Lecture #1:

Introduction to Approximation Algorithms

Basic situation

- Many discrete optimization problems are NP-hard.
  - ⇒ unless P=NP, cannot solve them exactly.
- Also don't believe that P=NP (most people).
- So, get solutions that are near optimal (in polynomial time)

Example:

- traveling salesperson problem (TSP) is NP-hard.
  - But can output solution that has cost ≤ 1.5 OPT for any metric space
    $\leq (1+\varepsilon)$ OPT in time $n^{O(1/\varepsilon^2)}$ in 2-dim Euclidean space

- Max Coverage problem is NP hard
  - But can approximate to within a factor of $\frac{e}{e-1}$.

- Vertex Cover is NP hard
  - But can approximate to within factor of 2.

- Set Cover is also NP hard
  - But approximate to within factor of 2.

In today's lecture

- See some optimization problems.

  Set cover, Vertex cover, Max Coverage.

- See some concepts
  - Linear programming duality
  - Approximation ratios, Integrality gaps.
- See some techniques
  - Greedy algorithms
  - Relax-and-round
  - Dual Fitting

etc.

Let's start:

Min ~~Vertex~~ Vertex Cover.

~~Given~~ Input: graph $G = (V, E)$.

may have weights on vertices. (non negative)

Output: Set $C \subseteq V$ such that

every edge ~~has~~ has at least one endpoint in C.

↑ it is "covered".

Want: minimize size of C (or weight of C).

Set Cover:

Universe U. $|U| = n$

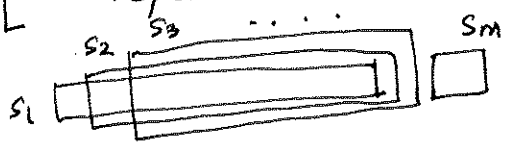
- Sets $S_1, S_2 \ldots S_m$ subsets of U. say $\mathcal{F} = \{S_1, S_2 \ldots S_m\}$
- Say union of S_i's $= U$. (they "cover" U)

Want: smallest sub collection of sets that cover U.

——— X ———

NP hard

Algo 0: [pick largest set.
repeat.



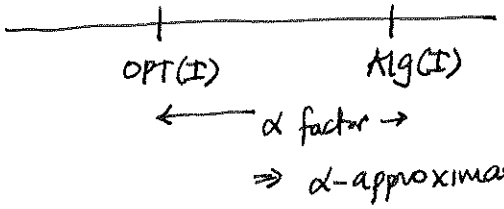
will pick large sets regardless of whether covering uncovered elements

Algo 1: [pick set in $\mathcal{F}$ that has largest # of uncovered elements
repeat.

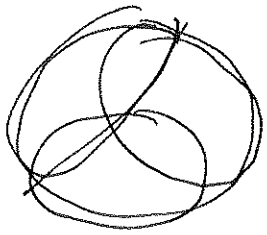
Claim: Greedy algo is an $(\ln n)$-approximation

↑ on each instance I,

Algo(I) has size $\leq (\ln n) \cdot OPT(I)$.



Proof: Suppose OPT has k sets.



Let U_t = uncovered elements after algo picks t sets.

- $U_0 = U$. $\quad |U_0| = n$
- $|U_t| \leq |U_{t-1}|\left(1 - 1/k\right)$.

  Why: OPT covers all elements using k sets
  $\Rightarrow$ it covers U_{t-1} using k sets
  $\Rightarrow$ $\exists$ set that covers $\geq \frac{|U_{t-1}|}{k}$ elements.

$\Rightarrow |U_T| \leq |U_0| \cdot (1 - 1/k)^T$

but $1+x \leq e^x \ \forall x$
and $1+x < e^x \ \forall x \neq 0$

$< n \cdot (e^{-1/k})^T$

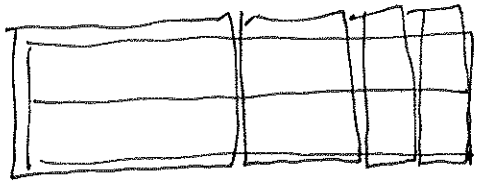
$= n e^{-T/k}$.

So if $T = k \ln n \Rightarrow n e^{-T/k} = 1 \Rightarrow |U_T| < 1 \Rightarrow |U_T| = 0$.

$\Rightarrow$ Algo stops after at most $k \ln n$ sets picked! ▨

Is this "tight"?

