The Set Cover Problem: LP-Rounding f Approximation

08 February 2021 09:59

Universe of elements $[n]$ elements U

Collection of subsets of U

$$\mathcal{S} = \{S_1, S_2, \dots S_m\} \text{ whr. } S_i \subseteq U$$

Goal:

choose $X \subseteq \mathcal{S}$ st X covers U

meaning

$$\boxed{\bigcup_{S \in X} S = U}$$

Feasible sol^n. $X = \mathcal{S}$

(ASSUME that all sets in $\mathcal{S}$ collectively cover U)

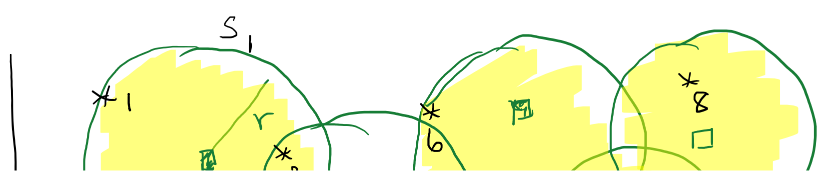
Objective Fn

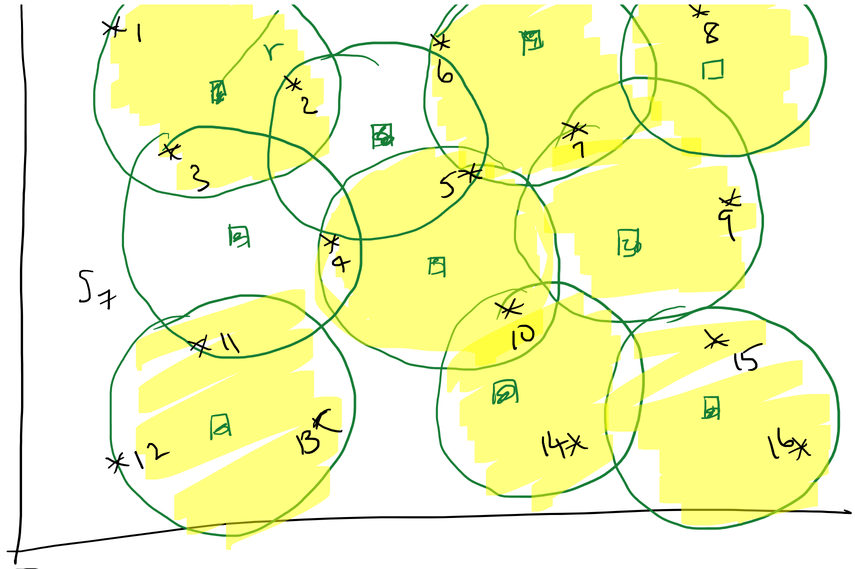
Find X of min. cardinality $|X|$

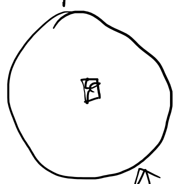
(ie) pick as few sets to cover the universe

EXAMPLE

Geometric Disk Coverage







each is a "set"

disc

each * is a neighbourhood in a city w/ $\geq$ 100 population

Goal

Choose min # of discs to cover the entire city (all *s)

In this example

$$U = \{e_1, e_2, \dots e_{16}\}$$

$$\mathcal{S} = \{S_1, S_2, \dots S_{10}\}$$

$$S_1 = \{e_1, e_2, e_3\}$$

$$S_7 = \{e_3, e_4, e_{11}\}$$ and so on.

In this case, we found a cover of 8 sets.

Is this the best ??

How close to optimal is it?

In general set cover, there is no real

In general set-cover, there is no real structure ~~to~~ sets & elements (ies) U & $\mathcal{S}$ can be arbitrary.

★ Set Cover is NP-complete [Garey ?? Johnson]

Need to settle for approximation algos.

Weighted Set Cover Problem:

Same as above, but each set is associated with a "cost" (non-negative)

POSSIBLE ALGORITHMS:

(1A) Greedy Algorithm

↳ @ any time, chose set which covers max. # remaining elements

{ seems reasonable for unit weights }

(1B) For greedy algo & costs, pick set which maximizes

$$\frac{\#\text{ new elts covered}}{\text{cost}}$$

— cost

(3) Additional idea: first include definite sets, which cover some elts uniquely then run greedy.

(4) Flip a coin for each set !
↳ What probability ?

(5) Write an LP and infer soln from it.

What does an LP for set cover look like?

x_i = variable for set $S_i \in \mathcal{S}$
w_i was the cost/weight given in input

$$\boxed{\begin{array}{lll} \text{Min} \sum_{i=1}^{m} w_i x_i & & \\ \sum_{i : e_j \in S_i} x_i \geq 1 & & \forall j \in 1..n \\ x_i \geq 0 & & \forall i \in 1 \dots n \end{array}}$$

m variables & n constraints

Fact ①

LP can be solved in poly(n, m) time

Let x^* denote the optimal solution

Lemma ①

$\sum w_i x_i^* \leq$ OPTIMAL SOLUTION'S COST

Proof :-

we can generate a feasible sol^n for LP using the optimal sol^n

Sps $\bar{X}$ is opt sol^n,

set $\bar{x}_i = 1$ if $S_i \in \bar{X}$

$= 0$ otherwise

Then easy to see that

$\sum w_i \bar{x}_i = \text{cost}(\bar{X})$

& all constraints are satisfied

Q How can we use these $\{x_i^*\}$ values to larger ... algorithm :

(A5) continued.
Use x_i^* as a "weight"/bias to picking S_i

[Pick set S_i with probability x_i^*]

repeat until feasible

(A6) Sort of greedy algo using x_i^*

Pick highest x_i^*, and choose that set

& repeat on remaining elements

(A7) Use the LP for finding a sol^n without solving the LP.
[using duals]
↑ TOMORROW.

Simple Algo (may be close to A6)

let "f" = $\max_{\text{elements } e}$ #sets covering e

elements e

f-approximation algorithm

$\boxed{\text{choose all sets st } x_i^* \geq \frac{1}{f}}$

Q1: Why is cost $\leq f \cdot OPT$?

Q2: why is it feasible ?

Ans(Q2): If all x_i^* for a particular elt are $< \frac{1}{f}$,

then how is the $\sum x_i^* \geq 1$ for it ?

✓

Ans (Q1): Let $L = \{ i : x_i^* \geq \frac{1}{f} \}$

$$\text{Cost(Alg)} = \sum_{i \in L} w_i \leq \sum_{i \in L} w_i f x_i^*$$

$$\leq f \sum_i w_i x_i^*$$

$$\leq f \cdot \text{OPT COST}$$

lemma 1

This Algo. can actually outperform greedy Algo (A1, A2)

greedy Algo (A1, A2)
if f is very small.

TOMORROW we'll present f-approx without solving LP!!

HW think about the dual of set cover LP.

Last class we saw an LP-based f-approximation

- Main drawback:
  LPs, while efficient (polynomial time) are slow for large datasets.
- Search online: running time of best LP solver?

Today's Lecture
Use LPs conceptually to design faster f-Apx Algo.

PRIMAL LP

$$\text{Min} \sum w_S x_S$$

$$\sum_{S: e \in S} x_S \geq 1 \quad \forall e \in U$$

$$x_S \geq 0 \quad \forall S \in \mathcal{S}$$

Variables x_S $\forall S \in \mathcal{S}$
Constraints for each elt $e \in U$

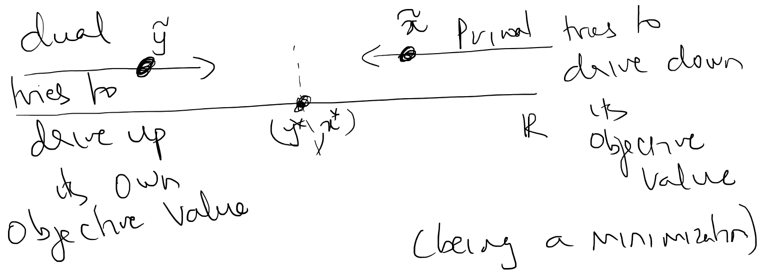
DUAL LP

$$\text{Max} \sum_{e \in U} y_e$$

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

$$y_e \geq 0 \quad \forall e$$

Variables y_e $\forall e \in U$
Constraints for each set.



Weak Duality
by the way we've constructed the dual,

if $\tilde{x}$ is feasible for primal
$\tilde{y}$ is feasible for dual,

then $\sum_{S \in \mathcal{S}} w_S \tilde{x}_S \geq \sum_{e \in U} \tilde{y}_e$

In particular, if $\bar{x}^*$ is the optimal set cover &

$\tilde{y}$ is any feasible dual,

$$\sum \tilde{y}_e \leq \sum_S w_S \bar{x}^* = \text{Cost(OPT set cover)}$$

In words

Any feasible dual sol^n gives a good lower bound on OPT.

ALGORITHM

Initialize $F = \phi$
↑ solution

Initialize $\tilde{y}_e = 0 \quad \forall e$

While F is not feasible set cover,

Increase all unfrozen $\tilde{y}_e$ at uniform rate

Some dual constraint $\sum_{e \in S} \tilde{y}_e = w_S$ becomes tight

Pick set S, and freeze all $\tilde{y}_e$ for $e \in S$
(add S to F)

DUAL LP

$$\text{Max} \sum_{e \in U} y_e$$

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

$$y_e \geq 0 \quad \forall e$$

HW: Q: How do we implement this algorithm efficiently?

↑ what are data structures, what is the running time, etc.

Observations :-

① If F is not feasible, then there are unfrozen y_e variables

② $\{\tilde{y}_e\}$ is always a feasible dual solution

③ How do we compare the cost(F) wrt Optimal soln?

for any set $S \in F$, we know

$$w_S = \sum_{e \in S} \tilde{y}_e$$

$$\Rightarrow \text{Cost}(F) = \sum_{S \in F} w_S = \sum_{S \in F} \sum_{e \in S} \tilde{y}_e$$

$$\leq f \cdot \sum_{e \in U} \tilde{y}_e$$

Weak Duality $\leq f \cdot OPT$. ☒

Dual was used to give us 2 ideas

1. good lower bound on OPT
2. good idea which sets to include.

PRIMAL-DUAL FRAMEWORK

Q These algos are good when 'f' is small, but what do we do when 'f' is large?

- Back to solving the LP.

$$\text{Min} \sum w_S x_S$$

$$\sum_{S: e \in S} x_S \geq 1 \quad \forall e \in U$$

$$x_S \geq 0 \quad \forall S \in \mathcal{S}$$

Let's solve the LP, and $\{x^*\}$ is Optimal LP sol^n.

RANDOMIZED ALGO

repeat T times

$\forall S \in \mathcal{S}$, choose S w.p x_S^*

We want to claim for some reasonable T, both ① cost is good ② sol^n is feasible

---

Sps $T = 1$:

Let $Y_S = 1$ if S is included.

Expected cost Incurred in one round

$$= E\left[\sum Y_S w_S\right] = \sum E[Y_S] \cdot w_S$$

$$= \sum w_S x_S^*$$

$$\leq OPT$$

In T rounds

$E[\text{Algo Cost}] \leq T \cdot OPT \leftarrow$ linearity of expectation

---

Fix e

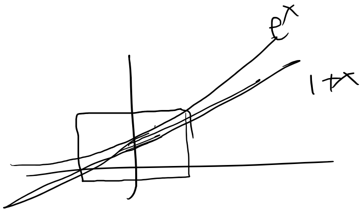
Pr[elt 'e' is not covered in 1 round] *

Fix e

Pr [elt 'e' is not covered in 1 round]

$$\prod_{S:e\in S}(1-x_S^*) \le \prod_{S:e\in S} e^{-x_S^*} = e^{-\sum_{S\ni e} x_S^*} \le 1/e$$

$\forall x \in \mathbb{R}$ $\boxed{1+x \le e^x}$



Intuitively

One round covers 1/2 the elements
$\Rightarrow$ if $T = \log n$, we should ideally cover all elts.

If f is very large $\gg \log n$, say,
RND. ROUNDING gives a $\Theta(\log n)$ approximation

From yesterday's lecture, we get that

① $E[\text{cost of one round}] \le \sum w_S x_S^* \le OPT$

② $\forall e$, $Pr[e \text{ is uncovered}] \le 1/e$

By repeating this process T times, we get

(1) $E[\text{Cost of Algo}] \le T \cdot OPT$

(2) $\forall e$, $\Pr[e \text{ is uncovered after all } T \text{ rounds}] \le \left(\frac{1}{e}\right)^T$

set $T = 2\ln n$ [can be improved, think]

to get

1) $E[\text{Cost}] \le 2\ln n \cdot OPT$

2) $\underbrace{\forall e, \Pr[e \text{ is uncovered}] \le \frac{1}{n^2}}$

(2) $\Rightarrow$ $\Pr[\text{Algo is infeasible after } T \text{ rounds}]$

$= \Pr[\exists \text{ some uncovered element}]$

$= \Pr[e_1 \text{ is uncovered or } e_2 \text{ is uncovered or } \ldots \text{ } e_n \text{ is uncovered}]$

$\overset{\text{UNION}}{\underset{\text{BOUND}}{\le}} \sum_{i=1}^{n} \Pr[e_i \text{ is uncovered}]$

$\le n \cdot \frac{1}{n^2}$

$= \frac{1}{n}$

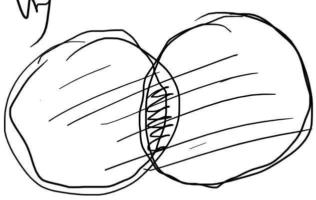
$\Pr(A \cup B) \le$

$\Pr(A \cup B) \leq$

$\Pr(A) + \Pr(B)$

regardless.

proof by picture



by setting $T = 2\ln n$,

a) $E[\text{Cost}(\text{Alg})] \leq 2\ln n \cdot \sum w_S x_S^*$

b) $\Pr[\text{Alg is INFEASIBLE}] \leq 1/n$

$\Pr[\text{Cost}(\text{Alg}) \geq 4\ln n \cdot \sum w_S x_S^*] \leq \frac{1}{2}$

$\Rightarrow$

$\Pr[\text{Alg is infeasible (OR) has cost} \geq 4\ln n \sum w_S x_S^*]$

$\leq \frac{1}{2} + \frac{1}{n} \leq \frac{2}{3}$

we can say:

THEOREM

Alg outputs a feasible sol'n with cost $\leq 4\ln n \sum w_S x_S^*$ with probability $\geq 1/3$.

Just rerun whole algo if infeasible to "boost" success Pr.

Good News! Does well if 'f' is very large as guarantee is indep. of f

Can further tighten the analysis to get $\ln n + \Theta(\ln \ln n)$ Approx.

Drawback

Is the need for solving an LP to begin with.

Set Cover: Greedy Algorithm Analysis I

08 February 2021 10:56

Given sets $\mathcal{S}$ and universe U

cost w_S for $S \in \mathcal{S}$

Algo:

- Initialize $R = U$ (remaining elts to be covered)

While $R \neq \phi$

Choose $S \in \mathcal{S}$ of minimum $\frac{w_S}{|R \cap S|}$

update $R = R \setminus S$

THM: Algo is a $\Theta(\log n)$-approximation

Todo: Think of what the running time of this algo will be?

Proof of THM:-

let $O^* \subseteq \mathcal{S}$ be the optimal sol^n

$$OPT = \sum_{S \in O^*} w_S$$

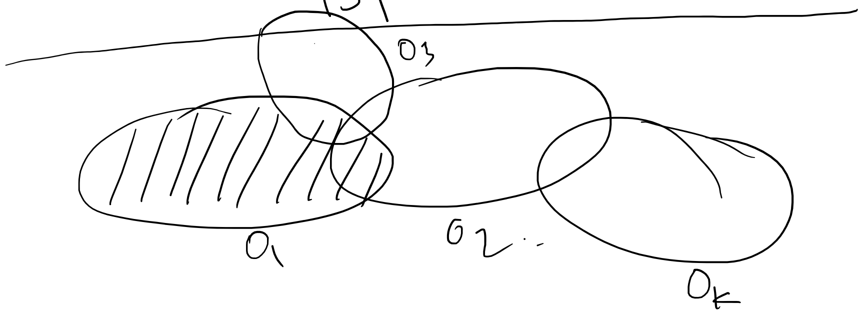
& suppose $|O^*| = k$

Claim

the first set alg includes satisfies

$$\frac{w_S}{|S|} \leq \frac{OPT}{n}$$

$$\frac{\quad}{|S|} = \frac{\quad}{n}$$



Let n_i denote the # of elements we assigned to set O_i in OPT

Order the sets in OPT
$O_1, O_2, \ldots O_K$

$\forall e$, assign e to the <u>first</u> set containing it

$$\left.\begin{aligned} \sum_{i=1}^{k} n_i &= n \\ \sum_{i=1}^{k} w_i &= OPT \end{aligned}\right\} \Rightarrow \exists \text{ set in OPT st } \frac{w_i}{n_i} \le \frac{OPT}{n}$$

Algo is greedy, so we will definitely pick a set S st $\frac{w_S}{|S|} \le \frac{OPT}{n}$

For each covered element, give it a price $= \frac{w_S}{|S|}$.

$\Rightarrow$ Total price charged to all covered elements

$= W_S$.

More generally, sps R is the set of remaining elts and algo picks a set S at this step

LEMMA

Then $\dfrac{W_S}{|S \cap R|} \le \dfrac{OPT}{|R|}$

Same proof as above, but apply to Reduced instance over R instead of U

Let $e_1, e_2, e_3, \ldots, e_n$ be the order in which algo covers the elements

lets look at the prices charged to these elements

remaining $n-k_1$ | remaining $n-k_2$ . . .

