Lecture 01/04/15

Friday, April 1, 2016 2:46 PM

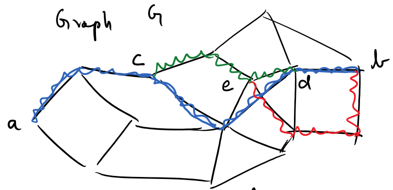
Today :-

→ Online Virtual Circuit Routing

→ Secretary Problem (Online Stochastic)

Problem :-

requests arrive over time
↓
(source, destination, 1 Mbps)



every edge supports ideally 1 Mbps

(eg) ① (a, b, 1)

algo decides which virtual circuit to route request on (path)

② (c, d, 1)

algo goal: find a virtual circuit routing to minimize max load on any edge.

③ (e, b, 1)

Can we find a good path selection algorithm (works online) such that no edge is overused too often.

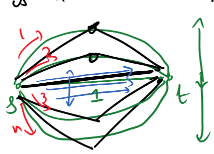
Competitive Ratio

$$\frac{\max_e \text{load}(e) \text{ due to Algo}}{\max_e \text{ load}(e) \text{ of optimal sol}^n}$$

Toy Solution

→ for each request, send 1 unit b/w on shortest path

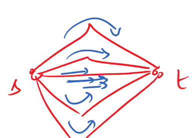
Q) How good is it wrt competitive ratio?



n length 2 paths & 1 length 1 path

σ = request sequence = (s, t, 1), (s, t, 1), (s, t, 1) . . ← n times →

$$CR = \frac{\text{maxload of online Alg}}{\text{max load of OPT}} = \frac{n}{1} = n$$ too high !!



Algo 2 :- Make shortest path weighted

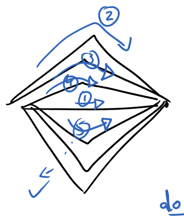
at time t, let load(e, t) denote the # of past ~~~~ paths using edge e.

Then for new request, find shortest path in the weighted graph w(e) = load(e, t) + 1

final load if we are to route along that edge

request 2,

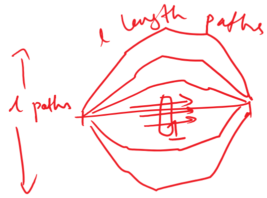
route along that edge



For request 2,
w(direct edge) = 2
w(indirect edge) = 1 each.

doing great!

Worst Case
CR (Algo 2) = $\frac{l}{1} = \sqrt{n}$



1st request : uses direct path
sees cost $\frac{1}{l}$ on direct path
on indirect path
uses direct path.

2nd request : on direct path
sees cost $\frac{2}{l}$ on indirect path

l^{th} request : uses direct path

Need a different balance b/w length & congestion incurred.
shd. be dictated by the obj. function we care about.
max load on any edge

Attempt ③ : $w(e,t) = (\text{load}(e,t)+1)^2$ & find wtd. shortest path.
Similar bad example, $CR = n^{1/3}$.

Good Algo :- use $w(e,t)$ = exponential (load(e,t))
finds shortest path!

a) avoids highly congested edges
b) if all edges are equally congested, prefers short paths w/ few hops.

Algo :-
PROMISE :- Assume that we know max load of optimal solution for the requests which are going to arrive (λ^*)

① At time t, let load(e,t) be # past paths using edge e.

② Set $w(e,t) = (1+\varepsilon)^{\frac{\text{load}(e,t)+1}{\lambda^*}} - (1+\varepsilon)^{\frac{\text{load}(e,t)}{\lambda^*}}$

③ For new request $(s_t, d_t, 1)$, route along shortest path w.r.t weights $w(e,t)$.

④ update loads.