- Can we show an example where algo actually is this bad, or can we improve the analysis?



$K = 2$. (= OPT)

Greedy Algo picks $\log_2 n$ sets

$\Rightarrow$ gap $\geq \frac{\log_2 n}{2}$.

$\Rightarrow$ Greedy algo always outputs solution with

$\leq (\ln n) \cdot OPT$ sets $= O(\log n)\, OPT$

And there are instances where it outputs

$\geq \frac{\log_2 n}{2} \cdot OPT$ sets $= \Omega(\log n)\, OPT.$

$\Rightarrow$ Greedy algo is an $\Theta(\log n)$-approximation.

——— X ———

BTW: suppose sets have cost. S_i has cost $c(S_i) \geq 0$.

Want least cost set cover.

Algo Greedy: pick set $S \in \mathcal{F}$ such that maximizes $\frac{\#\text{uncovered elements in } S}{c(S)}$.

↑ "bang per buck"

Thm: always outputs solution of cost $\leq (\ln n) \cdot OPT$.

Proof: suppose U_{t-1} uncovered after $t-1$ sets picked.

$\Rightarrow$ can cover $\frac{|U_{t-1}|}{OPT}$ elements at cost $c(OPT)$

$\Rightarrow \exists$ set such that $\frac{\#\text{uncovered elts covered}}{\text{cost of set}} \geq \frac{|U_{t-1}|}{c(OPT)}$

$\Rightarrow |U_t| \leq |U_{t-1}|\left(1 - \frac{c_t}{OPT}\right)$ ← cost of t^{th} set picked

$\Rightarrow |U_T| \leq n \cdot \left(1 - \frac{c_1}{OPT}\right)\left(1 - \frac{c_2}{OPT}\right) \cdots \left(1 - \frac{c_t}{OPT}\right) < n \cdot e^{-\left(\frac{\sum c_t}{OPT}\right)} \leq 1$

if $\sum c_t \geq OPT \ln n$. ∎

OK: can we do better?

: can we extend these ideas to other problems?

: can we analyze this algorithm better?

Can we do better?

Idea: Relax and round.

① Write an integer Linear program for the problem.

$$\min \sum_{S \in \mathcal{F}} c(S)\, x_S$$

$$\text{s.t.} \sum_{S : e \in S} x_S \geq 1 \quad \forall e \in U$$

$$x_S \in \{0,1\}.$$

Exactly captures the problem.

But NP hard to solve. (exactly set cover). because.

② "Relax" the problem to reals.

$$x_S \in \{0,1\} \longrightarrow x_S \in [0,1].$$

Now is a linear program.

Thm: Linear programs can be solved in polynomial time [Khachiyan 79]

③ "round" the fractional solution to integers

Simplest rounding: view each x_S as probability and pick S w.p. x_S.

$\Rightarrow$ element covered if at least one set containing it covered.

$$\Pr[e \text{ not covered}] = \prod_{S \ni e} (1 - x_S)$$

$$\leq \prod_{S \ni e} e^{-x_S} = e^{-\sum_{S \ni e} x_S} \leq e^{-1} \quad \text{if } \sum_{S \ni e} x_S \geq 1.$$

$\Rightarrow$ element uncovered with probability $\leq 1/e$.

Hmm. not good.

Expected $n(1/e)$ elements uncovered ☹

Pick more aggressively:

pick S w.p. $p_S = \min(x_S \cdot \ln n, 1)$

$\Rightarrow \Pr[e \text{ not covered}] \leq e^{-\sum_{S \ni e} p_S} = 0$ if some $p_S = 1$

else all $p_S = x_S \ln n$

$$\Rightarrow \Pr[e \text{ not covered}] \leq e^{-\sum_{S \ni e} p_S}$$

↑ same calcs above

$$= e^{-\sum_{S \ni e} x_S \ln n} \leq e^{-\ln n} = 1/n.$$

$\Rightarrow \Pr[e \text{ not covered}] \leq 1/n.$

What now? $\forall S$, pick S w.p. $p_S = \min(x_S \ln n, 1)$ independently.

$\forall e$ not covered

pick cheapest set containing e.

$$E[\text{cost}] = \underbrace{E[\text{cost of indep rounding}]}_{\leq \sum_S x_S \cdot \ln n = (LP)\cdot \ln n} + \sum_e \underbrace{\Pr(e \text{ not covered})}_{\leq 1/n} \cdot \underbrace{c(\text{cheapest set } \ni e)}_{\leq LP}$$

$\leq LP$

Fact: $LP \leq OPT$

$\Rightarrow E[\text{total cost}] \leq LP(\ln n + 1) \leq OPT(\ln n + 1)$

Yet another algorithm with cost $\leq (\log n)\, OPT$.

