Today:
- Online Virtual Circuit Routing
- Secretary Problem (Online Stochastic)

**Problem:**
requests arrive over time
\( (\text{source, destination, req rate}) \)

1. \( (a, 1, 1) \)
   - algo decides which
   virtual circuit to route
   request on

2. \( (c, 1, 1) \)
   - algo goal: find a virtual circuit routing to
   minimize worst load on any edge.

3. \( (e, 1, 1) \)
   - Can we find a good path allocation algorithm
     (works online) such that no edge is
     overloaded too often.

**Competitive Ratio**

\[
\text{CR} = \frac{\text{max load of online algo}}{\text{max load of optimal}}
\]

**Toy Solution**
- for each request, send 1 unit flow on shortest path

1. How good is it wrt competitive ratio?

\[\sigma = (3, 1, 1, 4, 1, 1, 4, 1, 1)\]

\[n = \text{length of } \sigma\]

\[\text{CR} = \frac{\text{max load of online}}{\text{max load of } \sigma}\]

**Algo 2:** Make shortest path weighted
- at time \( t \), let \( \text{load}(e, t) = \text{deg}\)
  \( e \)
- out paths using edge \( e \)
- Then for new request, find shortest path in
  the weighted graph \( \text{w}(e) = \text{load}(e, t) + 1 \)

Find load if we are to
  rock along that edge
Need a different balance b/w length & congestion incurred. And be dictated by the Qij function we care about. May load on any edge.

Attempt 3: \( w(r,i) = (\text{load}(r,i)+i)^2 \) & find wtd. shortest path.

Similar load example, \( CR = \sqrt{3} \).

Good Algo: Use \( w(r,i) = \text{exponential}(\text{load}(r,i)) \) & finds shortest path !

- a) avoids highly congested edges
- b) if all edges are equally congested, progress slower paths w/ few steps.

Algo:

**PROMISE:** Assume that we know max load of optimal solution for the requests which are going to arrive \((X^*)\).

1. At time \( t \), let \( \text{load}(r,i) \) be the past paths using edge \( e \).
2. Set \( w(r,i) = \left(1+\varepsilon\right)^{\frac{\text{load}(r,i)}{X^*}} \).\( (1+\varepsilon)^{\frac{\text{load}(r,i)}{X^*}} \)
3. For new request \((b,r,d,i)\), route along shortest path w/ weights \( w(r,i) \).
4. Update loads.

**Theorem:** For above algo, \( CR \leq O(\log m) \), \( m \) is # edges in graph.

0. 1. 0. 2. ... to follow
Proof: Suffices to show max load on any edge in $G_T \leq O(\log n) \cdot \lambda^x$.

(since denominator in CR definition = $\lambda^x$ due to $\gamma_r$-approx.)

Potential Function based proof:

$$\phi(t) = \sum_{e \in E} (1 + \epsilon) \frac{\text{load}(e, t)}{\lambda^x}$$

$$\phi(t+1) - \phi(t) = \sum_{e \in E(t+1)} (1 + \epsilon) \frac{\text{load}(e, t+1)}{\lambda^x} - (1 + \epsilon) \frac{\text{load}(e, t)}{\lambda^x}$$

$$\leq \sum_{e \in E_{t+1}} (1 + \epsilon) \left[ \frac{\text{load}(e, t+1)}{\lambda^x} - \frac{\text{load}(e, t)}{\lambda^x} \right]$$

In fact, this sum weight that edge set on edge $e$.

$$\leq \sum_{e \in E_{t+1}} (1 + \epsilon) \frac{\text{load}(e, t+1)}{\lambda^x} \\ \text{path chosen by hands of OPT solution}$$

$$= \sum_{e \in E_{t+1}} (1 + \epsilon) \left[ \frac{1}{\lambda^x} \cdot \frac{\text{load}(e, t+1)}{\lambda^x} - 1 \right]$$

$$\approx \sum_{e \in E_{t+1}} (1 + \epsilon) \frac{\text{load}(e, t+1)}{\lambda^x} \cdot \frac{\epsilon}{\lambda^x}$$

$$\phi(t+1) - \phi(t) \leq \sum_{e \in E_{t+1}} (1 + \epsilon) \frac{\text{load}(e, t+1)}{\lambda^x} \cdot \frac{\epsilon}{\lambda^x}$$