Theorem :-
For above algo, $\boxed{CR \le O(\log m)}$, m is # edges in graph

to show

Proof :- Suffices to show max load on any edge in Alg $\leq O(\log m) \cdot \lambda^*$

(since denominator in CR definition $= \lambda^*$ due to promise)

Potential Function based proof.

$$\phi(t) = \sum_e (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \qquad \phi(0) = m$$

$$\phi(t+1) - \phi(t)$$

$$= \sum_{e \in P_{t+1}} \underbrace{\left[(1+\varepsilon)^{\frac{load(e,t)+1}{\lambda^*}} - (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \right]}$$

P_{t+1}: algo chose

in fact, this was weight that algo set on edge e

$$\leq \sum_{e \in P^*_{t+1}} \left[(1+\varepsilon)^{\frac{load(e,t)+1}{\lambda^*}} - (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \right]$$

P^*_{t+1}: path chosen by hindsight OPT solution

$$= \sum_{e \in P^*_{t+1}} (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \left[(1+\varepsilon)^{\frac{1}{\lambda^*}} - 1 \right]$$

$$\approx \sum_{e \in P^*_{t+1}} (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \left[1 + \frac{\varepsilon}{\lambda^*} - 1 \right]$$

$$\approx \sum_{e \in P^*_{t+1}} (1+\varepsilon)^{\frac{load(e,t)}{\lambda^*}} \cdot \frac{\varepsilon}{\lambda^*}$$

$$\phi(t+1) - \phi(t) \leq \sum_{e \in P^*_{t+1}} (1+\varepsilon)^{\frac{load(e,T)}{\lambda^*}} \cdot \frac{\varepsilon}{\lambda^*}$$

recall : P^*_{t+1} is OPT path to route $(s_{t+1}, d_{t+1}, 1)$ request

Sum over all $t = 0, 1, 2, \ldots, T-1$

$$\phi(T) - \phi(0) \leq \sum_{t=0}^{T-1} \sum_{e \in P^*_{t+1}} (1+\varepsilon)^{\frac{load(e,T)}{\lambda^*}} \cdot \frac{\varepsilon}{\lambda^*}$$

$$\leq \sum_e \cancel{\lambda^*} \cdot \frac{\varepsilon}{\cancel{\lambda^*}} \cdot (1+\varepsilon)^{\frac{load(e,T)}{\lambda^*}}$$

$$\phi(T) - \phi(0) \leq \varepsilon \sum_e (1+\varepsilon)^{\frac{load(e,T)}{\lambda^*}}$$

$$\phi(T) - \phi(0) \leq \varepsilon\, \phi(T)$$

$$(1-\varepsilon)(\phi(T)) \leq \phi(0) = m$$

$$\phi(T) \leq \frac{m}{\ldots}$$

$$\phi(T) \le \frac{\quad}{1-\varepsilon}$$

Set $\varepsilon = 1/2$

$$\sum_e \left(\frac{3}{2}\right)^{\frac{load(e,T)}{\lambda^*}} \le 2m$$

In particular, for every edge,

$$\left(\frac{3}{2}\right)^{\frac{load(e,T)}{\lambda^*}} \le 2m$$

$$\frac{load(e,T)}{\lambda^*} \cdot \log \frac{3}{2} \le \log(2m)$$

$$\Rightarrow \boxed{load(e,T) \le O(\log m) \cdot \lambda^*}$$

Potential ϕ acts like a "soft-max" function

How to avoid knowing λ^*

↓

use algo performance to guess/estimate λ^*

↓

whenever we violate our Thm's performance guarantee, we double our guess for future requests

guesses

1	2	4	8	16	. . . 2^{i^*}

true λ^*

every edge $\le O(\log m)$ + every edge $\le O(2\log m)$ + $O(4\log m)$ + $O(8\log m)$ + $O(16\log m)$ + $O(2^{i^*}\log m)$

total load on algo $\le O(2^{i^*+1}\log m)$
$= O(\log m) \cdot \lambda^*$.

Idea: use Algo + Thm statement to infer OPT value.

Online Stochastic Optimization

(toy example) $\leftarrow$ general application: admission control.

→ There are n people, each has inherent utility $\in [0,1]$

→ they arrive online

→ Algo has to decide who to give resource (instantaneous)

Goal :- max utility,

⇒ max Pr [Algo giving resource to highest utility person]

→ Fully online setting, no meaningful algo can find best utility with non-trivial prob.

→ ε $\sqrt{\varepsilon}$ $\varepsilon^{1/3}$. $\varepsilon^{1/2}$ $\frac{1}{2}$

Observation :- nature is not - worst - case

nature is random somehow

↓

typically a $\underbrace{\text{mix of random + adversarial.}}$

How to come up with model.