ELTS $\underbrace{e_1 \; e_2 \; e_3 \; e_4 \cdots e_{k_1}}$ | $\underbrace{e_{k_1+1} \cdots e_{k_2}}$ | $\underbrace{e_{k_2+1} \cdots e_{k_3}}$ $\cdots e_n$

Prices $\le \frac{OPT}{n}$ $\le \frac{OPT}{n}$ $\le \frac{OPT}{n}$ . . . . $\le \frac{OPT}{n}$ | $\le \frac{OPT}{n-k_1}$ . . . . $\frac{OPT}{n-k_1}$ | $\frac{OPT}{n-k_2}$ . . . $\frac{OPT}{n-k_2}$ . . . . .

Also

$\sum_{e} \text{Prices}(e)$ = Total cost of Algorithm

In each step where Alg picks sets,
it newly covers
$|S \cap R|$ elements,
each of which is
charged a price of

$$\frac{w_S}{|S \cap R|}.$$

Cost(Alg) = Total Price charged to all elts

~~Back to price chart~~

$$\begin{array}{cccccccccccc} e_1 & e_2 & e_3 & \cdots & e_{k_1} & e_{k_1+1} & \cdots & e_{k_2} & e_{k_2+1} & \cdots & e_{k_3} & \cdots & e_n \\ \le \frac{OPT}{n} & \le \frac{OPT}{n} & \cdots & & \le \frac{OPT}{n} & \frac{OPT}{n-k_1} & \cdots & \le \frac{OPT}{n-k_1} & \le \frac{OPT}{n-k_2} & \cdots & \le \frac{OPT}{n-k_2} & \cdots \end{array}$$

To get a reasonably clean form expression,
let's be a bit more lossy

$$\begin{array}{ccccccccccc} e_1 & e_2 & e_3 & \cdots & e_{k_1} & e_{k_1+1} & \cdots & e_{k_2} & e_{k_2+1} & \cdots & e_{k_3} & \cdots & e_n \\ \le \frac{OPT}{n} & \le \frac{OPT}{n-1} & \le \frac{OPT}{n-2} & \cdots & \frac{OPT}{n-k_1+1} & \frac{OPT}{n-k_1} & \cdots & \frac{OPT}{n-k_2+1} & \frac{OPT}{n-k_2} & \cdots & & & \frac{OPT}{1} \end{array}$$

Total price

$$\le OPT \left(\boxed{\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + 1} \right)$$

$$= OPT \, (H_n)$$

$$\approx OPT \cdot \ln n.$$

Intuition

$$\int \frac{1}{x}\,dx = \ln x$$

Good: No need to solve LP

Not so good: $\ln n$ factor vs OPT cost

whereas LP vs $O(\ln n)$

wrt LP optimal cost.

Friday

One more analysis of greedy Algo.

Set Cover Greedy Algorithm Analysis II

08 February 2021 10:57

Story so far..

Set Cover

$\downarrow$

f-approx (LP Rounding) f: max # sets that cover an element

$\downarrow$

f-approx (Primal-Dual) (better b/c we don't solve any LP)

$\downarrow$

f could be very large in general

$\downarrow$

Useful when $f \gg \log n$ $\left\{ \begin{array}{l} O(\log n)\text{-approximation (LP + Randomized Rounding)} \\ \downarrow \\ \Theta(\log n)\text{-approximation (greedy algo).} \end{array} \right.$

$\downarrow$

Today: Another analysis of greedy algorithms.

ANALYSIS OF GREEDY ALGORITHM USING LINEAR PROGRAMMING.

Problem Recap

Given U (universe of n elts and $\mathcal{S} = \{S_1, S_2, \ldots S_m\}$ of m sets, with each set $S \in \mathcal{S}$ having a cost $w_S \geq 0$, pick min cost collection of sets to cover U (all elts).

Recap (Greedy Algorithm)

- Start with remaining element set $R = U$
- Until $R = \phi$
  - choose set $S \in \mathcal{S}$ which minimizes $\dfrac{w_S}{|R \cap S|}$
  - update $R = R \setminus S$

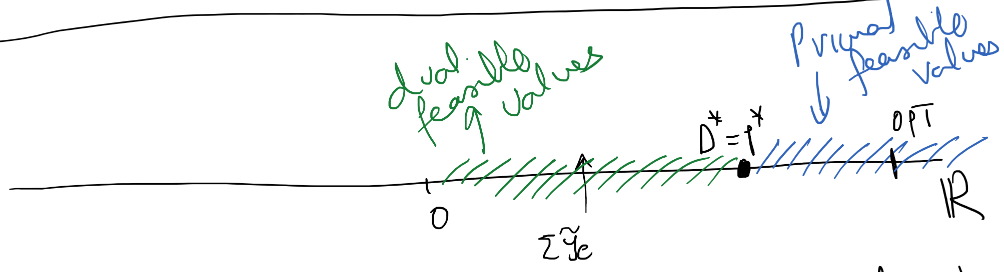
Recap LP Relaxation & Dual for Set Cover

$$\min \sum_{S \in \mathcal{S}} w_S x_S \qquad\qquad \max \sum_{e \in U} y_e$$

$$\forall e \in U \quad \sum_{S : e \in S} x_S \geq 1 \qquad\qquad \forall S \in \mathcal{S} \quad \sum_{e \in S} y_e \leq w_S$$

$$x_S \geq 0 \qquad\qquad y_e \geq 0$$

Primal Relaxation of Set Cover | Dual of Primal



P^* = primal optimal

OPT = Actual set cover optimal

D^* = Dual optimal

Agenda for today:-

We'll construct a dual feasible solution $\{\tilde{y}\}$ s.t

$$\text{Cost (Greedy Algo)} \leq \lambda \cdot \sum \tilde{y}_e$$

for some suitable λ.

$\Rightarrow$ Cost (Greedy Algo) $\leq \lambda \cdot$ LPOPT
$\leq \lambda \cdot$ OPT

BACK to greedy :-

- Start with remaining element set $R = U$
- Until $R = \phi$
  - choose set $S \in \mathcal{S}$ which minimizes $\dfrac{w_S}{|R \cap S|}$
  - update $R = R \setminus S$

ANALYSIS

Construct dual values $\tilde{y}_e$ such that they are the "prices" elements incur to be covered.

Ex: First step, algo picks a set S_1

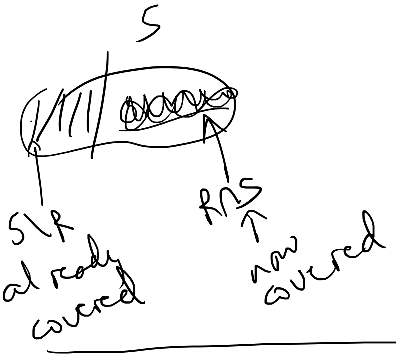
ITS cost $= w_{S_1}$

IT covers $|S_1|$ elts.

$\Rightarrow$ we can try to set $y_e = \dfrac{w_{S_1}}{\lambda |S_1|}$ for all $e \in S_1$.

In general, if R is set uncovered elts, and greedy picks a set S, we assign a price of

$$\tilde{y}_e = \frac{w_S}{\lambda |S \cap R|} \quad \forall e \in$$



Lem ①

When greedy algo finishes, we'd have set a price for all elements.

Lem ②

All $\tilde{y}_e \geq 0$

Lem ③

$$\sum_{e \in U} \tilde{y}_e = \frac{\text{Cost (Greedy Algorithm)}}{\lambda}.$$

Lem ④

$\{\tilde{y}_e\}$ is "~~almost~~" feasible for the dual problem.

(i.e)

$$\sum_{e \in S} \tilde{y}_e \leq \cancel{\lambda} \cdot w_S \qquad \forall S$$

~~for suitable λ~~

$\Rightarrow$

$$\text{cost(Greedy)} = \lambda \cdot \sum \tilde{y}_e \quad \text{(from Lem ③)}$$

and

$\{\tilde{y}_e\}$ is dual feasible

$\Rightarrow \sum \tilde{y}_e \leq D^* = P^* \leq OPT$.

Need to show: there is suitably small value of λ

s.t

(*) $\sum_{e \in S} \tilde{y}_e \leq w_S$ for all sets $S \in \mathcal{S}$.

Fix a set $S \in \mathcal{S}^E$

note: it may or may not have been selected by greedy

$\boxed{e_1 \quad e_2 \quad e_3 \quad \cdots \quad e_\ell}$ are the elements of S

let us order them by when they got covered in greedy algo.

e_1 got covered first among elts of S
e_2 got covered second, etc ....

Greedy algo assigns these elts prices based on when they got covered.

can $\tilde{y}_{e_1}$ be really large?

Obs ①

$\tilde{y}_{e_1} \leq \tilde{y}_{e_2} \leq \cdots \leq \tilde{y}_{e_\ell}$

(b/c greedy chooses min. price rule)

Obs ② -

min. price)

Obs ② $\tilde{y}_{e_1} \le \frac{w_S}{\lambda \cdot \ell}$ ← because greedy had a choice of picking S, and it went with best price choice.

Similarly,

$\tilde{y}_{e_2} \le \frac{w_S}{\lambda(\ell-1)}$

& in general

$\tilde{y}_{e_j} \le \frac{w_S}{\lambda(\ell-j+1)}$

greedy always has S as a choice!! offering good price.

So,

$$\sum_{e \in S} \tilde{y}_e \le \frac{w_S}{\lambda \ell} + \frac{w_S}{\lambda(\ell-1)} + \frac{w_S}{\boxed{\quad}} \quad \frac{w_S}{\lambda}$$

$$= \frac{w_S}{\lambda}\left[\frac{1}{\ell} + \frac{1}{\ell-1} + \cdots + 1\right]$$

$$= \frac{w_S}{\lambda} \cdot H_\ell$$

Harmonic#(ℓ) $\approx \ln(\ell)$.

So we can set

$$\boxed{\lambda = \max_{S \in \mathcal{S}} H_{|S|}} \le \ln n$$

$\Rightarrow$ $\boxed{\tilde{y}_e \text{ will be dual feasible for this choice of } \lambda}$

$\tilde{y}_e$ choice of λ

$$
\begin{aligned}
\text{Cost(Greedy)} &= \lambda \cdot \sum \tilde{y}_e \\
&\le \lambda \cdot D^* \quad (\tilde{y}_e \text{ is dual feasible}) \\
&= \lambda\, p^* \\
&\le \lambda \cdot OPT \quad \boxed{}
\end{aligned}
$$

where $\lambda = \max_{S \in \mathcal{S}} H_{AR}(|S|)$

Advantages over earlier analysis?

(1) - factor is better
$\max_S \ln |S|$ is better than $\ln n$

(2) - It's bound is wrt p^* which could be much lower than OPT

$\boxed{}$

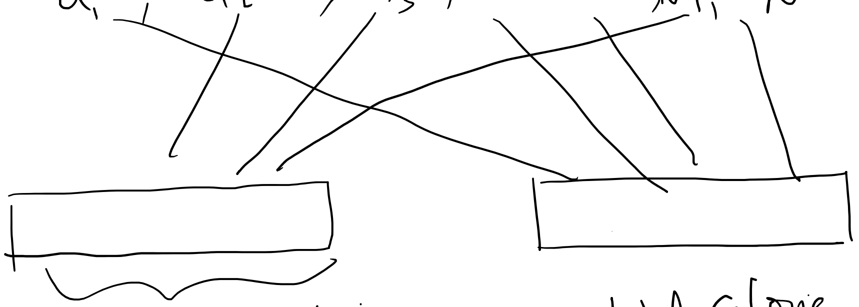
A FAIR ALLOCATION PROBLEM

There are 'N' food items

Each has a specific caloric value
$[0, 1]$

We want to split these items into 2 groups such that the total calorie count in each group is "close" to each other.

$a_1, a_2, a_3, \ldots a_{N-1}, a_N$



T_1 = total calorie count $\approx$ total calorie count $= T_2$

In other words,

$|T_1 - T_2|$ as small as possible.

Possible Algorithms :-

① Target is $\frac{\text{SUM}}{2}$,
so look for knapsack for smallest possible 'd'

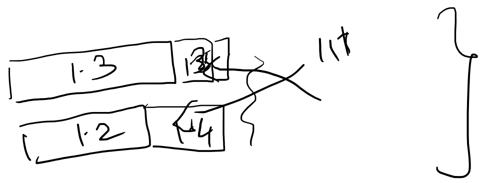
$$\boxed{\begin{array}{l}\text{Max } \sum \text{total Calorie} \\ \text{Total Calorie} \le \frac{\text{SUM}}{2} + d\end{array}}$$

"or some variant of this"

Issue: knapsack / Subset-Sum problem is not poly-time.
So, need to resort to Approximation Algorithms.

$\boxed{\text{Q: What sort of guarantees can we get?}}$

(2) Sort elts in descending order
greedy assignment to bucket of lower total weight



↑ at end of process, how bad can the difference be?

Ans: $|T_1 - T_2| \le$ Max wt ≤ 1.

let's say we're happy with this

Q2

What if there are two criteria to be fair over?

Items	1	2	3	...	N
CALORIE	a_1	a_2	a_3	...	a_N
PROTIEN	b_1	b_2	b_3	...	b_N

PROTIEN b_1 b_2 b_3 - - - - - - b_N

Let's assume all a_i & b_i are between 0 & 1.

Again, partition into 2 buckets to be as fair on both criteria?

<u>Optimistic Goal</u>:

Regardless of how large N is, can we find an allocation of 'discrepancy' $\leq O(1)$?

Possible Algorithms?

1. Keep a target $\begin{bmatrix} \frac{\sum A_i}{2} \\ \frac{\sum B_i}{2} \end{bmatrix}$

   and keep adding "largest" item as long as feasible?

2. Add next item to the bin "furthest" from the target

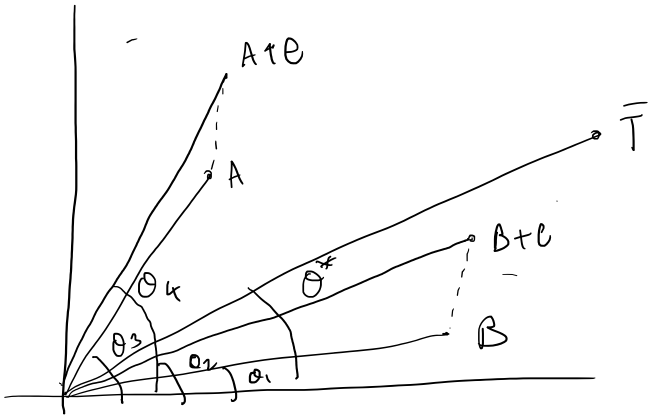
$$\left\| \begin{pmatrix} \frac{\sum A_i}{2} \\ \frac{\sum B_i}{2} \end{pmatrix} - \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \right\|_2$$

Q1 Why l_2 distance?

Q2 what item is next? any item?

3. Let target vector $\bar{T} = \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \begin{pmatrix} \frac{\sum A_i}{2} \\ \frac{\sum B_i}{2} \end{pmatrix}$

(T_2) $\left(\frac{\Sigma B_i}{2} \right)$



Compare $\theta_2 - \theta_1$ with $\theta_3 - \theta_4$

I don't know what analysis we got here?

THOUGHT EXERCISE

More Challenging :-

The a_i & b_i values can be -ve also!!
in range $[-1, 1]$

Now which algo works?

Many "greedy"-like algorithms don't work,
Meaning the discrepancy won't be a constant.

If you can think of greedy-like Algo which has $O(1)$-discrepancy, please let me know!

LPs to the rescue!!

Variables: x_i for i^{th} item.

In my mind, $x_i = +1$ means put it in first bin

$x_i = -1$ means put it in 2^{nd} bin

Ideal Formulation

$$\left.\begin{aligned} &\min \lambda \\ &\left|\sum a_i x_i\right| \le \lambda \\ &\left|\sum b_i x_i\right| \le \lambda \\ &x_i \in \{-1, 1\} \end{aligned}\right\}$$

Cant hope to solve this efficiently b/c Integer Programming is NP-hard.

Relax to

$$\boxed{\begin{aligned} &\min \lambda \\ &\left|\sum a_i x_i\right| \le \lambda \\ &\left|\sum b_i x_i\right| \le \lambda \\ &-1 \le x_i \le 1 \\ &\lambda \ge 0 \end{aligned}}$$

Linear Program! can solve in poly-time

Abs. Value constraints

$$\sum a_i x_i \le \lambda$$

& $\sum a_i x_i \ge -\lambda$

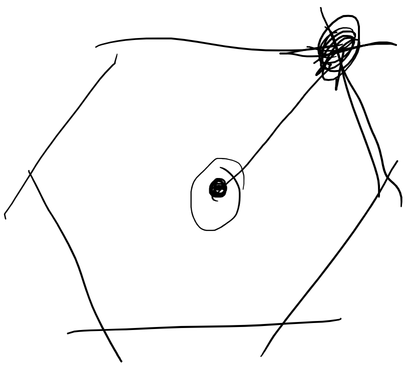
" can be written as

A-priori, this LP doesn't seem useful.

All $x_i = 0$, $\lambda = 0$ satisfies

$$\boxed{\begin{aligned} \sum a_i x_i &= 0 \\ \sum b_i x_i &= 0 \\ x_i &\geq -1 \\ x_i &\leq 1 \end{aligned}}$$

is a polytope in 'n' dimensional space.

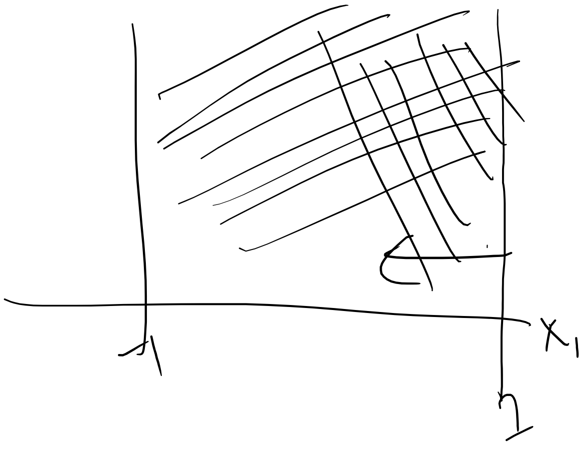


All $(0, 0, \ldots 0)$ is a feasible point, but its in the interior of the polytope.

