programming can be used to compute the optimal off-line solution for the sequence of all previous requests

Work Function Algorithm (continued)

- Consider an optimum off-line algorithm that starts with servers at locations $S_0 \subseteq V$, where $|S_0| = k$, and then handles the first i requests $R_1, R_2, ..., R_i$
- For each $S \subseteq V$ with $|S| = k$, let the work function $f_i(S) = \text{cost}_{OPT}(R_1, R_2, ..., R_i) + \text{[minimum cost needed to move servers from final locations to S]}$
- Let $S_{i-1}$ denote the set of server locations before serving request $R_i$. The work function algorithm serves request $R_i$ by moving a server from whichever $x \in S_{i-1}$ minimizes this sum:
  
  $f_i(S_i) + w(x, R_i), \text{ where } S_i = S_{i-1} - \{x\} \cup \{R_i\}$

Summary

- k-server problem yields k-competitive on-line algorithms for each of these special cases:
  - When $k = 2$
  - When every $w(x,y) = 1$
  - When $|V| = k+1$
  - When the underlying graph is obtained from a tree

- (2k-1)-competitive on-line algorithm exists for arbitrary graphs
- Conjecture: k-competitive on-line algorithm exists for arbitrary graphs???