Today:

- Online Set Cover
  - Algorithm + Proof

- Online Routing
  - Algorithm + Proof

\[ U = \text{universe of possible elements} \]
\[ S = \text{collection of sets} = m \]
\[ X \subseteq U = \text{online set of elements that need to be covered} \]
\[ S_1 \subseteq S = \text{online set cover for } X \]
\[ |X| = n \]

- Algorithm has to cover (done w/o knowledge of future).

- Goal: minimize competitive ratio

\[ CR(A_\text{alg}) = \frac{\text{Max possible input } X}{\text{Optimal set needed for } X} \]

Last class:

- CH algo was proposed, but also shown why bad.
- Fractional algo (MW-inspired, increases confidence in set based on past utility)

**ONLINE** FRACTIONAL ALGORITHM **ONLINE Rounding Scheme**

Maintain fractional assignments for all sets.

\[ \text{Make sure that } S \text{ is} \]

Quick Notes Page 1
Maintain for all sets $x_1 = 0.7, x_2 = 0.2, x_3 = 0.3$.

- Updated monotonically over time.

- $e_1$ arrived
  - $x_1 = 0.5, x_2 = 0.3, x_3 = 0.2$

- $e_2$ arrived
  - $x_3 \rightarrow 0.8$
  - $x_2 \rightarrow 0.9$
  - $x_3 = 0.2$

Make sure that $S_1$ is included with probability $\alpha x_1$.

Include $S_1$ w.p. $\alpha C x_1$

Include $S_2$ w.p. $C x_2$

Include $S_3$ w.p. $C x_3$

Maintaining invariant that at any time, every set is chosen w.r.t.

$\sum z_s \leq O(\log n) \cdot OPT$

Then (1)

There exists an online algo. for maintaining fractional solution $S_{opt}^n$ such that $\sum z_s^{(t)} \leq O(\log n) \cdot OPT^t$.

Then (2)

There is a rounding scheme to select sets (based on fractional solution) $S_t$ at time $t$.

$\sum_{S_t} E[\sum z_s^{(t)}] \leq O(\log n) \sum z_s^{(t)}$

Indicator variable for whether $S$ has been rounded or not.

Then (1) + (2)

$\Rightarrow \ E[\text{cost of online algo}] \leq O(\log n \cdot \log n) \cdot OPT^t$

# sets # sets that show wp = $|X|$
Thus \( D \): fractional algorithm

Initialize all sets to \( x_s^{(0)} = \frac{1}{m} \)

\( \sum x_s^{(0)} = 1 \)

\( \rightarrow \) When element \( e_t \) arrives

\[ \text{While } \sum_{S \in \mathcal{E}} x_S < 1 \]

\[ x_S' = 2 \cdot x_S + S : \forall S \in \mathcal{E} \] (More useful sets increase value rapidly)

Resulting \( \{x_s\} \) values from fractional solution at time \( t \), called \( x_t \).

\[ \text{Repeatedly useful sets} \]

\[ \frac{1}{m} \rightarrow \frac{1}{2m} \rightarrow \frac{1}{4m} \rightarrow \ldots \rightarrow \frac{1}{2^t m} \leq O(\log m) \text{ times} \]

\[ \{ \text{Each element participates in a doubling step} \} \leq O(\log m) \text{ times} \]

Thus \( D \)

At any time \( t \), \( \sum x_S^t = \text{fractional cost of online algo at time } t \)

\( \leq O(\log m) \) OPT

Proof:

\[ \min \sum x_S \]

\[ \sum x_S > 1 \quad \forall e_1, e_2, \ldots, e_t \]

\[ \max \sum y_S \]

\[ \sum y_S \leq 1 \quad \forall S \in \mathcal{E} \]

Lemma 6

Will construct a feasible dual solution \( \sum y_S \) \( \leq \)

Optimal LP cost \( = \sum x_S^t \leq O(\log m) \sum y_S \leq O(\log m) \) OPT

Weak duality

Our feasible dual \( \leq \) Optimal dual

Minimized for primal
Q1) How do we construct the dual?

Two goals:
1. It should be feasible
2. It should help us bound cost of online algorithm

Initialize all sets to \( x^{(0)}_s = \frac{1}{m} \)
\[ \sum x^{(0)}_s = 1 \]

\[ \rightarrow \text{when element } e_k \text{ arrives} \]
\[ \rightarrow \text{while } \sum_{s \in S} x_s < 1 \]
\[ x_s \leftarrow 2x_s + S' \in S \quad \text{(more useful sets increase value rapidly)} \]

Online algo increases cost whenever it executes the while loop.