But, we can ask the LP solver to return an 'extreme' point of this!

Basic feasible solution

It arises as the intersection of 'n' hyperplanes which are satisfied at equality



Implies that N-2 variables are actually forced to be -1 or +1.

Just "round" the 2 fractional variables to ± 1

Total harm done to constraints is ≤ 2 in total!!

Example of a situation where non LP based algos are hard to think about, but LPs make it super easy!

Knapsack Problem

17 February 2021 08:03

Given 'n' items i with item i having profit $p_i \geq 0$ and size $s_i \geq 0$, and budget $B \geq 0$,
Choose a subset $X \subseteq [n]$ of items s.t. $\sum_{i \in X} s_i \leq B$ [total size fits into the bag], and total profit $\sum_{i \in X} p_i$ is maximized.

$$\# \text{input bits} = \sum_{i \in [n]} \text{bitcomplexity}(s_i, p_i)$$

Polytime Algo ⇒ polynomial running time in # input bits.

GOAL for TODAY

PTAS (Polynomial-Time Approximation Scheme) for Knapsack.

Given any CONSTANT $\epsilon \geq 0$, the algorithm runs in time poly(inputsize), and computes a soln with profit $\geq (1-\varepsilon) \cdot$ OPTIMAL VALUE.

Ex: Algo can runtime $n^{1/\varepsilon}$ is allowed

Ex: Algo can runtime $n^{\dots}$ ...

Algo w/ runtime $\left(\frac{1}{\varepsilon}\right)^n$ is not allowed

Even better if Algo runs in time poly$(n, 1/\varepsilon)$.

FULLY POLYNOMIAL-TIME APPROXIMATION SCHEMES (FPTAS).

$\boxed{\text{Allows for a clean trade-off b/w solution Quality and Running Time}}$

No real limit on how good our solutions can be.

ALGORITHMS for KNAPSACK

① Greedy: Choose item w/ max $\frac{p_i}{s_i}$, insert it and repeat.

Can be problematic in some cases.

① In second round, chosen item can have size > remaining budget, so need to filter it out.

In fact, it's an issue in all the rounds.

A:

the rounds.

Q: Can we be making a mis-step by filtering out these items in step 2 onwards

Ex: budget = 100

Item ①: $p_1 = 10$
$s_1 = 1$

Item ②: $p_2 = 999$
$s_2 = 100$

But algo is not too bad if we allow a slight violation of the constraint (let's say factor 2).

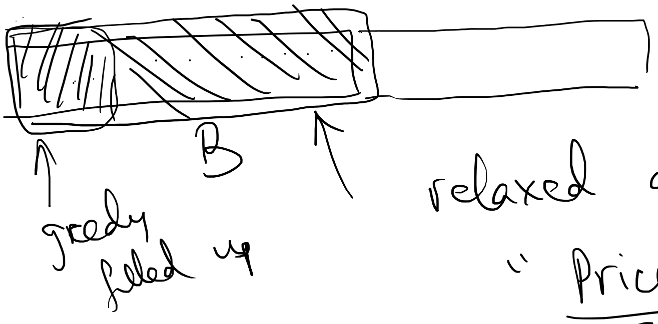
But in real-life some constraints are $\boxed{\text{HARD}}$, cannot be violated.

We'll devise an FPTAS based on dynamic Programming.

I think greedy algo will achive ≥ Optimal profit if it can use up to 2-times the budget.

" (but OPT - sol^n must respect the budget).

The previous algo was bad only b/c it had unused capacity - which it couldn't fill



relaxed greedy has as good "$\frac{\text{Price}}{\text{Size}}$" ratio as any item in OPT, for the first B size.

Try to formalize this proof

$\Rightarrow$ No item which OPT selects will be filtered out by our algorithm, for the first B size.

Back to dynamic Programming

Lets consider knapsack, but all profits p_i are integer valued

and $0 \le p_1 \le p_2 \le p_3 \le \dots \le p_n$.

Sizes & budget can be real-valued still.

Goal: Find DP for knapsack which runs in time poly(n, P_n)

$\boxed{\text{Let } V(i, p) \text{ denote the size of the minimum size subset of [i] items which achieves a profit of exactly } p}$

first i items

In general, recursive form of

$$V(i, p) = \min\left(V(i-1, p),\ V(i-1, p-p_i) + s_i \right)$$

and base case:

$$V(1, p_1) = s_1$$

$$V(1, p) = \infty \quad \forall\ p \neq p_1 \text{ or } 0$$

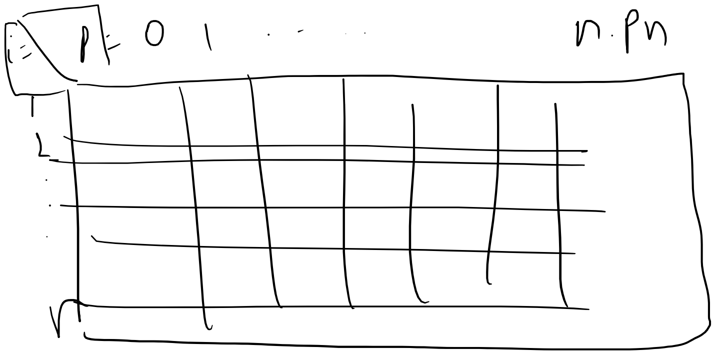
$$V(1, 0) = 0$$

Eventually we want max p s.t. $V(n, p) \leq B$

Clearly desired $p \leq nP_n$, Upper bound on max profit of OPT

Total run time $\propto$ # cells in DP table

$\propto$ $\boxed{n^2 \cdot p_n}$.



"Add a ROUNDING / DISCTRETRATION"
step to reduce a
general knapsack
instance to
this form of integer-valued
instance

Example

$$\begin{cases} p_1 = p_2 \\ s_1 > s_2 \end{cases}$$

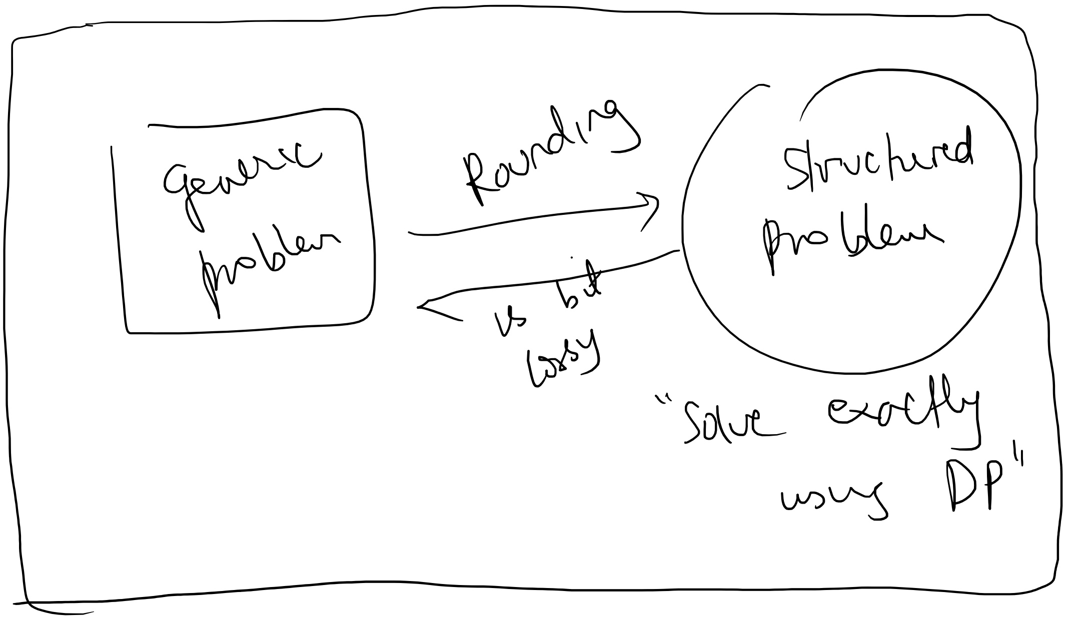
$B = s_2$

$V(2, p_1) = \min\big(V(1, p_1),\ V(1, 0) + s_2\big)$

$= \min(s_1, s_2)$

$= \min(s_1, s_2)$

$= s_2$



From general instances to "discretized" instances

Given n items $0 \le p_1 \le \dots \le p_n$ (need not be integers) and sizes $s_1, s_2, \dots s_n$, how do we convert this to a discrete instance on the profits?

Original instance I

Firstly, recall that $OPT \le n \cdot p_n$

Moreover, let's assume that all sizes $s_i \le B$ (else we can discard such items)

$\Rightarrow OPT \geq p_n$

Now, consider items which are profit $< \frac{\varepsilon \cdot p_n}{n}$

Idea:

Even if all these items are aggregated,

the total ~~profit they~~ $\cancel{\varepsilon\, p_n}$ $\leq \varepsilon \cdot OPT$

What if we "round down" all profits to the nearest multiple of $\frac{\varepsilon p_n}{n}$??

Instance $\hat{I}$: For each item i, set $\hat{p}_i$ to be "rounded-down" value.

Now, for any subset of items S,

what is

$$0 \leq \sum_{i \in S} (p_i - \hat{p}_i) \leq \frac{\varepsilon \cdot p_n}{n} \cdot |S| \leq \varepsilon p_n \leq \varepsilon \cdot OPT$$

In particular,
the optimal value of new instance $\widehat{OPT}$
$\geq (1-\varepsilon)\,OPT$

Moreover, since we are only rounding down values a good soln in new instance will be at least as good in original problem also.

In new instance, all profits are <u>I</u>ntegral multiples of $\frac{\varepsilon P_n}{n}$.

Let's focus on an "equivalent" instance $\bar{I}$ where item i has

profit $\bar{P}_i = \dfrac{\hat{P}_i}{\left(\frac{\varepsilon P_n}{n}\right)}$

$\Rightarrow$

$$\overline{OPT} = \frac{\widehat{OPT}}{\left(\frac{\varepsilon P_n}{n}\right)}$$

<u>Obs ①</u> $\bar{P}_i$ are integral for all i

Obs ② $\max \bar{P}_i \leq \dfrac{P_n\, n}{\varepsilon P_n} \leq \dfrac{n}{\varepsilon}$.

$\Rightarrow$ can use dynamic program to

Runtime of DP to solve $\bar{I}$
exactly solve the instance $\leq O(n^2 \cdot \frac{n}{\varepsilon})$ $\bar{I}$

$= O(n^3/\varepsilon)$

Time — Polytime regardless of how large/small the values can be!

DP for $\bar{I}$ finds a sol'n with profit $\geq \overline{OPT} = \frac{\widehat{OPT}}{\varepsilon p_n/n}$

Same sol'n has profit $\geq \widehat{OPT}$ for $\hat{I}$

$\Rightarrow$ Same sol'n has profit $\geq \widehat{OPT}$ for I

$\geq (1-\varepsilon) \cdot OPT$ ∎

Point raised in class:
To compute residual instance $\hat{P}$, we need to find

$$\left\lfloor \frac{\varepsilon P_n}{n} \right\rfloor$$

How much time does this operation take?

Can do some form of long division poly(log P_n) time.

$\boxed{\text{Run Time of overall algo} = \text{poly}(n, \log P_n, \tfrac{1}{\varepsilon})}$

FPTAS (fully poly-time Apx Scheme).

One could sometimes settle for

PTAS

(where dependence on $\frac{1}{\varepsilon}$ could be very bad)

Turns out there are some problems where neither FPTAS or PTAS is possible, if $P \neq NP$

Example

$\boxed{\text{set Cover}}$

$\boxed{\text{3SAT}}$

$\boxed{\text{HARDNESS OF APPROXIMATION}}$

[Feige, Moshkovitz],

There exists constant $\varepsilon > 0$ s.t. if we have a poly-time $(1-\varepsilon)\ln n$ approximation algorithm for Set Cover, then we can solve 3SAT in poly-time.

[i.e. P = NP].

Similarly,

$\exists\ \varepsilon > 0$ st $\left(\frac{7}{8}+\varepsilon\right)$-approx for Max3SAT

$\Rightarrow$ we can solve 3SAT in poly time (P = NP)

PCP Theorem

Miracle, because it says exact 3SAT is no harder than $(7/8+\varepsilon)$-approximation

[Håstad, 2000]

$\rightarrow$ NP-Complete to do better

7/8 apx trivial in poly time

Similarly set cover

NP-complete $\times$

$(1-\varepsilon)\ln(n)$

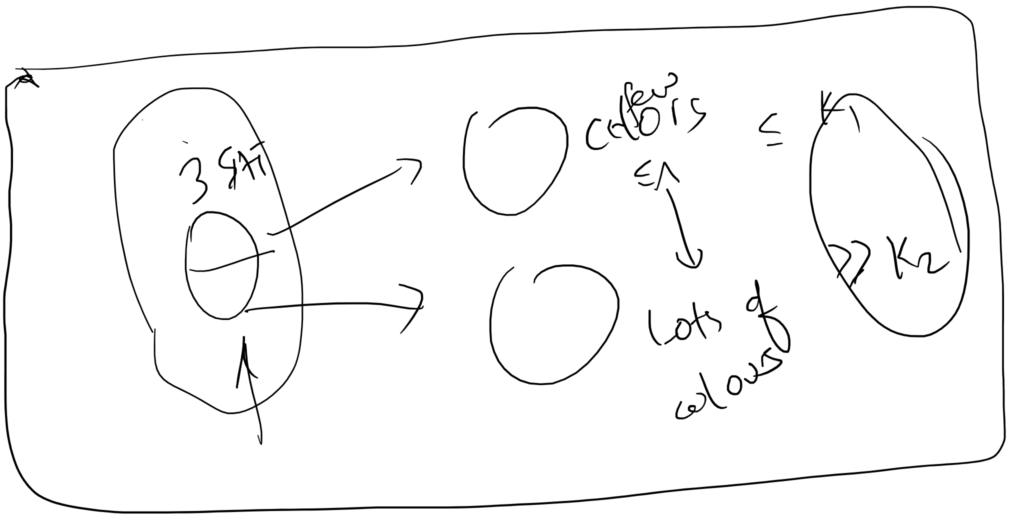
$\longrightarrow$ ✓

NP-complete ×

ln n
approx
(greedy)

Graph colouring (extremely hard problem)

NP hard to get $n^{1-\varepsilon}$ approx. for colouring

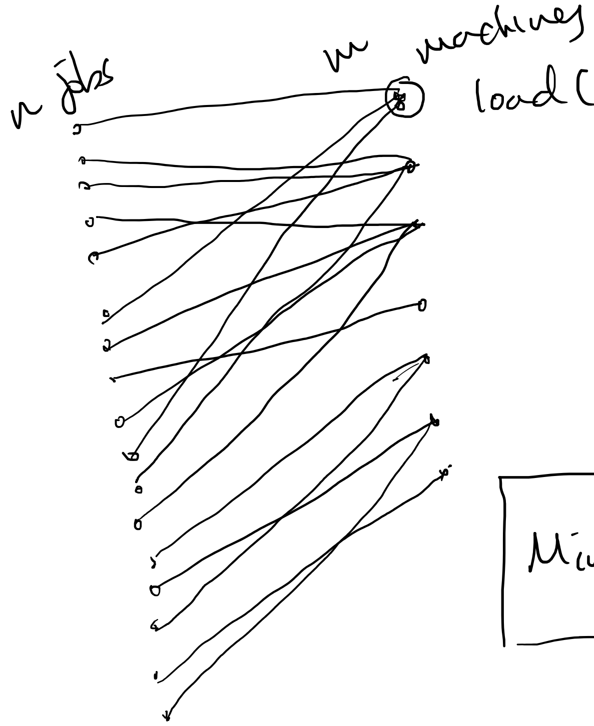


Machine Scheduling

22 February 2021 10:03

Given n jobs, each with load $p_j \geq 0$, and m machines,

goal:
assign jobs to machines to minimize the max load on any machine



$\text{load}(i) = \sum_{j \to i} p_j$

"sum load of all jobs assigned to M/C i"

$$\boxed{\min \max_i \text{load}(i)}$$

NP - complete, so seek Approximation Algorithms}.

Potential Algorithms :-

① LP formulation ?

② Greedy Algorithm ?
Sort p_i's in descending order

Assign each job to M/C with least current load.

③ Algo ②, w/o sorting up front.

{ "Online Algorithm" }

{ Decisions are done as and when input is revealed to us }

Good News :-

Even this simple online Algorithm is a 2-approximation algo for the problem.

[Problem is called "Makespan Problem" in scheduling theory.]

Study of Approx Algos

① Need Algo

② Need understanding of OPT for analysis.

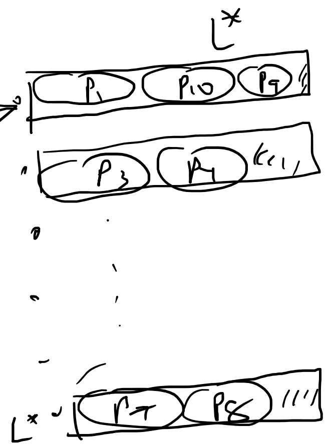
In our case, 2 simple facts give us this understanding.

Sps L^* = optimal Makespan,

① $\boxed{L^* \geq \max_j p_j}$

② $\boxed{L^* \geq \dfrac{\sum_j p_j}{m}}$

$$\sum_j p_j \leq m \cdot L^* \Leftarrow \begin{cases} p_1 + p_{10} + p_9 \leq L^* \\ p_3 + p_4 \leq L^* \\ \vdots \\ p_7 + p_8 \leq L^* \end{cases}$$



ANALYSIS of greedy :-

Let L_j denote the load on the M/C to which we assigned job j, <u>at the time we assigned job j</u>.

time we assigned job j.

