Lecture 30/03

Wednesday, March 30, 2016 1:46 PM

Today:

→ Online Set Cover
  ↳ Algo + Proof

→ Online Routing
  ↳ Algo + Proof

U = universe of possible elements

$\mathcal{S}$ = collection of sets = m

$X \subseteq U$ is online set of elements that need to be covered,

$S_A \in \mathcal{S}$ is online set cover for X.

$|X| = n$

$e_1 \quad e_2 \quad \ldots \quad e_n$

↳ Alg has to cover (done w/o knowledge of future).

Goal: minimize competitive ratio

$$\boxed{CR(Alg) = \max_{\substack{\text{All possible}\\ \text{inputs } X}} \frac{\#\text{ sets alg picks for input } X}{\text{Optimal }\#\text{ sets needed for } X}}.$$

Last class

↳ CG algo was proposed, but also shown why bad.

↳ Fractional algo (MW-inspired, increases confidence in sets based on past utility)

ONLINE FRACTIONAL ALGORITHM

Maintain fractional assignments for all sets

→ ONLINE ROUNDING SCHEME

Make sure that S_i is ... probability

Maintain fractional soln.
for all sets

$x_{S_1} = 0.7$, $x_{S_2} = 0.2$, $x_{S_3} = 0.3$, etc.

updated monotonically over time

Make sure that S_1 is included with probability $\propto x_{S_1}$

eg:- e_1 arrived

$x_{S_1} = 0.5$, $x_{S_2} = 0.3$

$x_{S_3} = 0.2$

Include S_1 wp $\propto C \cdot x_{S_1}$

S_2 wp $= C \cdot x_{S_2}$

S_3 wp $= C\, x_{S_3}$

e_2 arrived

$x_{S_1} \to 0.8$

$x_{S_2} \to 0.9$

$x_{S_3} = 0.2$

Include S_1 wp $\propto C \Delta x_{S_1}$

S_2 wp $\propto C \Delta x_2$

Maintaining

Invariant that at any time, every set is chosen in soln w.p $= \underbrace{C \cdot x_S}$

Thm ①

There exists online algo. for maintaining fractional solutions st cost of fractional soln $\le O(\log m) \cdot OPT$

$$\sum x_S^{(t)} \le O(\log m)\ OPT^{(t)}$$

Thm ②

There is a rounding scheme to select sets (based on fractional soln) st @ time t, $E\left[\sum Z_S^{(t)}\right] \le O(\log n) \sum x_S^{(t)}$

Indicator variable for whether S has been rounded or not.

Thm ① + ②

$\Rightarrow$ $E[\text{Cost of online Algo}] \le O(\log m \log n)\ OPT^{(t)}$

sets

elts that show up $= |X|$

Thm ① : fractional algorithm

Initialize all sets to $x_S^{(0)} = \frac{1}{m}$ $\qquad \sum x_S^{(0)} = 1$

→ when element e_t arrives

While $\sum_{S: e_t \in S} x_S < 1$

$x_S \leftarrow 2 x_S \quad \forall S : e_t \in S$ (More useful sets increase value rapidly)

Resulting $\{x_S\}$ values form fractional solution @ time t, called x_t.

Repeatedly useful sets

$\frac{1}{m} \to \frac{2}{m} \to \frac{4}{m} \to \cdots$ 1

$\leq O(\log m)$ times

Each element participates in a doubling ~~step~~ step $\leq O(\log m)$ times

Thm ①

At any time t, $\sum x_S^t$ = fractional cost of online algo at time t

$\leq O(\log m) \, OPT^t$

↑ optimal set cover for elements arrived upto time t.

Proof :-

Min $\sum x_S$

$\sum_{S: e \in S} x_S \geq 1 \quad \forall e_1, e_2, \ldots, e_t$

Max $\sum y_e$

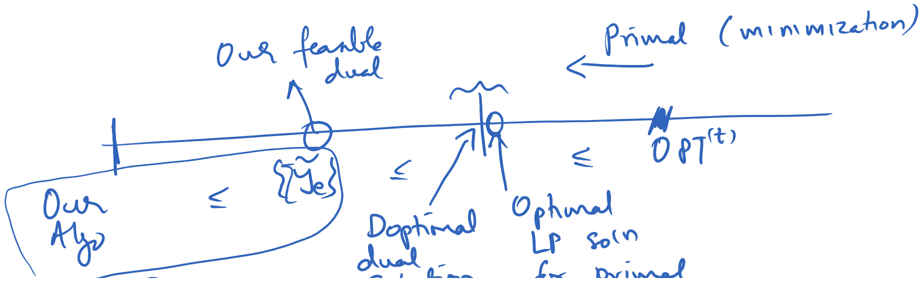
$\sum_{e \in S} y_e \leq 1 \quad \forall S \in \mathcal{S}$