Can't we do better? We'll come back to this

Recap of relax-and-round

(1) write Integer LP for problem. $ILP = OPT$

(2) "relax" to LP

LP has fewer constraints (and minimization problem)

$\Rightarrow LP \leq ILP \leq OPT$

(3) Find solution of cost $\leq \alpha\, LP \leq \alpha \cdot OPT$.

Versatile and general technique!

Limitation of approach

= "integrality gap"

- Suppose ∃ instance such that $LP \le \frac{OPT}{\alpha}$ "LP cheats by α"

  then any solution we find must cost more than $\alpha \cdot OPT$

  so $ALG \ge \alpha \cdot LP$

- So this kind of proof cannot succeed if integrality gap is large.

  $\text{Intgap} = \max_I \frac{OPT(I)}{LP(I)}$

In other words, using LP as a surrogate for OPT

if LP far from OPT its a poor surrogate.

Example: elements are d-bit vectors.

- U = d bit vectors with $d/2$ 1s in them.

  $|U| = \binom{d}{d/2} \doteq \Theta\left(\frac{2^d}{\sqrt{d}}\right)$ $\Rightarrow$ $d \doteq \log n$.

- Sets are $S_i = \{x \mid x_i = 1\}$. $\Rightarrow$ d sets.

- Optimal integer solution has at least $d/2$ sets.

  Else some element not covered.

- optimal fractional solution.

  Let $x_{S_i} = 2/d$ for all i

  $\Rightarrow \sum_{S \ni e} x_S \ge d/2 \cdot 2/d = 1$.

  cost of LP solution $= \sum_{S \in \mathcal{F}} 2/d = d \cdot 2/d = 2$.

  $\Rightarrow$ gap $= \frac{d/2}{2} = \Omega(d)$.

OK. saw greedy algo $\leq OPT \cdot \ln n$ ← and tight example
relax+round $\leq OPT(\ln n + 1)$ ← and integrality gap.

What else can we do?

- local search algo: also $OPT \ln n$. (another time)

Is there a barrier to getting beyond $\ln n$?

— X —

Sadly, yes! (or excitingly, yes!)

Thm [Lund-Yannakakis, Feige, Dinur-Steurer]
If there an $(1-\varepsilon)\ln n$-approximation for Set Cover $\Rightarrow P = NP$.

So we have an optimal algo for set cover, unless $P = NP$!

— X —

Close cousin of Set Cover
– Max k-coverage.

Given $U, \mathcal{F}$, and k (integer)
Find k sets in $\mathcal{F}$ whose union is as large as possible!

Algo: greedy (as before).

Claim: $OPT - A_t \leq \left(1 - \frac{1}{k}\right)^t \cdot OPT$

(OPT: covered by OPT; A_t: covered by algo after t steps)

Pf: Initially $A_t = 0 \Rightarrow OPT \leq \left(1 - \frac{1}{k}\right)^0 OPT$ ✓

Now: at step t, at least $OPT - A_{t-1}$ elements in OPT uncovered

$\Rightarrow \exists$ set that covers $\geq \frac{OPT - A_{t-1}}{k}$ elements.

$$\Rightarrow A_t - A_{t-1} \geq \frac{OPT - A_{t-1}}{k}$$

$$\Rightarrow (OPT - A_{t-1}) - (OPT - A_t) \geq \frac{(OPT - A_{t-1})}{k}$$

$$\Rightarrow (OPT - A_t) \leq (OPT - A_{t-1})\left(1 - \frac{1}{k}\right) \leq \left(1 - \tfrac{1}{k}\right)^{t-1} \cdot \left(1 - \tfrac{1}{k}\right) OPT$$

↑ inductive hypothesis ✓ ■

$$\Rightarrow OPT - A_k \leq \left(1 - \tfrac{1}{k}\right)^k \cdot OPT \leq \tfrac{1}{e} OPT$$

$$\Rightarrow \boxed{A_k \geq OPT\left(1 - \tfrac{1}{e}\right).}$$

$\Rightarrow$ greedy algo covers at least 63% of what OPT covers.