Load of this M/C after we assigned job $= L_j + P_j$

At the time job j was considered, all M/C^s have a load of at least L_j !

$$\sum_{\text{all } j} P_j \geq \sum_{j' \text{ before } j} P_{j'} \geq m L_j$$

$\Rightarrow \quad m L_j \leq m L^*$ (from ②)

$\Rightarrow \quad \boxed{L_j \leq L^*}$

$\Rightarrow$ load of this M/C after we assign the job is

$= L_j + P_j$

$\leq L^* + P_j$

$\leq 2L^*$ (from ①).

apply this to all jobs to infer that

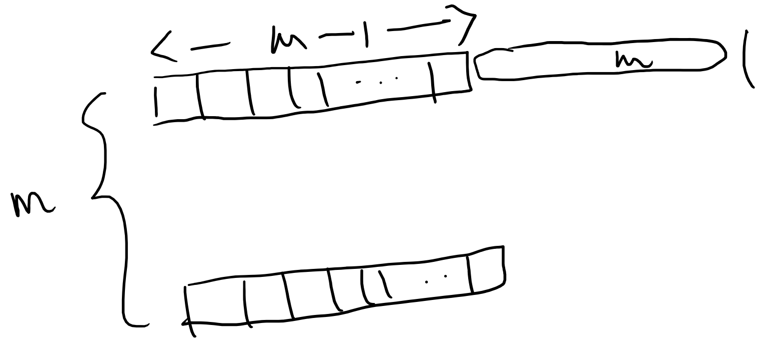
load on all machines is $\leq 2L^*$, giving us the desired 2-approximation ☒

Q_1) Can this algo have a better analysis?

EXAMPLE:

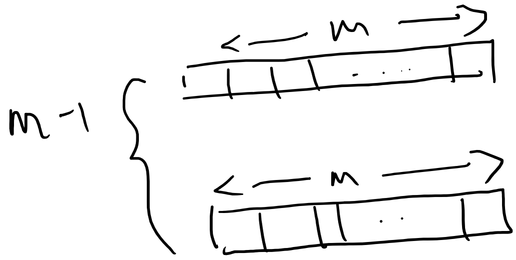
There are $m(m-1)$ jobs of size 1, and 1 job of size m.

Sps we process all small jobs before the large job, what will greedy algo do?



Greedy Makespan $= m-1+m$
$= 2m-1$

Whereas Opt



m

Optimal Makespan $= m = L^*$

factor: $\frac{2m-1}{m} = 2 - \frac{1}{m} \approx 2$ ☹

Point raised

Bad Example is somewhat cheating because Algo works in Online model but OPT reserved 1 M/C for large job.

(Crucially was offline nature of input).

fair, but the example was just to show that we couldn't have done a better analysis for this algo in the offline problem. □

You could try Algo 2 to see

If that does better.

↑ THINK / READ UP about what analysis this can give

Instead, we'll present a PTAS

↓ It will take long time to run, but will compute $(1+\varepsilon) \cdot$ OPT solution.

Ideas like for Knapsack

1. Discretization / Rounding
2. Enumeration (brute force "Small instances")
3. Guessing framework.
4. Accommodate greedy Algorithm.

New ideas

<u>Idea</u> :-

Divide jobs into 2 categories

If $p_j \le \varepsilon L^*$, j is "small"

$p_j > \varepsilon L^*$, j is "large"

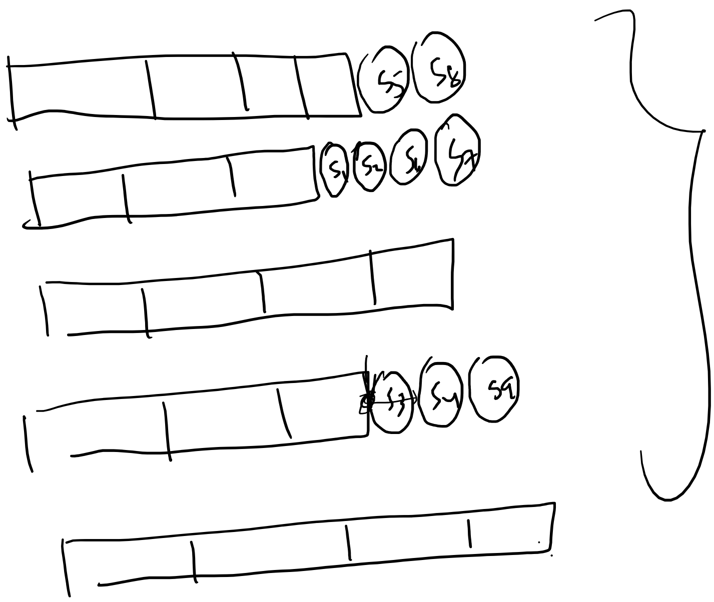
Note: We assume that we "know" the value of L^*. We'll assume for now & get rid of assumptions at the end.

Guessing Framework

Idea:

If we run greedy Algo only on small jobs, then its analysis will be very good.

Use "enumeration + rounding" to find good assignment of large jobs.



Assignment of large jobs.

If large assignment is good, meaning all M/Cs have load $\leq L^*$, then

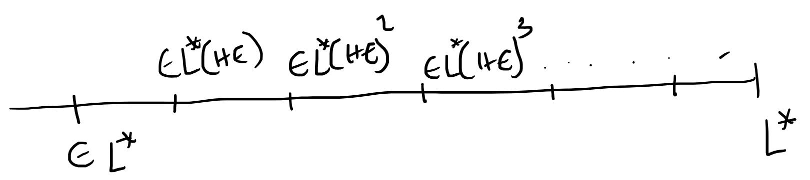
Overall algo is good, all M/Cs have makespan $\leq (1+\varepsilon)L^*$.

SAME ANALYSIS AS BEFORE.

It remains to find a good assignment of the large jobs alone.

23/02/2021.

How do we handle the large jobs?



Large jobs are jobs whose processing times range between ϵL^* and L^*.

Say a job belongs to class i if its $p_j \in [\epsilon L^*(1+\epsilon)^{i-1}, \epsilon L^*(1+\epsilon)^i)$

Moreover, "roundup" its processing time to

$$\hat{p}_j = \epsilon L^* (1+\epsilon)^i$$

Q1: How many non-empty classes are there?

The "last" class i^* is smallest value satisfying $\epsilon L^* (1+\epsilon)^{i^*} \geq L^*$

$$(1+\epsilon)^{i^*} \geq \frac{1}{\epsilon}$$

$$i^* \log(1+\epsilon) \geq \log\left(\frac{1}{\epsilon}\right)$$

Morally,

using $\boxed{\log(1+\epsilon) \approx \epsilon \text{ for small enough } \epsilon}$ TAYLOR SERIES

$$i^* \cdot \epsilon \geq \log 1/\epsilon$$

$$\boxed{i^* \approx 1/\epsilon \log 1/\epsilon}$$

$\Rightarrow$ There are only $K = \frac{1}{\epsilon} \log \frac{1}{\epsilon}$ many classes to consider!

(constantly many types).

OPT of "rounded instance $\hat{I}$" (only comprising large jobs w/ $\hat{p}_j$ values)
$\leq (1+\varepsilon) L^*$.

$\therefore$ $\boxed{\forall j : 1 \leq \frac{\hat{p}_j}{p_j} \leq (1+\epsilon)}$

$\Rightarrow$ If we find a good sol^n for $\hat{I}$ of makespan $\leq (1+\epsilon) C^*$, then we are done.

<u>Idea</u>:

There are only constantly many job types and moreover, each M/c can accept $\leq$ constantly many jobs in the optimal sol^n.

Highly Structured Instance !!

We'll claim that there are only polynomially many types of schedules of the above structure, so we can enumerate over them and pick the best.

Let $K = \frac{1}{\epsilon}\log\frac{1}{\epsilon}$ (# Interesting Classes)

$L = \frac{1}{\epsilon}$ (Max # jobs any M/c gets in OPT soln),

b/c all jobs $p_j \geq \epsilon L^*$.

Let n_ℓ be the # jobs of class ℓ

Each Machine i receives $n_\ell(i)$ many jobs of class ℓ

s.t. $\sum_{i=1}^{m} n_\ell(i) = n_\ell$

In any schedule

distinct "configurations" or distinct assignments any M/c can get?

Each $0 \leq n_\ell(i) \leq L$ (L+1 choices) for all classes

$l = 0, 1, \ldots K$

$\Rightarrow$ # possible assignments any m/c can receive is

$$\le (L+1)^{(K+1)} = \text{constant}$$

$C_1 \quad C_2, \ldots \ldots \quad C_M$

be these distinct configurations

$\Rightarrow$ Really we're asking

How many m/c's get config C_1,
config C_2,
⋮
config C_M?

enumerate over all such allocations and pick the best sol^n.
which is feasible
(ie) for all l,

$$\sum n_l(i) = n_l.$$

---

enumerations $\le$

to go over

m balls for m/cs

M-1 sticks for the configs

$$\binom{m+M-1}{M-1}$$

$$\le m^{M-1}$$

$$\le m^{\text{constant}} \quad \left[M \text{ b/c is a constant}\right]$$

If we look closely: $m^{(1/\epsilon)^{1/\epsilon}}$, still polytime for constant ϵ !!

Overall Algo:

1. Guess L^*
2. Enumerate all types of configs among m machines
3. Pick best feasible one
4. Greedily Allocate small jobs

gives $(1+\epsilon)L^*$ Makespan.

L

(4) Greedily Allocate small jobs

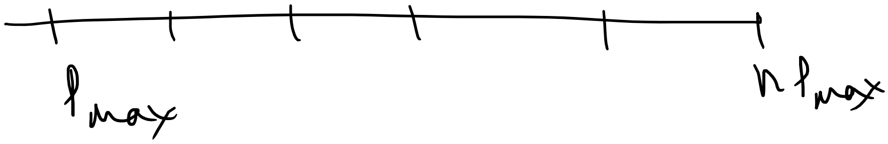
RUNTIME: $\boxed{poly(n)}$.

NOTE

More interesting from theoretical perspective, nobody is going to use this as is in practice.

For guessing L^*, note that

$L^* \geq P_{max}$ & $L^* \leq n \cdot P_{max}$



Say L^* belongs to class j, if

$$L^* \in \left[P_{max}(1+\epsilon)^{j-1},\ P_{max}(1+\epsilon)^{j}\right)$$

Need to consider $\leq \frac{1}{\epsilon}\log n$ many classes for L^*.

Try running above algo for all "classes" of L^* (ie) $P_{max}(1+\epsilon)^{j}$ for all j, and choose the smallest which works.

Overall our makespan would be

$(1+\epsilon)^2 L^*$

↑

One $(1+\epsilon)$ comes from approx. guess for L^*

another comes from Algo.

Set ϵ' to $\epsilon/3$ and run w/ ϵ'

to get overall

$(1+\epsilon')^2 \cdot L^* \leq (1+\epsilon) \cdot L^*$ approximation

∎

What to do TOMORROW if P_j (load of job) is Machine dependent

↑

actually, it's P_{ij}. (Non-Identical Machines)

aka Unrelated Machines in scheduling literature

Unrelated Machines

24 February 2021 07:57

- n jobs, m machines
- job j has a "load" p_{ij} on machine i

  Note that p_{ij} can be different from $p_{i'j}$.

Q: WHY study this extension?

A: - Machines can be human resources.
- " " " non-identical (ex. CPU, GPU, RAM, etc)

ex. If some M/C has too little RAM to accommodate a job, we can set $p_{ij} = \infty$ for i.

Can also capture speeds S_i on M/C i.

In that case we can think of

$$p_{ij} = \frac{p_j}{S_i}$$

Related Machines Scheduling

easier than unrelated Machines, PTAS is known for this problem as well

64 65

Brainstorming

① Candidate 1: Greedy,

- Process jobs in arbitrary order,
- Insert j into M/c i with lowest resulting load.

cur-load (i) + P_{ij}

- Lets think for a bit

Here is an interesting "bad example" for this algo

Machines $m = 2^k$

M_1 M_2 . . . . . . M_{2^k}

job j_1 has unit load on M_1 & M_2, ∞ on rest.

job j_2 " " " on M_3 & M_4 ∞ on rest.

⋮

job $j_{2^{k-1}}$ has unit load on M_{2^k-1} & M_{2^k} ∞ on rest

Any "online Algo" (in particular greedy) has this "type" of lower bound.

We may assume greedy did the following:

job → Machine

1 2

	Job	→	Machine
Jobs of Type 1	1		2
	2		4
	3		6
	⋮		
	2^{k-1}		2^k

Recurse on these M/Cs greedy used.

Jobs of Type 2

1 can go on M/c 2 & 4
∞ on rest.

2 " " " " 6 & 8
∞ on rest

2^{k-2} " " " " $2^k - 2$ & 2^k
∞ on rest.

Makespan of Greedy After Type 2 jobs :

Odd M/c have 0 load.
1/2 of even M/c have 1 load
1/2 of even M/c have 2 load

If we had just this instance, what would OPT look like ?

Type 1 jobs can go to odd M/cs,
All M/cs can have load 1.

To get worse example, recurse on even M/cs.

lets say greedy sent type 2 job to 4th machine.

Type 3 jobs only go on these ☹

$$\begin{cases} 1 \to 4 \text{ or } 8, & \infty \text{ elsewhere} \\ 2 \to 12 \text{ or } 16, & \infty \text{ elsewhere} \\ \vdots & \\ 2^{k-3} \to 2^{k-1} \text{ or } 2^{k} & \infty \text{ elsewhere.} \end{cases}$$

By continually focussing only on Machines of high load, we are forcing greedy to keep increasing its load by 1 in each type/phase.

By the end of $k = \log m$ rounds,
Greedy Makespan $= k$
Opt Makespan $= 1$

Lower bound

lower bound
for any
(deterministic) ONLINE ALGORITHM
which takes actions
immediately upon
seeing a new
job.

Informal note - Needs to be more precise

Could try to think about randomized algorithm
{ if things look equal, pick randomly }

THIS is still a lower bound for RND Algorithms if the Nature of type of jobs can depend on Algos choices.

Not "as stated" a lower bound if input jobs is independent of Algos randomness.

Present an LP - based algorithm :-

by guessing to within $(1+\varepsilon)$, lets assume that we know L^* (optimal Makespan)

x_{ij} : denotes if job j is assigned to machine i.

What are the legitimate constraints we can enforce on $\{x_{ij}\}$ variables !

Min 0 (Interestingly, no obj. fn)

$$\begin{cases} ① & \sum_i x_{ij} = 1 \quad \forall j \text{ (Job assignment constraint)} \\ ② & \sum_j x_{ij} p_{ij} \le L^* \quad \forall i \text{ (machine Load constraints)} \\ & x_{ij} \geq 0 \end{cases}$$

From ①, all x_{ij}'s will also never exceed 1.

Lets say we solve this LP & it is feasible.

How do we find an integral solution from it?

Q: what if we think of L^* as a variable and try to minimize L^*?

(With hindsight, we don't follow this approach.

Really, LP is a "feasibility LP" and we are trying to exploit structure of BFS/Vertex/Extreme Points.

Idea ①

Randomized Rounding

- each job chooses its M/C according to $\{x_{ij}\}$ as a distribution.

↓

Can show $O(\log m)$ - approximation using Chernoff bounds.

We are equipped to study this algorithm!

Idea ②: Exploit structure of BFS/Extreme pts.

↓

- mn dimensional space,
- BFS / Vertices are identified by the intersection of mn hyperplanes.

$\Rightarrow$ mn constraints are satisfied at equality

$\boxed{\Rightarrow \text{at least } mn-(m+n) \text{ of the } x_{ij}\text{'s are actually } 0}$

$\Rightarrow$ At most $(m+n)$ variables are non-zero in the LP.

Lets ~~think~~ visualize the sol^n like a graph

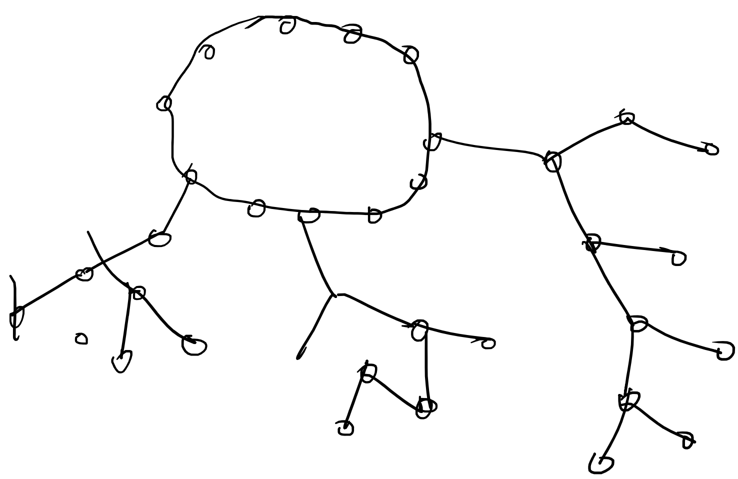
Vertex for every job & machine
Edge (i,j) if $x_{ij} > 0$

vertices : $m+n$
edges : $\leq m+n$

The most complicated soln to deal with is when # edges $= m+n$.

Fewer Non Zero variables ⇒ LP gives greater clarity.

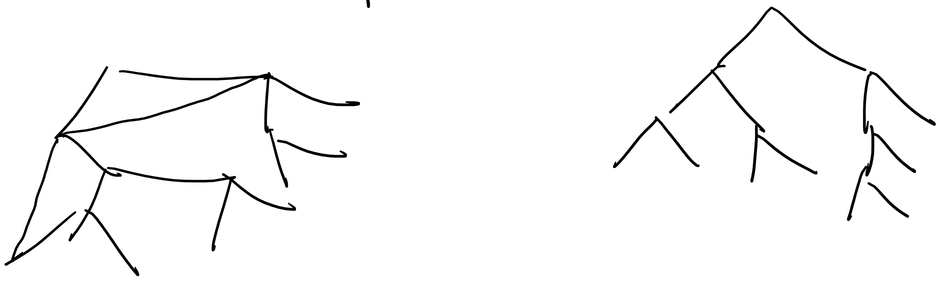
Example 1: This LP-soln-Graph looks like a tree + one cycle.