Lemma ①

Will construct a feasible dual solution $\{\tilde{y}_e^t\}$ st

Online Frac Cost $= \sum x_S^t \leq O(\log m) \sum_e \tilde{y}_e^t \leq O(\log m) \, OPT^{(t)}$

↑ weak duality



Our Algo $\leq$ 4 (yes) Optimal dual solution 'Optimal LP soln for primal

Lemma 1

Q) How do we construct the dual?

two fold goal

(a) It should be feasible

(b) It should help me bound cost of online algorithm

Initialize all sets to $x_S^{(0)} = \frac{1}{m}$ $\qquad \sum x_S^{(0)} = 1$

→ When element e_t arrives

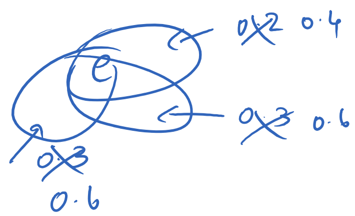
While $\sum_{S: e_t \in S} x_S < 1$

$x_S \leftarrow 2 x_S \quad \forall S : e_t \in S$ (More useful sets increase value rapidly)

(

Online algo increases cost whenever it executes the while loop - (in one iteration of while loop)

Q) How much does cost increase be? (in one iteration of while loop)



$\sum_{S \text{ covering } e} x_S = 0.8$

↓ while loop is entered

Overall new soln. has

$\sum_{S \text{ covering } e} x_S = 1.6$

In this example ← no other x_S was increased →

Overall fractional increase = 0.8.

<u>Cl</u>: In any iteration, fractional cost increases by $\leq$ 1

↳ (lets increase $y_e \leftarrow y_e + 1$ in this iteration)

For every while loop iteration, see which element caused that iteration,

increase $y_e \leftarrow y_e + 1$.

let $\{\tilde{y}_e^t\}$ denote final dual variables.

$$\boxed{\text{Final fractional cost} = \sum_{\text{all } S} x_S^t \leq \sum_e \tilde{y}_e^t}$$

← by design

Worry: Dual solution may not be feasible :(

↳ Some y_e^s can be > 1

$$\boxed{\text{Not true that } \sum_{e \in S} \tilde{y}_e^t \leq 1 \quad \forall S}$$

Saving grace

Constraint wont be violated by too much !!

Claim :-

$$\underbrace{\sum_{e \in S} \tilde{y}_e^t}_{\text{LHS}} \leq (\log m + 1) \quad \forall S$$

Proof :-

Fix a set S.

look at all the iterations which increased the LHS

↳ Nice property :- in all such iterations, S must be covering the current element which caused the iteration.

in each such iteration, x_S value doubled.

↓

Can't cross 2 !

$$\underbrace{\frac{1}{m} \rightarrow \frac{2}{m} \rightarrow \frac{4}{m} \rightarrow \cdots}_{} \; \frac{m}{m} \rightarrow \frac{2m}{m}$$

iterations $= (\log m + 1)$

Our fractional cost @ time $t = \sum_{\text{all sets}} x_S^t \leq \sum_e \tilde{y}_e^t$ ⓐ

& $\left\{ \dfrac{\tilde{y}_e^{(t)}}{\log m + 1} \right\}$ is feasible for dual program. ⓑ

ⓐ & ⓑ ⇒ lemma ① ⇒ theorem ①

Summary:

- → MW inspired algo (more aggressive for more useful sets historically)
- → Analyzed Algo cost vs dual solution we constructed (upto $O(\log m)$)
- → duality ⇒ dual opt ≤ Real Set Cover Optimal.

In some iteration

	S_1	S_2	S_3	S_4	S_5	S_6
$\{x_S\}$ values	0.3	0.1	0.05	0.2	0.7	0.6

e arrives

3 sets cover e

0.6   0.1   0.4

→ Includes S_1 w.p $C \times 0.3$
Includes S_3 wp $C \times 0.05$
Includes S_4 wp $C \times 0.2$

Analysis of online rounding:-

@ time t, Pr[Set S is included by now] $= C x_S^t$

Look at any element arrived already,

E[# sets in my rounded sol^n which cover e] $\geq C \sum_{S \ni e} x_S^t \geq C$

Moreover,

E[Cost of rounding solution] $= C \sum_{\text{all sets } S} x_S^t$.