---

do better? ILP

$$\max \sum_e y_e$$

$$\text{s.t } y_e \leq \sum_{S \ni e} x_S$$

$$\sum_S x_S \leq k$$

$$y_e, x_S \in \{0, 1\}.$$

y_e = element covered $\Rightarrow$ at least one set picked.

LP.

$$\max \sum_e y_e$$

$$\text{s.t } y_e \leq \sum_{S \ni e} x_S$$

$$\sum_S x_S \leq k$$

$$0 \leq y_e, x_S \leq 1.$$

Round: Pick S w.p. x_S. $\Rightarrow$ $E[\#\text{sets picked}] = k$.

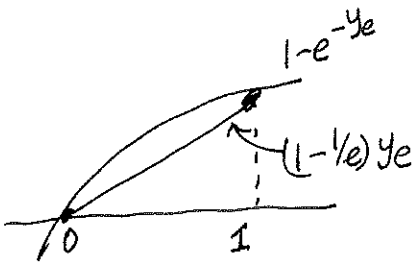
$$\Rightarrow \Pr[e \text{ picked}] = 1 - \prod_{S \ni e}(1 - x_S)$$

$$\geq 1 - e^{-\sum_{S \ni e} x_S} \geq 1 - e^{-y_e}$$

$$\geq (1 - 1/e)\, y_e \quad \text{for all } y_e \in [0,1]$$

concave function — linear function

$y_e = 0 \Rightarrow 0$	$y_e = 0 \Rightarrow 0$
$y_e = 1 \Rightarrow 1 - 1/e$	$y_e = 1 \Rightarrow 1 - 1/e$



$\Rightarrow$ pick k sets in expectation & cover $(1-1/e)$ LP elements in expectation.

How do we ensure pick k sets always and cover $(1-1/e)$ elements in expectation?

pipage rounding / correlated rounding — see HW or blog.

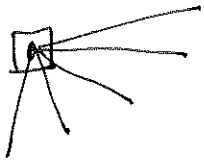
Again: Hardness

Thm [Feige 98]: if $\exists$ algo that always covers $(1 - 1/e + \varepsilon)$ fraction of OPT

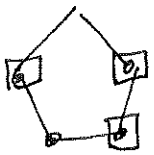
$\Rightarrow P = NP$!

Obs: setting $C = V$ is trivially a vertex cover.

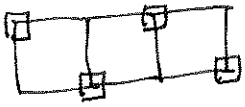
do better?



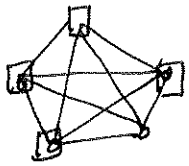
G = star
$\Rightarrow$ VC size = 1.



$G' = C_{2n+1}$
$\Rightarrow$ VC size = $n+1$



G = ladder
VC size = k.



$G = K_n \Rightarrow$ VC = $n-1$ size

Algorithm:

(1) Greedy: pick the vertex that covers most uncovered ~~set~~ edges.

Hmm: is $\Theta(\log m)$ approximation (~~something~~).
$= \Theta(\log n)$

$\Omega(\log m)$ follows from set cover
$\Omega(\log m)$ example — see HW or blog.

(2) How to ~~not~~ get ~~good~~ better algo?

- Let's think about OPT (the optimal solution).

Complicated to think directly.

- Can we give a lower bound on OPT(G)?

Say G has matching of size k

matching = edges that share no endpoint

then OPT $\geq k$

OPT(G) $\geq$ size of Any matching in G.

~~An extension to Vertex Cover~~

Algo: pick any edge that is not covered.
pick both its endpoints
repeat.

Fact: if k edges picked, Algo picks $2k$ vertices.

Fact: Algo stops when graph covered (by stopping criterion)

Fact: edges picked form a matching of size k

$\Rightarrow OPT \geq k$. $\Rightarrow Algo \leq 2k \leq 2OPT$.

$\Rightarrow$ Algo is 2-approximation
(for each graph G, $Algo(G) \leq 2 \cdot OPT(G)$)

Can we do better? What if we have costs? → cannot just do the same thing.

What other ideas can we use?

to be continued.

Duals:

$\min \sum c_v x_v$	$\min \sum_i y_{eu}$
s.t. $\sum x_u + x_v \geq 1$ $\forall uv \in E$	s.t. $\sum_{v: uv \in E} y_{uv} \leq c_v$ for all $v \in V$
$x \geq 0$	$y_{uv} \geq 0$.
(Primal)	(Dual).