Paging recap

1-bit LRU:

- Mark pages when requested.
- On page fault, toss arbitrary unmarked page.
- If all pages in cache are marked, end phase and clear marks.
- Achieves factor of k (easier to see $k+1$).

Randomized version (Marking algorithm):

- Replace “arbitrary” with “random”.
- H_k competitive for $N = k+1$. $2H_k$ competitive otherwise. (Exact value is $2H_k - 1$).

Deterministic lower bound of k. Randomized lower bound of H_k.

Generalization: k-Server Problem

We control k “servers” (pizza trucks) living in some metric space (e.g., 2-d). Repeatedly:

- We get request to some point in the space.
- We need to move some server there.
- We pay the distance moved.

Example: 2 servers on the line at points 0, 100. Requests at 0,1,0,1,....

Generalizes other problems.

- Paging
- Weighted-Caching (some misses more expensive than others – e.g., web browser)

What's known?

- [MMS '88] define problem, k-server conjecture. Solve for $k = 2$ or $N = k + 1$.
- [FRR '90] $O(k!^3)$-competitive alg.
- [Grove '91] $2^{O(k)}$-competitive alg.
- [KP '95] $2k - 1$.

Can solve for optimal *offline* using min-cost max-flow.

For now, let's focus on simplest case of $N = k + 1$.

Can think of following the "hole"...

Slight generalization called the Metrical Task System problem.

The Metrical Task System problem

[Borodin, Linial, Saks]

- Have a device that can be in one of n states.
- Can move from state i to state j at cost d_{ij}, where d is a distance metric.
- We are given, on-line, a sequence of tasks $\sigma = T_1, T_2, T_3, \ldots$. Each task has a cost vector, e.g., $T_1 = (3, 1, 2, 0)$, that gives the cost of processing the task in each of the n states of the device.

→ Given a new task T, we can stay in the current state i, paying $T[i]$, or move to a new state j and perform it there, paying $d_{ij} + T[j]$.

E.g., 4-lane highway. $d_{ij} = |i - j|$.

$(4, 1, 0, 1)$
$(0, 0, 5, 2)$
$(0, 3, 0, 2)$
$(1, 0, 0, 0)$
$(1, 1, 0, 1)$
$(1, 1, 0, 1)$

Examples

- k-server on $k+1$-point space.

  Task vectors look like: $(0, 0, 0, \infty, 0)$.

- Rent or buy. Two states: "don't own" and "own". Distance: (buy cost)/2.

  Need-tux task: (rent cost, 0).

  Outgrow-previous task: $(0, \infty)$

- Paging with direct-access option.

  Task vectors look like: $(0, 0, 0, 1, 0)$.

Hardest case is when task vectors look like

$$(0, \ldots, 0, \epsilon, 0, \ldots, 0)$$

How to compute OPT

Use dynamic programming:

- Let OPT$[i]$ be the optimal offline cost of processing tasks so far and ending at i.
- Update rule:

$$\text{OPT}[i] \leftarrow \min_j(\text{OPT}[j] + T[j] + d_{ij})$$

For example: 4-lane highway. $d_{ij} = |i - j|$.

(4, 1, 0, 1)
(0, 0, 5, 2)
(0, 3, 0, 2)
(1, 0, 0, 0)
(1, 1, 0, 1)
(1, 1, 0, 1)

Notice: (OPT$[i]$ – OPT$[j]$) $\leq d_{ij}$.

Finite diameter $\Rightarrow$ min $\approx$ avg $\approx$ max.

Lower bounds for MTS

Deterministic algorithm can't beat n.

Why?

Work-function algorithm

Idea:

- State i is pinned by j if $\mathrm{OPT}[i] = \mathrm{OPT}[j] + d_{ij}$
- Don't want to be at a pinned state.

Work-function alg: if your state becomes pinned, go to state that pins you. (Move before processing the task)

For example: (see previous slide)

Claim: WF is $2n$-competitive.

- Consider dumb/lazy/continuous version.
- $\sum \mathrm{OPT}[i] \geq$ processing cost.
- move cost $\leq$ processing cost.

So, Cost(alg) $\leq 2n\ \mathrm{avg}_i \mathrm{OPT}[i]$.

polylog(n) randomized alg (idea)

Bartal: can approximate metric space by bunch of well-separated clusters, each of same form recursively. (randomized)

Let's consider 2 clusters at distance d. For each we (recursively) have alg with C.R. $= r$. Ignore amortization for simplicity: think of as "unfair" 2-state MTS.

- What is global C.R. of work-function alg?
- Randomized version: $p_0 = \frac{\text{OPT}[1] - \text{OPT}[0]}{d}$.
- What happens if see $(\delta, 0)$ then $(0, \delta)$?
- Match 'em up. Or use potential function:

$$\Phi = -\frac{1}{2} r d p_0 p_1$$

$$\partial\Phi / \partial\text{OPT}[0] = -\frac{1}{2} r d (1 - 2p_0)(-1/d) = r(\frac{1}{2} - p_0)$$

Comp ratio is $r + 2$.

idea, contd

Prev idea is great for balanced binary tree. What about m-ary?

- Linear probs gives $r + m$.
- Cubic gives $r + 3m^{1/3}$
- Setting exponent to $\log m$ gives $r + O(\log m)$.
- Almost as good: weight-sharing version of "weighted majority" algorithm.

... Need to deal with unbalanced trees

... Need to deal with amortization

Next time: “combining expert advice”

- No cost to move
- but, zero lookahead.

Similar problems: online portfolio updating