Example 2 If graph with $m+n$

vertices & m+n edges;

> 1 components



m_1, n_1 $\quad\quad$ $m-m_1$, $n-n_1$

2 cycles $\quad\quad$ tree

We'll argue that
BFS/ extreme points
can't have
this structure!

<u>THM</u>:

In any BFS, it will look like
every component is
a tree with ≤ 1 cycle.

<u>Recap</u>: 26/02/2021

Min Makespan on Unrelated Machines
n jobs $\quad\quad$ m machines
Job j $\longrightarrow$ Machine i (p_{ij} load).

Assume we know OPT Makespan L^*
(i.e. can "query" in powers of

Assume we know OPT Makespan L^*

(we can "guess" in powers of $(1+\varepsilon)$)

$$\min \quad 0$$

(L^* is not a variable)

$$\left\{ \begin{array}{lll} \sum_i x_{ij} = 1 & \forall j \\ \sum_j p_{ij} x_{ij} \leq L^* & \forall i \\ x_{ij} \geq 0 & \end{array} \right\} \quad LP(M, J, L^*)$$

LP over Machine set M, Job set J & Makespan L^*

We solve this LP ~~optimally~~ and find any basic feasible sol^n / Extreme point / Vertex sol^n.

Let x^* denote such a sol^n.

Q: How many Non-Zero Variables can x^* have?

A: $\leq m+n$. (because x^* is BFS, it is a intersection of mn hyperplanes $\Rightarrow \geq mn-(m+n)$ must be of the form $x_{ij} = 0$

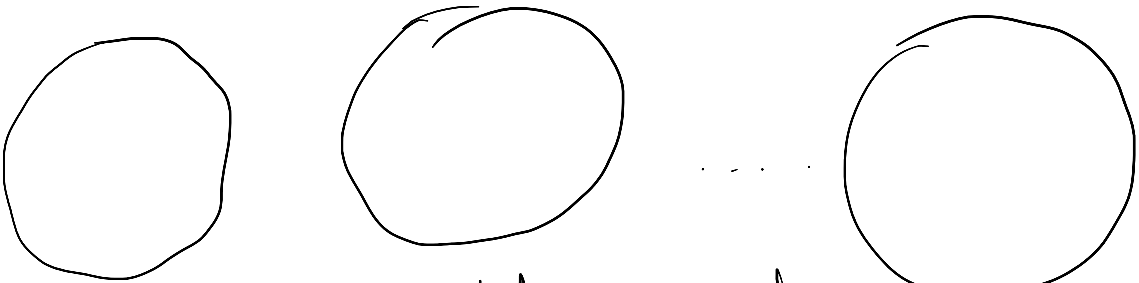
$x_{ij} = 0$

$\Rightarrow$ $\leq (m+n)$ x_{ij}'s can be > 0.

ONE MORE OBSERVATION:

Let's construct a graph with vertices $= M \cup J$ and edges for non-zero variables

$m+n$ vertices, $\leq m+n$ edges



Sps it has k - connected components

CLAIM ②

In Graph(x^*), each connected component is a "Pseudo Tree"

Meaning a tree with at most one extra edge.

Proof :-

Suppose some connected component is not a pseudo-tree.

Suppose some connected component
is not a pseudo-tree.

- Sps it has $\bar{M} \subseteq M$ as Machine set &

$\bar{J} \subseteq J$ as job set.

Sps $|\bar{M}| = \bar{m}$ & $|\bar{J}| = \bar{n}$.

by connectivity # edges in component $\geq \bar{n} + \bar{m} - 1$

Now, b/c it is not a pseudo-tree,

edges in component

$\geq \bar{m} + \bar{n} + 1$.

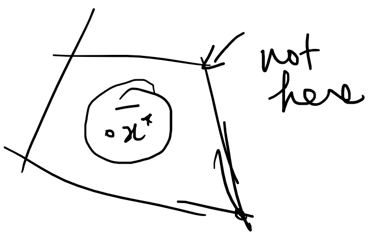
Let $\bar{x}^*$ denote the restriction of x^* to $\bar{J}$, $\bar{M}$.

We know that

$\bar{x}^*$ is feasible LP solution to $LP(\bar{M}, \bar{J}, L^*)$.

But it is not a BFS for this restricted LP.

Any BFS for restricted LP must have $\leq \bar{m} + \bar{n}$ Non-zero Variables !!



$\bar{x}^*$ is feasible for restricted $LP(\bar{M}, \bar{J}, L^*)$

but not BFS / Extreme Point
since it has more non-zero
variables than
BFS's can have for
the restricted polytope.

$\Rightarrow$ can write $\bar{x}^* = \alpha \bar{x}_1 + (1-\alpha) \bar{x}_2$

where $\bar{x}_1$ & $\bar{x}_2$ are
feasible for
$LP(\bar{M}, \bar{J}, L^*)$

because $\bar{x}^*$ is
not a BFS,
it is in interior
of polytope

$\Rightarrow$ Can write x^* as a
convex combination of
x_1 and x_2

for $LP(M, J, L^*)$

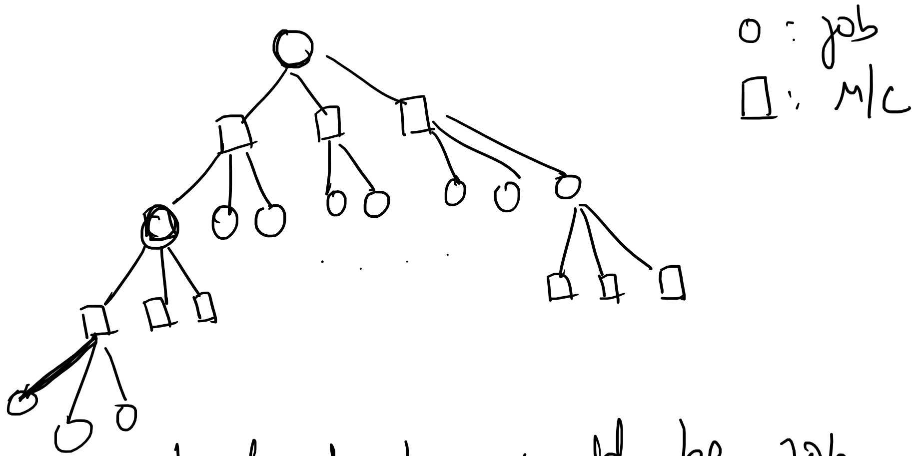
$x_1 = \bar{x}_1$, rest of x^*
$x_2 = \bar{x}_2$, rest of x^*

$\Rightarrow$ x^* can't be a BFS
since we're able to write
x^* as convex combination
of 2 feasible

of 2 feasible
solutions.

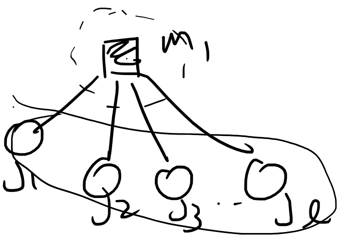
SUMMARY
(1) x^* is a BFS
(2) $\Rightarrow x^*$ restricted to all components have to be BFS for restricted problem
$\Rightarrow \overline{x^*}$ for each component is a pseudo-tree

LAST STEP:
→ Rounding pseudo-trees.
→ Let's see how to round trees, pseudo-trees will be similar in spirit.

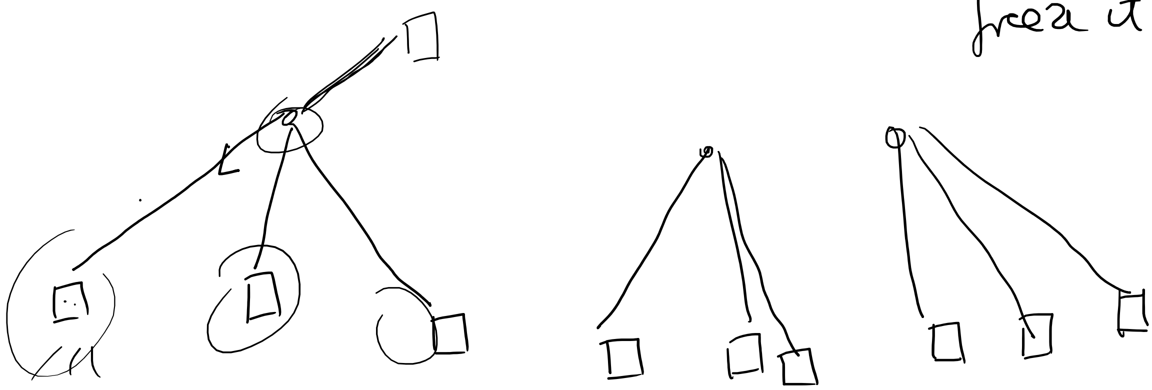


leaf of tree could be job or Machine

leaf is a job ⟹ $x_{ij} = 1$ for that job, so we can freeze that assignment



all must be $= 1$, so we can freeze it.



Now all leaves are Machines.

Simply assign the parent job to an arbitrary child M/C.

contributes to an "excess" load on the child M/C

$\leq \max_{i,j} p_{ij}$

CAN WORK UPWARD for the rounding

Can round x^* using pseudotree structure to assign jobs → M/Cs

M/Cs

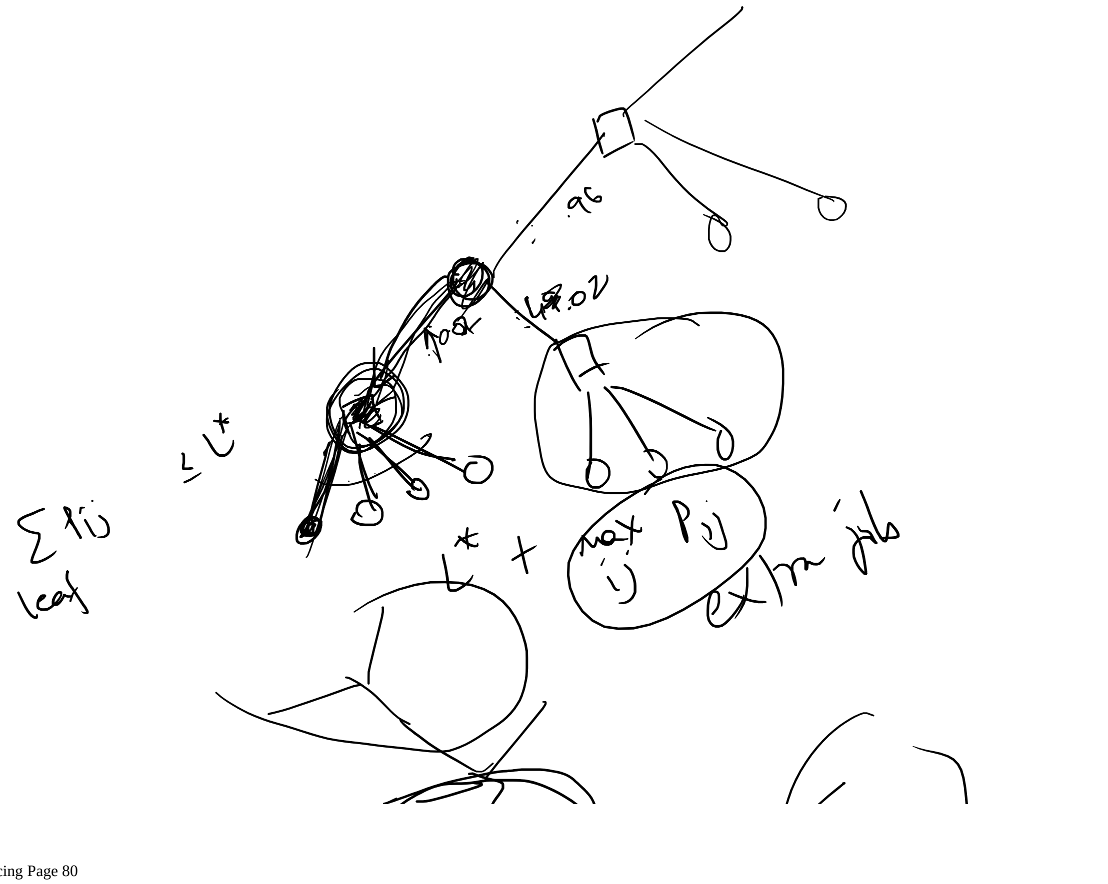
such that each M/C gets a load $\leq L^* + \max_{ij} p_{ij}$

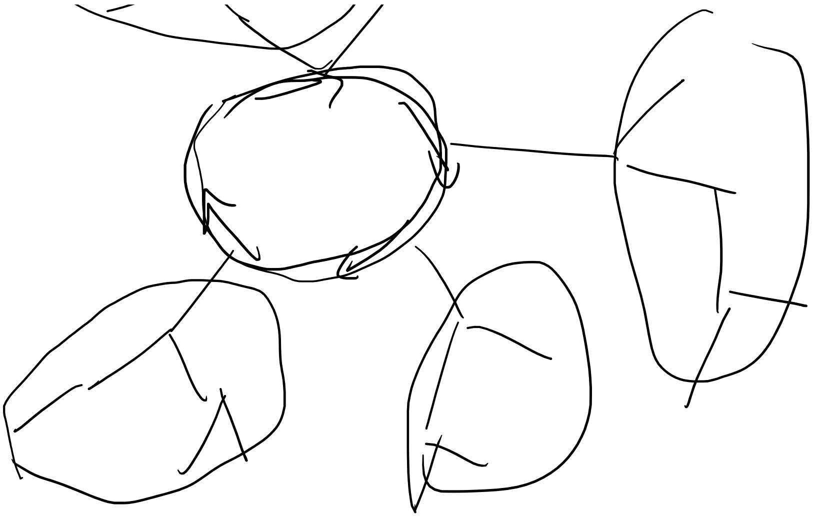
$\Rightarrow$ LAST STEP

Since we know L^*, we can for $x_{ij} = 0$ if $p_{ij} > L^*$ in the LP

(either explicit constraints or just delete these variables for large p_{ij} from original LP).

$\Rightarrow$ Gives 2-approximation to Makespan on "Unrelated M/C"





Unrelated Machines Scheduling

01 March 2021 10:04

Min Makespan Scheduling

- n Jobs, m M/C
- p_{ij} = load of job j on M/C i.

Find assignment of jobs to machines to minimize maximum load on any machine

$$\boxed{\text{Min}\ \max_i \sum_{j \to i} p_{ij}}$$

Technique is "Iterative LP Rounding"

<u>High level idea</u> :-

Sps there is an LP with n variables, with

$\rightarrow$ $0 \le x_i \le 1$ type of constraints

and $\rightarrow$ 'm' other linear constraints

<u>Totally</u>

$\text{Max}\ c^T x$

$\rightarrow$ $Ax \le b$ $\leftarrow$ m constraints

$0 \le x_i \le 1$ $\leftarrow$ n variables

If 'm' is very small, $\ll n$, then many variables will be satisfied Integrally.

in a BASIC FEASIBLE SOLUTION.

<u>Proof</u>: In a BFS, it is defined by the intersection of 'n' constraints satisfied @ equality

At most 'm' of these can come from $Ax \le b$

$\Rightarrow$ $\geq n-m$ variables are satisfied integrally.

Idea of Iterative Rounding is to reduce the 'm' value as much as possible, by removing constraints which can be handled through other means

little vague, but we'll see soon

BACK TO SCHEDULING :-

What's the LP?

① We "guess" Optimal Makespan L^*

$$\text{Min } 0$$

$$\rightarrow \sum_j p_{ij} x_{ij} \leq L^* \quad \forall i$$

$$\rightarrow \sum_i x_{ij} = 1 \quad \forall j$$

$$0 \leq x_{ij} \leq 1 \quad \forall (i,j)$$

Moreover, delete x_{ij} variables with $p_{ij} > L^*$.

Lem ①
If guess of L^* is correct, then LP is feasible.

More general LP (allowing different loads on each machine).

$$\text{Min } 0$$

$$\sum_j p_{ij} x_{ij} \leq L_i \quad \forall i \in M$$

$$\sum_i x_{ij} = 1 \quad \forall j \in J$$

$$0 \le x_{ij} \le 1 \quad \forall (i,j) \in E$$

set of allowed variables

Let $\bar{L} = (L_1, L_2, \ldots L_m)$, initially $E = \{(i,j) \text{ st } p_{ij} \le L^*\}$

$\boxed{\text{Then } LP(J, M, \bar{L}) \text{ is feasible for } \bar{L} = (L^*, L^*, \ldots L^*) \text{ for correct guess of } L^*.}$

ALGORITHM

1. Guess L^* value
2. While $(J \neq \phi)$
3. Solve $LP(J, M, E, (L^*, L^*, \ldots, L^*))$ and compute a BFS - x^*
4. If $\exists\, j \in J$ and $i \in M$ st $x^*_{ij} = 1$,
   - 4a assign $j \to i$ in our schedule
   - 4b Reduce $L_i = L_i - p_{ij}$
   - 4c Remove j from J, update E appropriately
   - 4d Go back to Step 2
5. If some $x^*_{ij} = 0$, remove this variable from E and go back to Step 2
6. If some machine $i \in M$ has only one job j with $x^*_{ij} > 0$

   Assign job $j \to$ Machine i.

   Remove j from J

   i from I

   and update E accordingly.

Goto Step ② ⟸ Can increase load on i beyond its capacity but "its a simple constraint"

⑦ if some M/C $i \in M$ has only <u>2 jobs</u> j_1 & j_2 with $x^*_{ij_1} > 0$, $x^*_{ij_2} > 0$ and $x^*_{ij_1} + x^*_{ij_2} \geq 1$

then assign both jobs to M/C i,
remove them from J
remove i from M.
update E accordingly.
Goto step ②

Job ① has $p_{1i} = 0.9L^*$, $x^*_{1i} = 1$

↓

freeze job 1 → M/C i
update $L_i = L^* - 0.9L^*$
$= 0.1L^*$

Next Step of Algo,
Job ② has $P_{2i} = L^*$, $x^*_{2i} = 0.1$
but this is only job going to M/C i.