OUR MODEL | Adversary chooses the n utility values

They are revealed in a random order (random permutation)

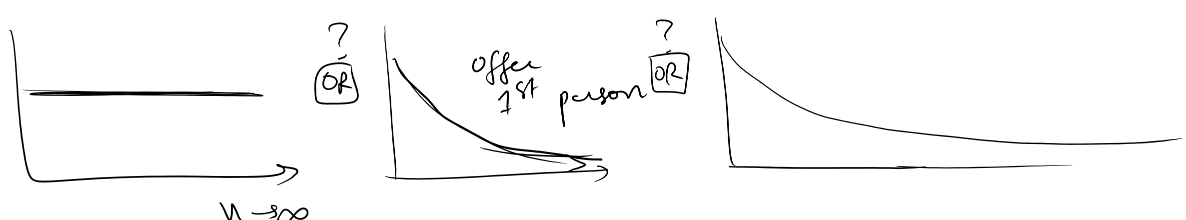
Goal :- Max Pr [we select the highest utility item]

Toy Algo :- offer the k^{th} person the resource.

→ Pr [selecting highest utility] = $\frac{1}{n}$.

Possibilities

P_n = optimal Algo's probability



Intuition

↳ Learn about the values by looking at a small sample.

↳ Algo :- $\boxed{\text{Just Observe}}$ the first K-1 utilities,

in {k, k+1, ... n}

$\hookrightarrow$ <u>Algo</u> :- $\boxed{\text{Just Observe}}$ the first $k-1$ utilities,

Offer to the first person in $\{k, k+1, \dots n\}$ better than the best in $\{1, 2, \dots k-1\}$

$\boxed{\text{What value of } k \text{ to choose}}$?

<u>Toy analysis</u> $(k = \frac{n}{2})$.

$\frac{n}{2}$ | choose 1st person better than best seen so far.

Sureshot success :-

2nd best appears in $\{1, \dots n/2\}$ & best does not. $\rightarrow$ 2nd is in training, 1st is in testing.

$$\Pr[\text{Success}] \geq \frac{1}{4};$$

we should also count 3rd best is in training, 1st & 2nd are both testing, but 1 comes before 2.

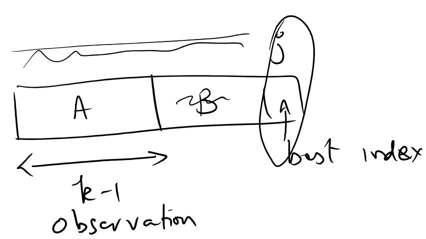
$$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$$

$\boxed{\text{why is } k = \frac{n}{2} \text{ optimal ?}}$

What is Optimal value of k ?

let X_n = random variable for index of best utility person

$$\Pr\left(\text{Success of } \underbrace{\text{Algo}(k)}_{\text{learn } k-1 \text{ best in rest}} \;\middle|\; X_n = j\right) = \begin{cases} 0 & j < k \\ \frac{k-1}{j-1} & j \geq k \end{cases}$$



Algo is successful if local 2nd best within the 1st j people appears in training phase

Overall success

$$\Pr[\text{Success of Alg}(k)] = \sum_{j=1}^{k-1} \Pr[X_n = j] \cdot 0 + \sum_{j=k}^{n} \Pr[X_n = j] \frac{k-1}{j-1}$$

$$= \frac{k-1}{n} \sum_{j=k}^{n} \frac{1}{j-1}$$

$$\simeq \frac{k-1}{n} \ln \frac{n}{k} \qquad \text{(cheat, but ok, not bad)}$$

Optimize over k :-

$$f(x) \approx x \ln \frac{n}{x}$$

$$f'(x) = 0$$

$$\Rightarrow \ln \frac{n}{x} + \frac{x \cdot x}{n} \cdot \left(-\frac{n}{x^2}\right) = 0$$

$$\Rightarrow \ln \frac{n}{x} = 1$$

$$\Rightarrow \frac{n}{x} = e$$

$$\Rightarrow x = \frac{n}{e} \qquad \left[k = \frac{n}{e}\right]$$

$$\boxed{\Pr[\text{Algo}(k) \text{ succeeds}] = \frac{1}{e}} >> \frac{1}{4}$$

-> Algo learns/estimates input parameters on the fly.

-> Rich history

-> Introduces semi online, semi stochastic model

-> No algo can beat 1/e. Best algorithm!

One of the proofs of optimality is via LP + duality