Q2) How much does cost increase be? (in one iteration of while loop)

\[ \begin{array}{c}
\text{Current } e \in S' \quad \text{Scovry} = 0.6 \\
\sum x_s = 0.8
\end{array} \]

\[ \text{Overall new soln. has} \quad \sum x_s = 1.6 \]

In this example, no other \( x_s \) was increased.

\[ \text{Overall fractional increase } = \frac{0.8}{1.6} = 0.5 \]

Claim: In any iteration, fractional cost increases by \( \leq \frac{1}{2} \)

\[ \rightarrow \text{(let's increase } y_e \leftarrow y_e + 1 \text{ in this iteration)} \]

For every while loop iteration,
see which element caused that iteration,
increase \( y_e \leftarrow y_e + 1 \).
let \( \{ \tilde{y}_e \} \) denote final dual variables.

\[
\text{Final fractional cost} = \sum_{s \in S} x_s^t \leq \sum_{e} \tilde{y}_e^t \tag{a}
\]

\( \text{by design} \)

Worry Dual solution may not be feasible :=

\( \Rightarrow \) Some \( \tilde{y}_e^t \) can be \( \geq 1 \)

\[
\text{Not true } \sum_{e \in S} \tilde{y}_e^t \leq 1 + S
\]

Saving grace constraint won't be violated by too much !!

Claim :=

\[
\sum_{e \in S} \tilde{y}_e^t \leq (\log m + 1) + S
\]

Proof :=

Fix a set \( S \).

Look at all the iterations which increase the ULS.

\( \Rightarrow \) Nice property := in all such iterations, \( S \) must be covering the current element which caused the iteration.

\[
\frac{1}{m} \rightarrow \frac{2}{m} \rightarrow \frac{4}{m} \rightarrow \ldots \rightarrow \frac{m}{m} \rightarrow \frac{2m}{m}
\]

\[
\# \text{ iterations} = (\log m + 1)
\]

Our fractional cost at time \( t \) = \( \sum_{s} x_s^t \leq \sum_{e} \tilde{y}_e^t \)

\[
\Rightarrow \{ \frac{y_e^{(t)}}{\log m + 1} \} \text{ is feasible for dual program.}
\]

\( \triangleright \) & \( \triangleright \) \Rightarrow \text{lemma} \( \triangleright \) \Rightarrow \text{theorem} \( \triangleright \)
Summary:

- NW inspired algo (more aggressive for more useful sets historically)
- Analyzed algo cost vs. dual solution we constructed (upto $O(\log m)$)
- duality $\Rightarrow$ dual opt $\leq$ real set cover optimal.

$\exists$ value $s_{i,1}$ $s_{i,2}$ $s_{i,0}$ $s_{i,-1}$

3 sets cover $e$

$0.6 \ 0.1 \ 0.4$

Includes $s_1$ wp $C \times 0.3$

Includes $s_3$ wp $C \times 0.05$

Includes $s_4$ wp $C \times 0.2$

Analysis of online rounding:

At time $t$, $P_r[\text{set } s \text{ is included}] = C x_s^t$ by now

Look at any element arrived already,

$E[\# \text{ sets in my rounded sol which cover } e] \geq C \sum s^t \geq C$ for $s \in e$.

Moreover,

$E[\text{cost of rounding solution}] = C \sum s^t \cdot$ all $s < S$.

$P_r[\text{ a fixed elt which has arrived is not covered}] \leq C$ by rounding solution

set $C = 2\log n$.

Then $\exists$ Above rounding algo maintains online set cover, at

@ time $t$, it covers all elements wp $1 - \frac{1}{n}$

& has expected cost $\leq O(\log n) \cdot \sum s^t$

Lower Bound Example:

No online algorithm can do better than some $\alpha$ factor.

$\alpha \leq 1^2$.
No online algorithm can do better than some $\alpha$ factor

$$U = \{ e_1, e_2, \ldots, e_L \} \quad \text{at } L = L^*$$

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

**Bad input sequence:**
- Present $e_1$, algo chooses say $S_1 \supset e_1$
- 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 $S_3$
- Present $e_4$ not in $S_1, S_2, \ldots, S_{k-1}$, algo forced to pick $S_k$

**Algo cost:** $L$

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

$\implies$ **Competitive ratio $= L$.**

**Ex:** how does this behave with $m \leq n$?

Huge gap blow offline set case & online set case

In offline set case, Randomized rounding $\leq O(\log n)$.

---

**Online Routing**

- Large graph $G$
- Unit capacities on edges

- Online requests arrive

Source vertex, destination vertex
Unit bandwidth requirement

- **Goal of Algo:** Fix a path to route this bandwidth
  $\implies$ Can't change later $\implies$ to ensure QoS.
Candidate 1: choose shortest path. \( \leftarrow \text{dumb candidate algo} \)

1st request \((s, t, 1)\)
2nd request \((s, t, 1)\)
3rd request \((s, t, 1)\)

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

Candidate 2: give weights to the edges? \( \checkmark \) (Current load??)

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

\(\text{Exponential weights are the right weights}!!\)

\[ w(e) = \text{curload}(e) \left(\frac{\lambda^2}{2}\right) \]

Soft form of max congestion (soft max)

Nice middle ground b/w max congestion \& length of path,

\( \text{can't change later} \rightarrow \text{QoS.} \)