Algo assigns job 2 to M/C i.

Is ALGO CLEAR ?

Not yet, because it can potentially cycle in ∞-loop.

↓

~~jee~~

Need to show

At any iteration, if x^* is a BFS, one of the conditions 4, 5, 6, 7 MUST HOLD !!

Proof :- Suppose x^* is a BFS where none of these conditions hold.

① Every job $j \in J$ has at least 2 non-zero x^*_{ij} variables

Proof: $\sum_i x^*_{ij} = 1$ in total, but NO single variable is 1.

② Every machine $i \in M$ has at least 2 incoming jobs with > 0 x^*_{ij} value.

Pf Otherwise it satisfies condition 6.

$\Rightarrow$ Total # of strictly positive variables in the LP?

If the # of jobs is $|J|$ and # machines is $|M|$, then

strictly +ve variables is $\geq |J| + |M|$

But x^* is a BFS to

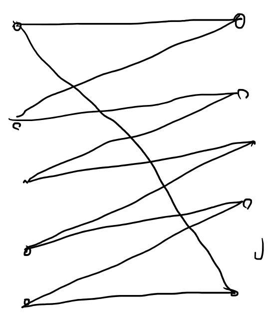
$$\sum x_{ij} p_{ij} \le L_i \quad \forall i \in M$$
$$\sum x_{ij} = 1 \quad \forall j \in J$$
$$0 \le x_{ij} \le 1 \quad \forall (ij) \in E$$

In any BFS At least $|E| - |M| - |N|$ variables must be 0 or 1.

This says that $|E|$ has to $= |M| + |N|$.

$\Rightarrow$ If LP soln does not satisfy conditions 4, 5 or 6, it must be very structured

(i.e. a) Exactly $|M| + |N|$ variables in LP,
b) Each job has 2 edges
c) Each M/C has 2 edges



Has to look like a disjoint collection of cycles.

$\Rightarrow$ at least 1 M/C satisfies

condition 7 & SO
we make progress
(CONTRADICTION).

02/03/2021

J : set of jobs

M : set of Machines

E = { edges, valid variables }

Initially $J = [n]$

$M = [m]$

$E = \{ (i,j) : p_{ij} \le L^* \}$

L_i = target residual load on M/C i

Initially, all $L_i = L^*$.

Min 0

$$\sum_{j : (i,j) \in E} p_{ij} x_{ij} \le L_i \quad \forall i$$

$$\sum_{i : (i,j) \in E} x_{ij} = 1 \quad \forall j$$

$$0 \le x_{ij} \le 1 \quad \forall (i,j) \in E$$

Algo

1) Guess L^*

2) While $J \neq \phi$

Solve LP(J, M, E, (..., ...)) to

(2) While $J \neq \phi$

(3) Solve $LP(J, M, E, (L_1, L_2, \ldots L_m))$ to get BFS x^*

(4) if $\exists$ (i,j) st $x^*_{ij} = 1$

↳ Update $L_i = L_i - P_{ij}$

↳ Assign job j to M/c i

↳ Remove job j from J and E

↳ Goto step (2)

(5) if $\exists$ (i,j) st $x^*_{ij} = 0$

↳ Remove (i,j) from E

↳ Goto step (2)

(6) if $\exists$ M/C i with exactly <u>or NONE</u> one incoming job, (ie) $x^*_{ij} > 0$,

↳ Assign job $j \to$ M/C i

↳ Remove i & j from M & J and update E

↳ Goto step (2)

7. If $\exists$ M/c i with exactly 2 +ve x^*_{ij} and

$x^*_{ij_1} + x^*_{ij_2} \geq 1$

↳ Assign both j_1 & j_2 to i

↳ Remove i, j_1, j_2 from M and J resp'ly

↳ Goto Step 2.

<u>Lemma 1</u>

- Algo doesn't get stuck in an ∞ while loop

If (i, c) x^* is a BFS, then one of 4, 5, 6 or 7 holds true.

<u>Proof</u>:

We show If 4, 5, and 6 don't hold for x^*, then 7 must hold.

Firstly, we claim that if 4 & 5 don't hold, then set of allowed/active variables in the LP is very small.

IP = Min T

LP = Min 0

$$\begin{cases} \sum p_{ij} x_{ij} \le L_i & \updownarrow m \\ \sum x_{ij} = 1 & \updownarrow n \end{cases}$$

$$0 \le x_{ij} \le 1 \quad \forall (i,j) \in E$$

In any BFS x^*, at most $|M| + |N|$ variables can be truly fractional, i.e.)

$$0 < x^*_{(ij)} < 1 .$$

Pf

We are looking for solutions in $|E|$-dim space

· These vertex pts of LP are intersections of $|E|$ tight constraints

$\Rightarrow \ge |E| - (m+n)$ must come from $0 \le x^*_{ij}$ & $x^*_{ij} \le 1$

But because (4) & (5) don't hold

$$|E| - (m+n) \le 0$$

$$\Rightarrow |E| \le (m+n) .$$

(**)

Next,

Because each job $j \in J$ (current set of active jobs)

(1) has $\sum_i x^*_{ij} = 1$ but no single variable is 1,

it has at least 2 active edges with $x^*_{ij} > 0$.

(2) Similarly, each M/C $i \in M$ (current set of active M/Cs)

has at least 2 active edges with $x^*_{ij} > 0$

[B/C step (6) also didn't occur]

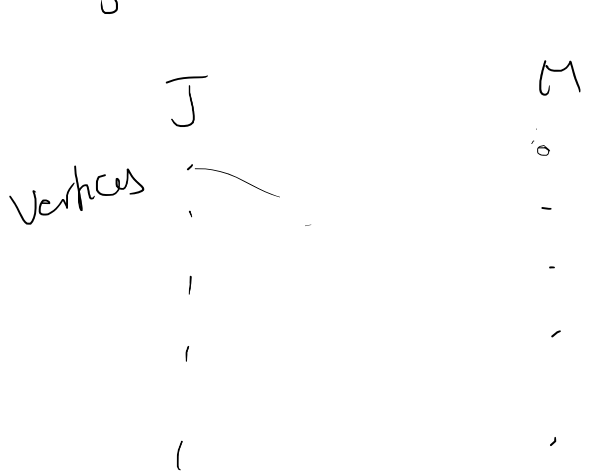
$\Rightarrow$ Total # of variables $\geq 2|J|$

Total # of variables $\geq 2|M|$

(*) $\Rightarrow$ By averaging, $|E| \geq |J| + |M|$.

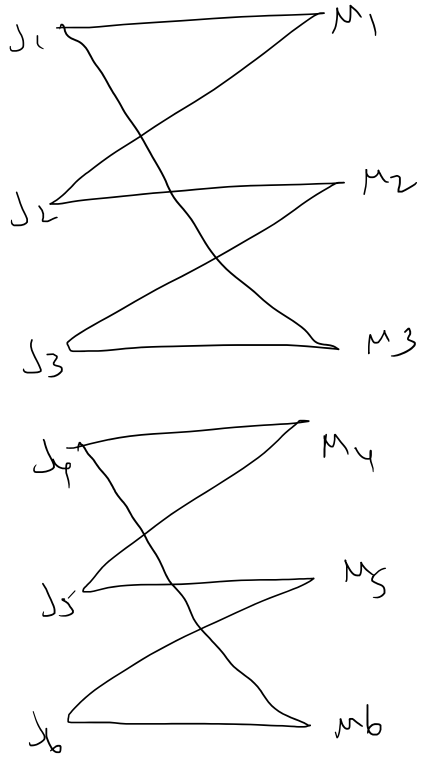
From (*) and (**) we get $|E| = |J| + |M|$

Next imagine a graph



This graph has # edges = # vertices
and every vertex has
degree = 2.

$\Rightarrow$ this graph has to be a UNION of
cycles.



From this, how do we conclude that
step 7 must happen?

In any cycle, $\sum_{i,j \in \text{cycle}} x^*_{ij}$ = # jobs in cycle

= # M/Cs in cycle.

$\Rightarrow \exists$ some M/C i with $\sum x^*_{ij} \geq 1$.

$\Rightarrow$ Step 7 holds !! $\boxtimes$

We find a good solution eventually.
(ie) it has Makespan $\leq 2L^*$

If i is an active Machine,
L_i + Current load assigned to i
$\leq L^*$

By Induction
- Initially true since Current load = 0.
- load only increases Step 4 or 6 or 7

Property continues to hold true in Step 4

- Need not hold after Step 6 or 7, but M/C i is no longer active after these!

If we get to Step 6,
we may increase load of i by 1 jobs which has $p_{ij} \leq L^*$
$\Rightarrow$ OK.

If we get to step (7),

we may increase load of i by 2 jobs

$\Rightarrow$ $3L^*$ guarantee is trivial

but we can do better by using $x^*_{ij_1} + x^*_{ij_2} \geq 1$

In our soln, both x^*_{ij} are rounded to 1

So total increase, when compared to L_i (which is excess load on $M(i)$)

$$= (1 - x^*_{ij_1}) p_{ij_1} + (1 - x^*_{ij_2}) p_{ij_2}$$

$$\leq (1 - x^*_{ij_1}) L^* + (1 - x^*_{ij_2}) L^*$$

$$\leq (2 - (x^*_{ij_1} + x^*_{ij_2})) L^*$$

$$\leq L^*$$

New Load to i $- L_i \leq p_{ij_1} + p_{ij_2} - x^*_{ij_1} p_{ij_1} - x^*_{ij_2} p_{ij_2}$

$$\leq L^*$$

$\Rightarrow$ New load to i $\leq L_i + L^*$

Existing load + New load $\leq$ Existing load + $L_i + L^*$

$\leq 2L^*$ ∎

MAIN TAKEAWAY

1) LP BFS are powerful (almost integral many times)

2) If there is simple constraint blocking a BFS from becoming more integral, we can remove it hope to handle it in other ways

ITERATIVE ROUNDING AND RELAXATION

Removing constraints

Same scheduling problem, different Algorithm

n jobs $\longrightarrow$ m Machines
Job j has load P_{ij} on m/c i.
Find an assignment to Min Max Load.

We saw that LP BFS can be used to
- get 2 approximation (2 different methods)

- Today: "Simpler" "more efficient" "worse guarantee"
algorithm $O(\log m)$ - approximation.

- Nice feature: can assign jobs one by one in arbitrary order without knowledge of future jobs.

Idea:-
Running "greedy"-type of algorithms
to $\boxed{\text{minimize } \max(\text{load}(1), \text{load}(2), \ldots \text{load}(m))}$

Not a very smooth or easy-to-understand function.

We will actually try to optimize a "different smoother objective function" which is close enough to the Max-Objective Function.

$$f(l_1, l_2, \ldots l_m) = l_1^p + l_2^p + \cdots + l_m^p$$
for large enough P.

"Intuition:" If p is larger and larger, then the large values of $\bar{l} = (l_1, .. l_d)$ start dominating the expression.

(ie) if $p >>$ large,

$$(\text{max load})^p \leq f(l_1, l_2, .. l_m) \leq m \cdot (\text{maxload})^p$$

Next lecture

Greedy Algo for scheduling using "surrogate" p-NORM objective

---

05/03/2021 Yet another algorithm for Makespan Minimization on Unrelated Machines

Idea: Come up with a suitable "smoother approximation" of the $\max_i (\text{load}(i))$ Objective.

In particular,

Minimizing $\text{Max}(x_1, x_2, .. x_m)$ closely related to

Minimizing $\sum_{i=1}^{m} x_i^q$ for large enough q.

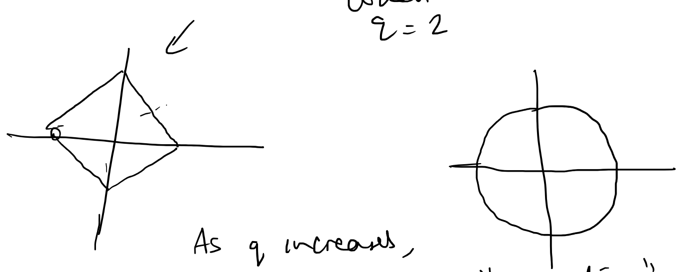
Intuition

let's say $M = 2$. and plot

$$|X|^q + |Y|^q = 1$$

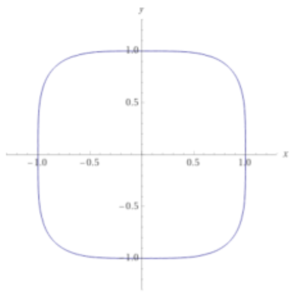
when $q = 1$

when $q = 2$



As q increases, the curve is "expanding" the curve is "expanding"

when $q=4$



As q increases x^q+y^q is dominated by $\max(x,y)^q$

We'll look to find a schedule which minimizes $\sum_{i=1}^{m} \text{load}(i)^q = \phi$

Potential Function

Sort of a Surrogate Objective

"Soft-Max Objective"

Q) What's a nice greedy algo for Soft-Max objective?

Suppose $t-1$ jobs have been scheduled ($\text{load}_{t-1}(i)$ is the current load of M/c i at this time).

$\Rightarrow$ $\boxed{\phi(t-1) = \sum_{i=1}^{m} \text{load}_{t-1}(i)^q}$

When trying to schedule t^{th} job, try to schedule it to M/c which minimizes $\Delta\phi$, (ie)

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

? ALGORITHM! Can work with

ONLINE ...

Online Sequence of jobs.

THM :- for any sequence of job insertions,
Max load of Algo $\leq \Theta(\log m)$ OPTIMAL MAX LOAD.

for $q = \Theta(\log m)$.

Says that even if we don't know all jobs ahead of time, we can do favourably against an all-knowing optimal soln.

Proof
We'll fix value of q later.
Let job insertions be numbered
$\rightarrow 1, 2, \dots, n$
and $\phi(t) = \sum_{i=1}^{m} \text{load}_t(i)$ be the potential after t insertions
- $\phi(0) = 0$.

$\text{load}_t(i)$ = load of M/C i after first t jobs have been inserted.

Recall
$\forall j = 1, 2, \dots n, \quad \forall i$
p_{ij} denotes load of job j if scheduled on M/C i.

Lets fix insertions upto time $t-1$ for $t \geq 1$ and consider the t^{th} job.
Suppose greedy Algo sent it to M/C $i(t)$.
Suppose Optimal solution of all jobs $1 \dots n$ sends job t to M/C $i^*(t)$.

... assigning the t^{th} job

After assigning the t^{th} job

$$\phi(t) - \phi(t-1) = load_t(i(t))^q - load_{t-1}(i(t))^q$$
$$= \left(load_{t-1}(i(t)) + p_{i(t),t}\right)^q - load_{t-1}(i(t))^q$$

Because we run greedy Algorithm $\longrightarrow$

$$\circledast \quad \le \left(load_{t-1}(i^*(t)) + p_{i^*(t),t}\right)^q - load_{t-1}(i^*(t))^q$$

Now, a quick cheating proof.

Let's look @ $\circledast$

$$= L_{t-1}(i^*(t))^q \left[\underbrace{\left(1 + \frac{p_{i^*(t),t}}{L_{t-1}(i^*(t))}\right)^q}_{} - 1 \right]$$

SLIGHT CHEAT

$\approx$ if $p << L_{t-1}$

$$L_{t-1}(i^*(t))^q \left[\cancel{1} + q \cdot \frac{p_{i^*(t),t}}{L_{t-1}(i^*(t))} - \cancel{1} \right]$$

$$= q \cdot Load_{t-1}(i^*(t))^{q-1} \cdot p_{i^*(t),t}$$

$$\le q \cdot Load_n(i^*(t))^{q-1} \cdot p_{i^*(t),t}$$

Also loads are increasing over time.

Back to cheating Analysis :- Let's sum up over all arrivals.

$$\phi(t) - \phi(t-1) \le q \cdot load_n(i^*(t))^{q-1} \cdot p_{i^*(t),t}$$

$\sum_{t=1}^{n} \Rightarrow$

$$\phi(n) - \phi(0) \le q \cdot \sum_{i=1}^{m} load_n(i)^{q-1} \sum_{t : i^*(t) = i} p_{i,j}$$

$$= q \sum_{i=1}^{m} load_n(i)^{q-1} \cdot load^*(i)$$

$$= q \sum_{i=1}^{m} load_n(i)^{q-1} \cdot load^*(i)$$

$$\Rightarrow \boxed{\sum_{i=1}^{m} load_n(i)^q \leq q \cdot \left(\sum_{i=1}^{m} load_n(i)^{q-1} \cdot load^*(i) \right)}$$

USE HÖLDER'S INEQUALITY.

$$\sum_i |a_i b_i| \leq \left(\sum |a_i|^x\right)^{\frac{1}{x}} \left(\sum |b_i|^y\right)^{\frac{1}{y}}$$

as long as $\frac{1}{x} + \frac{1}{y} = 1$

dual norms.

We want

$\sum load^q$, $\sum (load^*)^q$, etc

so, what x & y should we use on RHS?

$$x = \frac{q}{q-1}, \quad y = \frac{q}{1}.$$

RHS

$$\leq q \cdot \left(\sum_{i=1}^{m} \left(load_n(i)^{q-1} \right)^{\frac{q}{q-1}} \right)^{\frac{q-1}{q}} \cdot \left(\sum_{i=1}^{m} load^*(i)^q \right)^{\frac{1}{q}}$$

$$= q \left(\sum_{i=1}^{m} load_n(i)^q \right)^{\frac{q-1}{q}} \left(\sum_{i=1}^{m} load^*(i)^q \right)^{1/q}$$

Overall, get

$$\left(\sum_{i=1}^{m} load_n(i)^q \right)^{1} \leq q \cdot \left(\sum_{i=1}^{m} load_n(i)^q \right)^{\frac{q-1}{q}} \left(\sum_{i=1}^{m} load^*(i)^q \right)^{1/q}$$

$$\Rightarrow \left(\sum_{i=1}^{m} load_n(i)^q \right)^{1/q} \leq q \cdot \left(\sum_{i=1}^{m} load^*(i)^q \right)^{1/q}$$

$$\Rightarrow \sum_{i=1}^{m} load_n(i)^q \leq q^q \cdot \underbrace{\sum_{i=1}^{m} load^*(i)^q}. \quad \blacksquare$$

(*)

Suppo[illegible] optimal Makespan $= L^*$.