Pr[a fixed elt which has arrived is not covered by rounding solution] $\leq e^{-C}$

$= \frac{1}{n^2}$.

Set $C = 2\log n$.

Thm (2) Above rounding algo. maintains online set cover, st
@ time t, it covers all elements wp $1 - 1/n$
& has expected cost $\leq O(\log n) \sum x_S^t$

LOWER BOUND EXAMPLE :-

No online algorithm can do better than some α factor.

No online algorithm can do better than some α factor.

set $L = \ell^2$

$U = \{ e_1, e_2, \ldots, e_L \}$

$\mathcal{S} = \{$ all possible ℓ-sized subsets of $U \}$; $|\mathcal{S}| = \binom{L}{\ell}$

Bad input sequence :-

present e_1, algo chooses say $S_1 \ni e_1$

present e_2 not in S_1, algo has to choose new set, say S_2

present e_3 not in S_1 or S_2, algo forced to choose S_3

$\vdots$

present e_ℓ not in $\underbrace{S_1 \text{ or } S_2 \text{ or } \ldots \text{ or } S_{\ell-1}}$, algo forced to pick S_ℓ

Algo cost $= \ell$

Opt cost for $\{e_1, e_2 \ldots e_\ell\} = 1$

gap $= \ell = \Omega(n)$

$\Rightarrow$ Competitive ratio $= \ell$. ; Ex: how does this behave with m & n ?

[Huge gap b/w offline set cover & online set cover]

$\boxed{\text{In offline set cover, Randomized rounding } \leq O(\log n).}$

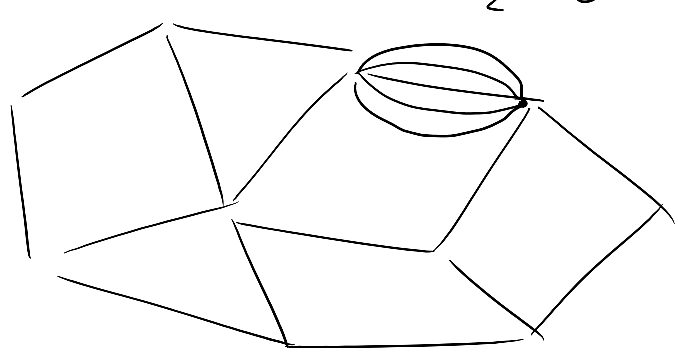
Online Routing

→ large graph G

→ Unit capacities on edges

→ Online requests arrives

Source vertex, destination vertex unit bandwidth requirement.



→ Goal of Algo :- a) Fix a path to route this bandwidth → can't change later ← to ensure QoS.

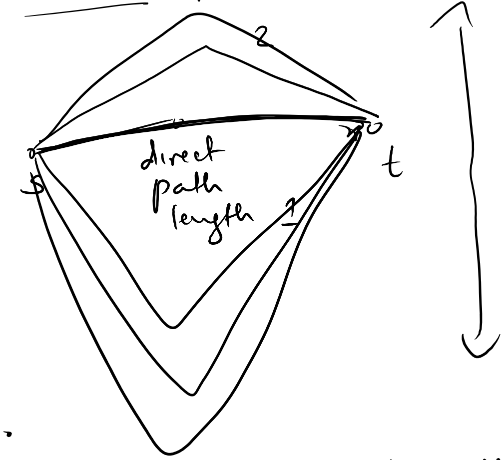
a) → can't change later – QOS.

b) Minimize the $\underset{\text{max}}{\downarrow}$ congestion on edges ← to ensure QOS.

Candidate 1 :- choose shortest path. ← dumb candidate algo !!

1st request (s, t, 1)
2nd request (s, t, 1)

nth request (s, t, 1)



n indirect paths of length 2.

Congestion of SP = n.
Optimal congestion = 1. (using indirect paths of length 2).

Candidate 2 :- give weights to the edges? → (current load ??)

Candidate 3 :- Greedily pick a path which minimizes max load after that path is chosen.

(eg : throw out high-congestion edges & find shortest path)

exponential weights are the right weights !!

$$w(e) = \left(\frac{3}{2}\right)^{\text{curload}(e)}$$

: Find shortest path !!

↑

Soft form of max congestion (soft max)

↓

Nice middleground b/w max congestion & length of path