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

$$\phi(T) - \phi(0) \leq \sum_{t=0}^{T-1} \sum_{e \in E_{t+1}} (1 + \epsilon) \frac{\text{load}(e, t+1)}{\lambda^x} \cdot \frac{\epsilon}{\lambda^x}$$

$$\leq \sum_{e \in E} \frac{\epsilon}{\lambda^x} \cdot (1 + \epsilon) \frac{\text{load}(e, T)}{\lambda^x}$$

$$\phi(T) - \phi(0) \leq \epsilon \sum_{e \in E} (1 + \epsilon) \frac{\text{load}(e, T)}{\lambda^x}$$

$$\phi(T) - \phi(0) \leq \epsilon \phi(T)$$

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

$$\phi(t+1) \leq \frac{m}{\epsilon}.$$

Quick Note Page 5
Set $\epsilon = \frac{1}{2}$

$$\sum_{e} \left( \frac{3}{2} \right) \frac{\text{load}(e, T)}{x^e} \leq 2m$$

In particular, for every edge,

$$\left( \frac{3}{2} \right) \frac{\text{load}(e, T)}{x^e} \leq 2m$$

$$\Rightarrow \frac{\text{load}(e, T)}{x^e} \log \frac{3}{2} \leq \log(2m)$$

$$\Rightarrow \text{load}(e, T) \leq O(\log m) \cdot x^e$$

Potential $\phi$ acts like a "soft-max" function

---

How to avoid knowing $x^e$:

- Use algo performance to guess/estimate $x^e$.
- Whenever we violate our Thm's performance guarantee, we double our guess for future requests.

- Guesses:
  - 1
  - 2
  - 4
  - 8
  - 16
  - $2^e$

- Every edge:
  - $\leq O(\log m) + O(8 \log m) + O(16 \log m) + O(2^e \log m)$

- Total load on algo $\leq O(2^{e+1} \log m) = O(\log m) \cdot x^e$.

- Idea: use AlgThm statement to infer OPT value.

---

Online Stochastic Optimization

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

- There are $n$ people, each has inherent utility $\epsilon \in [0, 1]$
- They arrive online
Algo has to decide who to give resource

$\text{Goal:} \max \text{ utility,}$

$\Rightarrow \max \text{ Pr [ Algo giving resource to highest utility person]}$

$\rightarrow \text{Fully online setting, no meaningful algo can find best utility with non-trivial prob.}$

Observation: nature is worst-case

$\Delta$

nature is random somehow

$\downarrow$

typically a 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: $\max \text{ Pr [ we select the highest utility item]}$

Toy Algo: offer the $k$th person the resource.

$\rightarrow \text{Pr [ selecting highest utility]} = \frac{1}{n}$

Possibilities

$P_n = \text{optimal Algo's probability}$

Intuition:

Learn about the values by looking at a small sample.

$\Rightarrow \text{Algo: Just observe the first } k-1 \text{ utilities,}$

... in $\{k, k+1, \ldots, n\}$
What value of k to choose?

Toy analysis ($k = \frac{n}{2}$).

Choose 1st person better than best seen so far.

Success rate:

- 2nd best appears in $\{1, \ldots, \frac{n}{2}\}$.
- 1st is in training, 2nd is in testing.

$\frac{1}{2}$ chance the 2nd best is in training, 1st is in testing.

$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$ chance 3rd best is in training, 1st & 2nd are both testing, but 1 comes before 2.

Why is $k = \frac{n}{2}$ optimal?

What is optimal value of k?

Let $X_n =$ random variable for index of least utility person.

$P\left( \text{Success of Algo}(k) \mid X_n = j \right) = \begin{cases} 0 & j < k \\ \frac{k-1}{j-1} & j \geq k \end{cases}$

Algo is successful if 2nd best within the first $j$ people appears in training phase.
Overall Success
\[ \Pr \left( \text{Success of Alg}(k) \right) = \sum_{j=1}^{k-1} \Pr \left( X_{n-j} = 0 \right) + \sum_{j=k}^{n} \Pr \left( X_{n-j} = k-1 \right) \]
\[ = \frac{k-1}{n} \sum_{j=k}^{n} \frac{1}{j-1} \]
\[ \approx \frac{k-1}{n} \ln \frac{n}{k} \quad (\text{cheat, but okay, not bad}) \]

Optimize over \( k \):
\[ f(k) = \alpha \ln \frac{n}{k} \]
\[ f'(k) = 0 \]
\[ \Rightarrow \ln \frac{n}{k} + \frac{\alpha}{n} \left( -\frac{n}{k^2} \right) = 0 \]
\[ \Rightarrow \ln \frac{n}{k} = \frac{\alpha}{n} \]
\[ \Rightarrow \frac{n}{k} = e \]
\[ \Rightarrow k = \frac{n}{e} \]

\[ \Pr \left[ \text{Alg}(k) \text{ succeeds} \right] = \frac{1}{e} \gg \frac{1}{e^2} \]

\( \Rightarrow \) Alg learns/estimates input parameters on the fly.
\( \Rightarrow \) Rich history
\( \Rightarrow \) Introduces semi online, semi stochastic model
\( \Rightarrow \) No algo can beat \( \frac{1}{e} \). Best algorithm!

One of the proofs of optimality is via LP + duality.