Suppose optimal Makespan $= L^*$.
We'd like to show that all our machines have a low-load.

(**) $\Rightarrow$

$$\forall i, \quad load_n(i)^q \leq q^q \cdot m (L^*)^q$$

$$\Rightarrow \quad load_n(i) \leq q \cdot m^{1/q} \cdot L^*$$

set $q = \log_2 m$

$$m^{1/q} = m^{\frac{1}{\log_2 m}} = 2.$$

$$\Rightarrow \boxed{load_n(i) \leq (2 \cdot \log_2 m) L^* \quad \forall i}$$

lets try to not cheat

(*) $\Delta\phi = \phi(t) - \phi(t-1)$

$$L_{t-1}(i^*(t))^q \left[\left(1 + \underbrace{\frac{p_{i^*(t),t}}{L_{t-1}(i^*(t))}}_{T_1}\right)^q - 1 \right]$$

either

$\frac{p_{i^*(t),t}}{L_{t-1}(i^*(t))}$ is small $\leq \frac{1}{q}$

or large $> \frac{1}{q}$

In small case, take full Taylor Series to get

$$T_1 \le \cancel{1+} \; \underline{\underline{2q}} \cdot \frac{p_{1^*(t),t}}{L_{t-1}(1^*(t))} \; \cancel{-1}$$

In large case

$$T_1 \le \left(1+\frac{p}{L}\right)^q - 1 \le \left(1+\frac{p}{L}\right)^q$$

$$\le \left(q\cdot\frac{p}{L}+\frac{p}{L}\right)^q$$

$$\le (q+1)^q \cdot \frac{p^q}{L^q}.$$

Therefore, always

$$\Delta\phi \le \underbrace{L^q \cdot 2q \cdot \frac{p_{1^*(t),t}}{L}}_{\text{good term}} + \underbrace{\cancel{L^q} \cdot (q+1)^q \cdot \frac{p^q_{1^*(t),t}}{\cancel{L^q}}}_{\text{bad term}}$$

good term proceeds as always,
bad term is also not bad

b/c we have only

Optimal sol^n terms in it.

(No ALG terms).

$$Alg \le q \cdot Alg^{\frac{q-1}{q}} \cdot Opt^{1/q} + OPT$$

Given 'n' objects, dissimilarity function

$d(\cdot) : [n] \to \mathbb{R}_{\geq 0}$,

group these into 'clusters' so that similar points are more likely in the same cluster.

Example Use Cases :-

① Categorizing data (websites/documents) based on content.

② Clustering songs into ragas.

③ cluster a city into neighborhoods to place utilities.

④ Ensuring diversity in a committee (first find clustering & then pick the committee by choosing from each cluster).

OUR MODELING in this COURSE

- n points, in a generic 'metric space'

way of formalizing a 'dis'-similarity function.

G
I
V

I a dis-similarity
V function.
E
N - Target 'k'.
- Objective Function f

Goal: Partition $[n]$ to clusters $S_1 \uplus S_2 \uplus \cdots S_k$

$$S_i \cap S_j = \phi$$
$$\bigcup_{i=1}^{k} S_i = [n].$$

to minimize $f(S_1, S_2, \ldots, S_k)$

Find efficient (approximation) algos for this problem

WHAT IS A METRIC SPACE?

Given n points,
a distance function $d(i,j)$ is a metric if it satisfies

(i) $d(i,j) \geq 0 \quad \forall \; i,j \in [n]$
(ii) $d(i,i) = 0 \quad \forall \; i \in [n]$

(iii) $d(i,j) = d(j,i) \quad \forall i,j \in [n]$

(iv) $d(i,j) \leq d(i,k) + d(k,j)$
$\forall i,j,k \in [n]$.

EXAMPLES OF METRIC SPACES :-

① - n points can be vectors in $\mathbb{R}^d$
- $d(i,j) = \|v_i - v_j\|_2$
or $\|v_i - v_j\|_p$ for any p.
✓ IS A METRIC

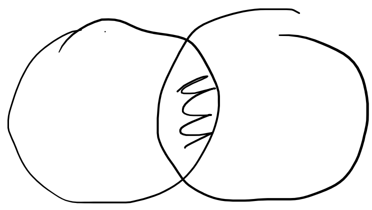
② n points can be vertices of $G=(V,E)$
$d(i,j)$ = shortest path b/w i&j
✓ is a METRIC

③ If G is directed, its not a metric.
[Δ^{le} inequality is fine, but symmetry doesn't hold]

④ Have a collection of documents, each of which contains english words.
Let document i contain S_i (set of words)

then $J(i,j)$ = JACCARD SIMILARITY

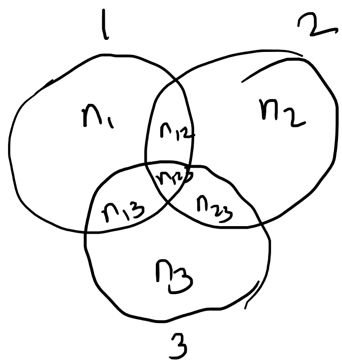
$$= \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$



then $d(i,j) = 1 - J(i,j)$

is a metric

Is Δ^{le} inequality easy to see?



$$d(1,2) = 1 - \frac{n_{12} + n_{123}}{T - n_3}$$

$$d(1,3) = 1 - \frac{n_{13} + n_{123}}{T - n_2}$$

$$d(2,3) = 1 - \frac{n_{23} + n_{123}}{T - n_1}$$

Need to show

$$d(1,2) \le d(1,3) + d(2,3)$$

$$1 - \frac{n_{12}+n_{123}}{T-n_3} \le 1 - \frac{n_{13}+n_{123}}{T-n_2} + 1 - \frac{n_{23}+n_{123}}{T-n_1}$$

$$\frac{n_{13}+n_{123}}{T-n_2} + \frac{n_{23}+n_{123}}{T-n_1} \le 1 + \frac{n_{12}+n_{123}}{T-n_3}$$

[Homework, Check if true]

Common Objective Functions :-

① k-Center Objective Function

(useful for placing police stations, etc)

First used in the 1950s, One of the earliest uses of approx Algos.

Basically, trying to min $\max_{k'=1}^{k} \max_{i,j \in S_{k'}} d(i,j)$

↓

(ie) Min Max Diameter of each cluster

Useful for Police Stations, Petrol bunks, Hospitals, etc.

FORMAL

Given a cluster S,

$d(i,j)$

Given a cluster S,

def. center of cluster = $\underset{i \in S}{\text{argmin}} \max_{j \in S} d(i,j)$

& radius of cluster = $\min_{i \in S} \max_{j \in S} d(i,j)$

<u>k-Center Problem</u>

Given [n] points, find a clustering of [n] into $S_1, S_2, \dots, S_k$ s.t we minimize maximum radius over all S_i.

- Prefers all clusters are smallish over a clustering where one is very big & many are very small.

Possible Algorithms :-

① Can we use set cover?

Elements = [n].

sets correspond to

$$S_i = \{ j : d(i,j) \le r \}$$

for some 'r'. Then choose a min set cover.

– Keep increasing 'r' & stop when we can find a set cover of size 'k'.

---

(2) Idea: Obj fn. tries to avoid any point which is 'far' from all the k centers.

Greedy like Algo :-

1. Pick C_1 as an arbitrary point from $[n]$.
2. For $t = 2, \dots k$
3. Pick $$C_t = \underset{i \in [n]}{\text{argmax}} \underbrace{\min_{1 \le t' \le t-1} d(i, c_{t'})}$$

4. Form the clusters with centers $\{C_1, C_2, \dots C_k\}$ by assigning each point to nearest center.

Idea: In step (3), we are checking if there is any point which hurts the obj. Fn. for the current set of centers. We pick the 'worst' such point & make it a new cluster center.

THM

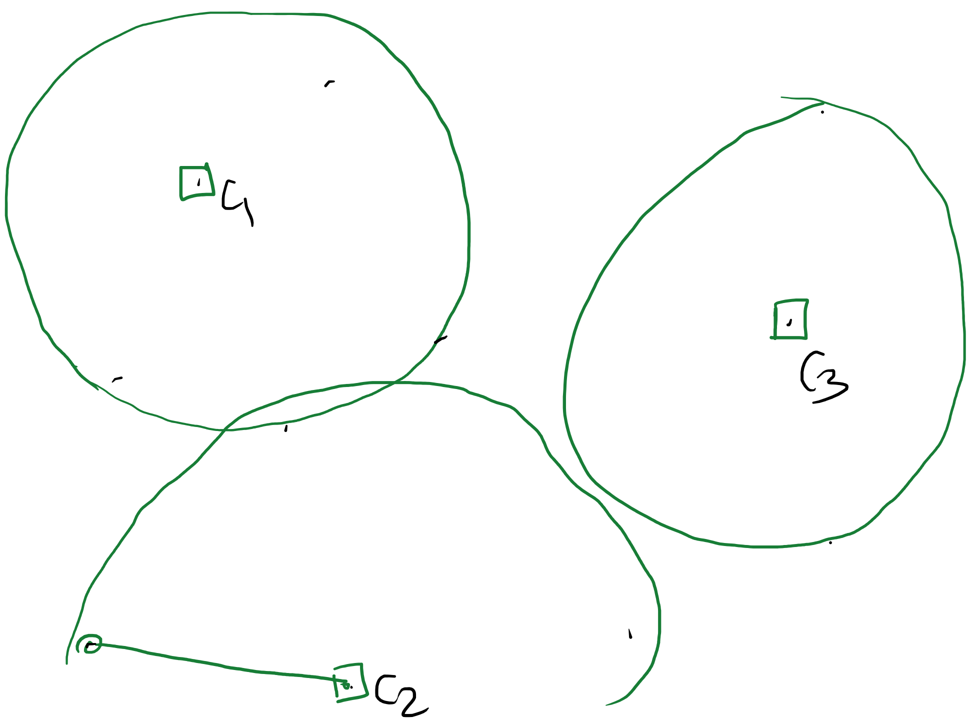
ALGO is 2-approx for k-Center problem

The k-Center Problem

09 March 2021 09:00

Given set P of 'n' points in a metric space, choose a subset of k centers $\mathcal{C} = \{c_1, c_2, \ldots, c_k\} \subseteq P$ such that we can cluster the points in P with minimum radius clusters.

(ie) minimize $\max_{p \in P} \min_{c_i \in \mathcal{C}} d(p, c_i)$.



Once we choose the centers, the clustering to minimize the max-radius is easy - each point is assigned to its nearest center.

Moreover, Max radius of all clusters

$$= \max_{p \in P} \min_{c_i \in \mathcal{C}} d(p, c_i)$$

Let's focus on the distance of each point to its nearest center

Once we fix the k centers

	dist-to-closest-center
p_1	5
p_2	10
	$\boxed{12}$
⋮	2
	0
p_n	7.

Restating the problem:

Choose k centers $\{c_1, c_2, \ldots c_k\} \subseteq P$

to minimize

$$\boxed{\max_{p \in P} \min_{c_i \in \mathcal{C}} d(p, c_i)}$$

- Motivates the greedy algorithm

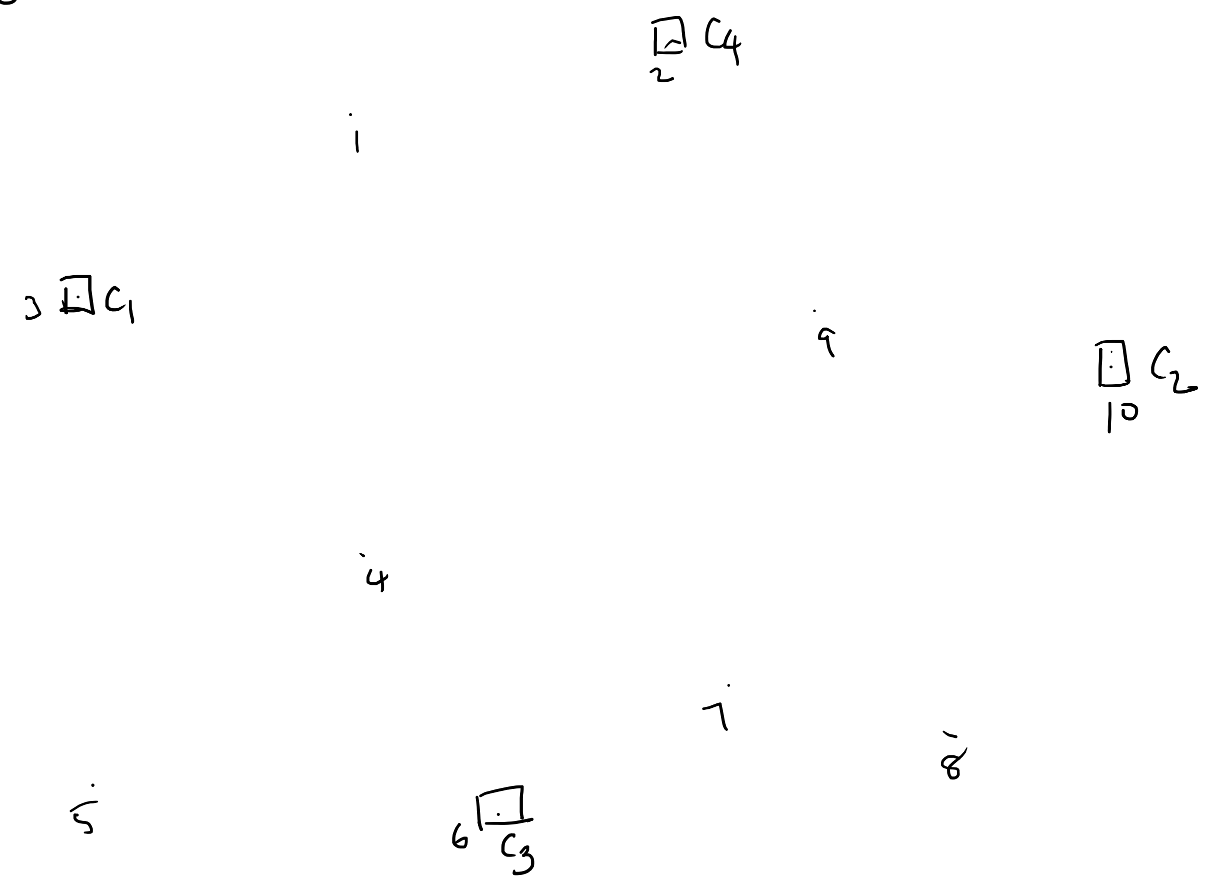
① choose c_1 arbitrarily from P

② for $t = 2, \ldots, k$

③ $\hookrightarrow$ let $c_t = \underset{p \in P}{\operatorname{argmax}} \; \min_{1 \le t' \le t-1} d(p, c_{t'})$.

④ Form the Clustering using $c_1, c_2, \ldots, c_k$ as centers. ☐

Toy Example



Suppose $k = 4$

THEOREM

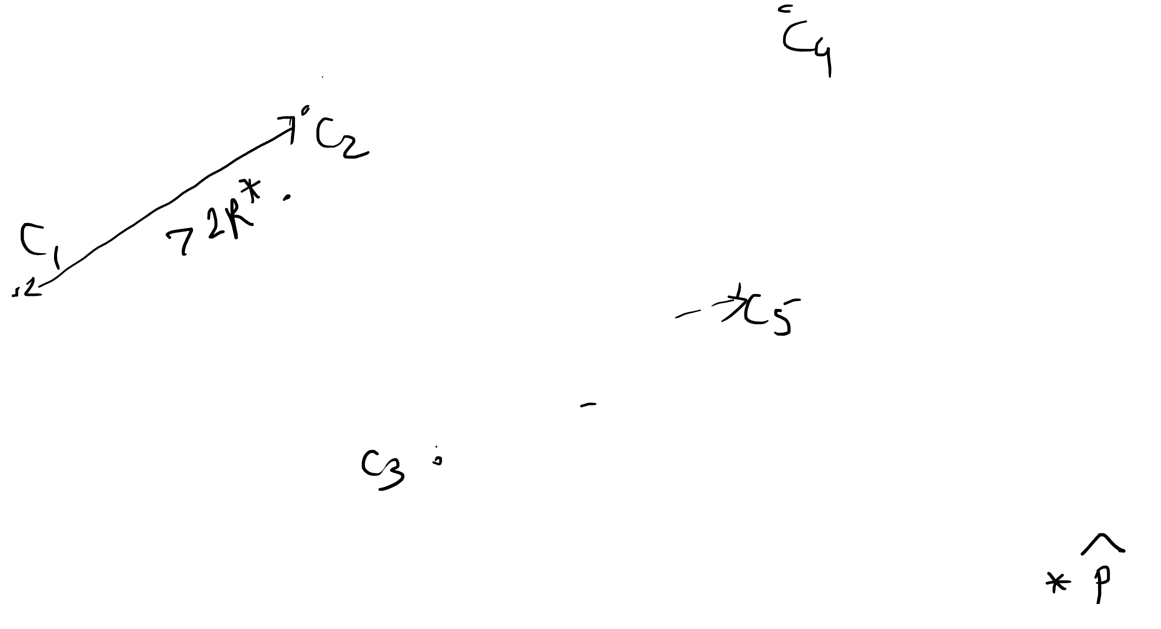
If optimal clustering has objective value R^*, our clustering has objective value $\leq 2R^*$.
($\Rightarrow$ 2-approximation).

Proof :-
Let $c_1, c_2, \ldots c_k$ be algorithms centers.
Let $c_1^*, c_2^*, \ldots, c_k^*$ be optimal centers.

Suppose for contradiction, our clustering has objective value $> 2R^*$.

$\Rightarrow \exists$ a point $\hat{p} \in P$ st it is far from all $c_1, c_2, \ldots c_k$

$\Rightarrow d(\hat{p}, c_i) > 2R^* \;\; \forall\, i = 1, 2, \ldots k$.



$* \hat{p}$

Because greedy algo chose

$$c_t = \underset{p \in P}{\operatorname{argmax}} \min_{1 \le t' \le t-1} d(p, c_{t'})$$

and $\min_{1 \le t' \le t} d(\hat{p}, c_{t'}) > 2R^*$

$\Rightarrow$ $\hat{p}$ is a candidate for all of $c_1 \ldots c_k$
but didn't get picked

$\Rightarrow$ $\forall t$, $\min_{1 \le t' \le t-1} d(c_t, c_{t'}) > 2R^*$
as well.

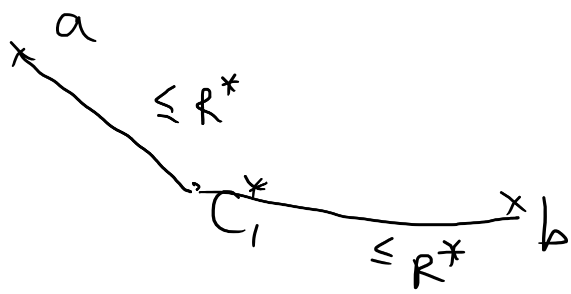
$\Rightarrow$ There exist $k+1$ points in P
namely $\{c_1, c_2, \ldots, c_k, \hat{p}\}$ st
each pair of points is at
$> 2R^*$ distance.

But now how are these points clustered well in OPT?

$\exists$ some cluster, say C_1^* with

≥ 2 points a and b

from $\{c_1, c_2, \ldots, c_k, \hat{p}\}$



Because OPT has radius R^*

$d(a, C_1^*) \leq R^*$ and

$d(b, C_1^*) \leq R^*$

Now, Δ^{le} inequality gives

$$d(a,b) \leq d(a, C_1^*) + d(C_1^*, b)$$
$$\leq 2R^*$$

[CONTRADICTION]

$\boxtimes$

The k-Median Problem

09 March 2021 09:36

Same type of input as k-center, but a slightly different objective function.

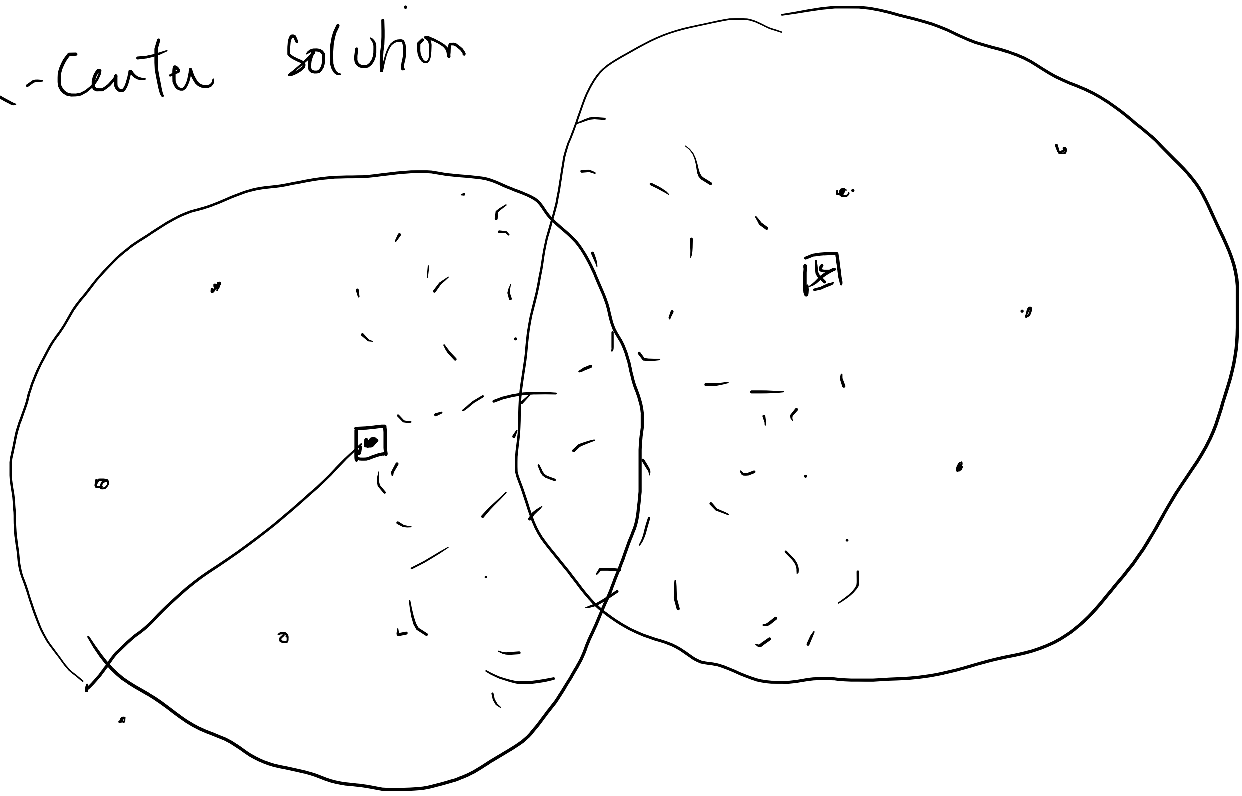
Given n points in a metric space
Find k centers $c_1, c_2, \ldots c_k$
to minimize $\boxed{\sum_{p \in P} \min_{i=1}^{k} d(p, c_i)}$.

Motivation of Objective Function

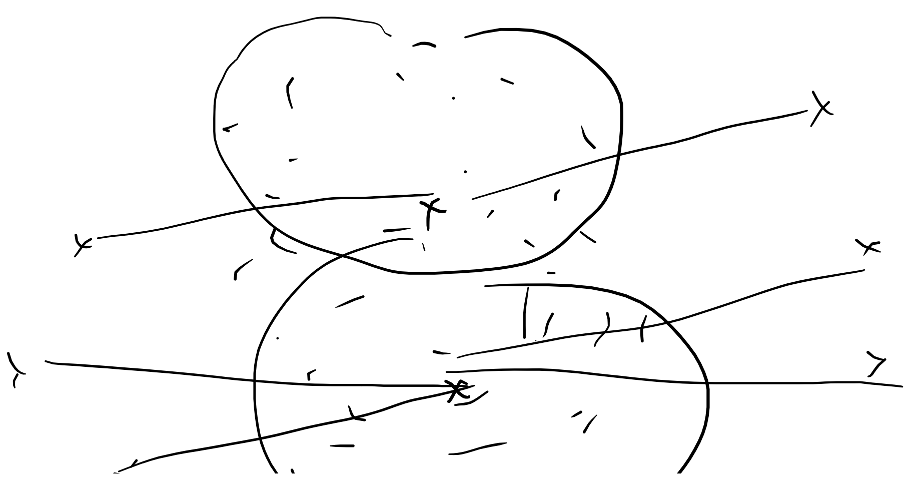
(1) Maybe we are trying to lay cables from k stations to all points of the city, where to place the power stations?

Total Cable length matters more than max.

k-Center solution



k-Median solution



Try to cover the dense region better while being OK with a few points paying a large cost.

Yet another problem

k-Means Problem.

Same as above, objective is

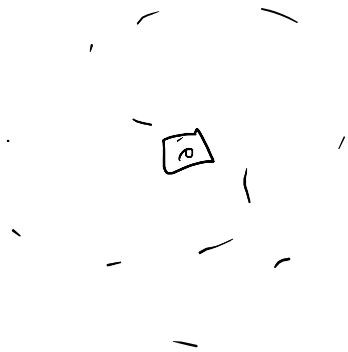
$$\text{minimize} \sum_{p \in P} \min_{i=1}^{k} d(p, c_i)^2$$

Has lots of applications in ML, unsupervised learning, etc.

especially when points are vectors in $\mathbb{R}^d$ and

$$d(i, j) = \|v_i - v_j\|_2$$

Has very nice physics connections to concepts such as



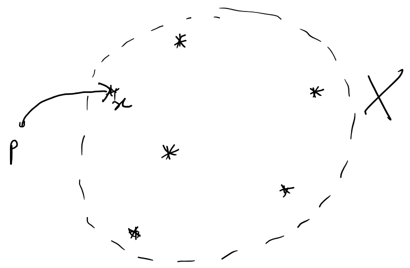
Center of gravity, etc.

Point which minimizes sum of squared distances is the 'mean'/centroid of dataset.

K-Means is a natural generalization to k-clusters.

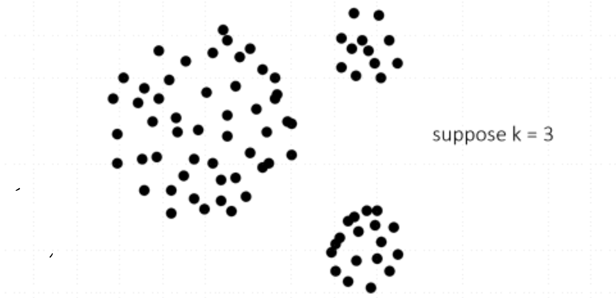
Given n points in a metric space P
find k centers $\mathcal{C} = \{c_1, c_2, \dots c_k\}$
to minimize $\sum_{p \in P} d(p, \mathcal{C})$

where $\boxed{d(p, X) = \min_{x \in X} d(p, x)}$

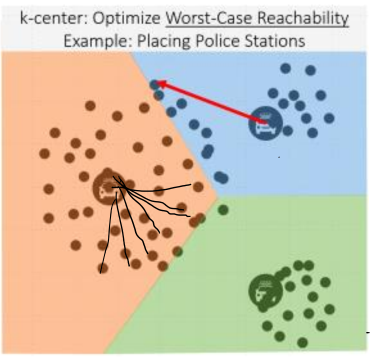


Recall k-Center objective was to minimize $\max_{p \in P} d(p, \mathcal{C})$

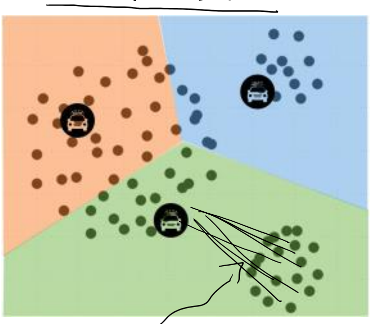
Illustration of k-Median vs k-Center :-



k-Center Solution



k-Median Solution



Maybe its ok for a few points to pay a large cost if Many points pay less.

---

k-Center admits 2-approximation using Greedy algorithm. what about k-Median?

k-center ... approx ... Greedy algorithm. What about k-Median?

Simple greedy-type Algos don't end-up being very good, we can resort to Linear Programming.

Y_i = Variable for whether i is chosen as a center or not.

X_{ij} = variable for whether point j is assigned / clustered to center @ i

LINEAR PROGRAM (K-MEDIAN)

$$\text{Min} \sum_{j} \left(\sum_{i} d(i,j) \cdot x_{ij} \right)$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 \qquad \forall j \in P$$

$$\sum_{i=1}^{n} y_i \leq k$$

$$x_{ij} \leq y_i \qquad \forall i,j$$

$$x_{ij} \geq 0$$

$$y_i \geq 0$$

Lemma 1

Let (x^*, y^*) be an optimal LP solution

Then $LP^* = \sum_j \sum_i d(i,j)\, x^*_{ij} \leq OPT$

where OPT is the k-Median cost of optimal solution.

Proof

Unknown optimal k-Median solution is feasible for the LP, which can only do better.

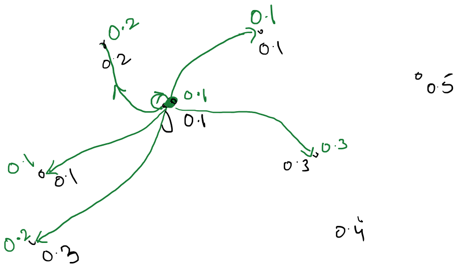
QUESTION

How do we "round" this fractional solution into a good clustering?

Also note :-

In the optimal LP solution, once we know the y^* values, the x^*_{ij} values can be easily derived

Let's look at some point j.



white values are y_i^* values.

green values are x^*_{ij} values

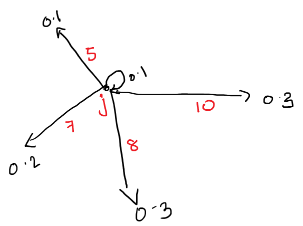
Idea

Infer some basic properties of what the LP optimal is trying to do.

In particular, let

D_j = LP-distance that point j incurs in the optimal solⁿ

$= \sum_i d(i,j) \cdot x^*_{ij}$



→ Distances are in Red

→ x^*_{ij} values are in white

D_j = LP-distance of point j

$= 0.1*0 + 0.1*5 + 0.2*7 + 0.3*8 + 0.3*10$

$= 0.5 + 1.4 + 2.4 + 3$

$= 7.3$

LP tries to cover point j by connecting it to a center at "distance" 7.3, so we use that as a guide.

LEMMA (2)

For any point j, let $B_j = \{ i : d(i,j) \le 2D_j \}$ be the points at distance $\le 2D_j$ from j.

Then $\sum_{i \in B_j} y_i^* \geq \sum_{i \in B_j} x_{ij}^* \geq \frac{1}{2}$.

Proof :-

Suppose not, and $\exists\, j$ st

$$\sum_{i \in B_j} x_{ij}^* < \frac{1}{2}.$$

Then

$$D_j = \sum_i d(i,j)\, x_{ij}^* = \sum_{i \in B_j} x_{ij}^*\, d(i,j) + \sum_{i \notin B_j} x_{ij}^*\, d(i,j)$$

$$> 0 + \Big(\sum_{i \notin B_j} x_{ij}^*\Big) \cdot 2D_j$$

$$> 0 + \frac{1}{2} \cdot 2D_j$$

$$= D_j \qquad \Rightarrow\Leftarrow.$$

For each j, if we place some center at a point in B_j, then its connection distance $\le 2D_j$

$\Rightarrow$ Overall cost $\le \sum_j 2D_j \le 2\,LP^* \le 2\,OPT$

$\boxed{12/03/2021}$

Algo (1)

(1) Sort j such that

$D_{j_1} \le D_{j_2} \le \dots \le D_{j_n}$

(2) Pick the set "near-independent" points J^* as follows

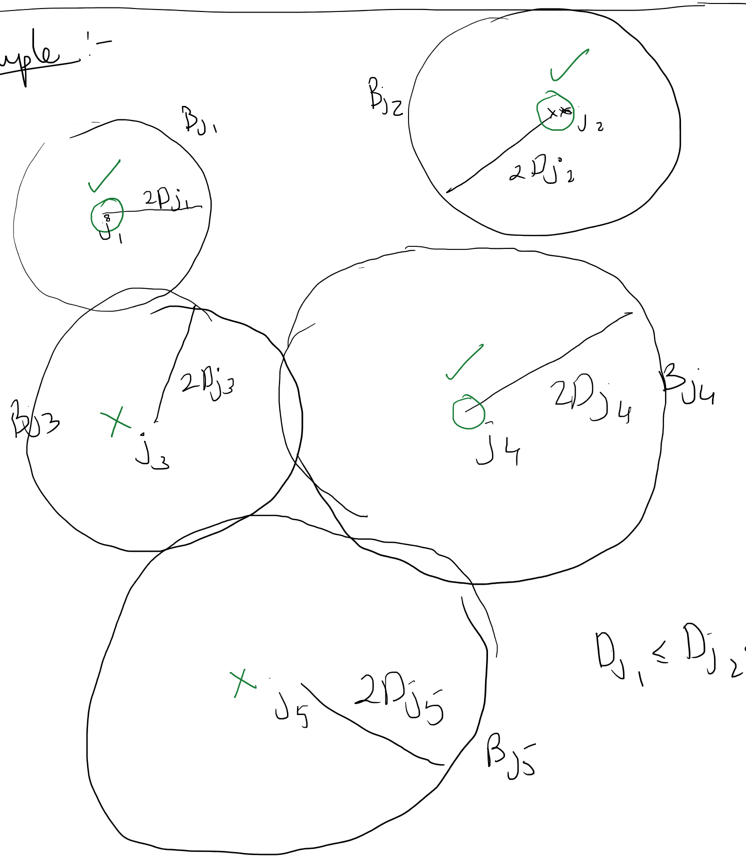
- For $\ell = 1 \dots n$ (in sorted order)

  If there exists no $j' \in J^*$ s.t $d(j_\ell, j') \le 2D_{j_\ell} + 2D_{j'}$

  then add j_ℓ to J^*.

③ Open a center at each $j \in J^*$

Example :-



$D_{j_1} \le D_{j_2} \le \dots \le D_{j_5}$

Finally, J^* contains j_1, j_2, and j_4.

Lemma ①

J^* is s.t $\forall\ j', j'' \in J^*$

$$B_{j'} \cap B_{j''} = \phi .$$

Proof

Sps $B_{j'} \cap B_{j''} \neq \phi$ and say i belongs to both.

then $d(j', j'') \le d(j', i) + d(i, j'')$

$\le 2D_{j'} + 2D_{j''}$

(contradiction to whichever got added later)

What did we do?

① Somehow identify points which are

not too overlapping

(2) Remaining points are "close" to the J^* points $\Rightarrow$ if we can handle the J^*, we can hope to handle the rest also.

SIMPLE ALGO (1) { Won't exactly give k centers, but might open up to $2k$ centers }.

$\boxed{\text{For each } j \in J^*, \text{ open a center at } j.}$

Claim (1)

$\forall j$, distance of j to nearest open center $\leq 4 D_j$

Claim (2)

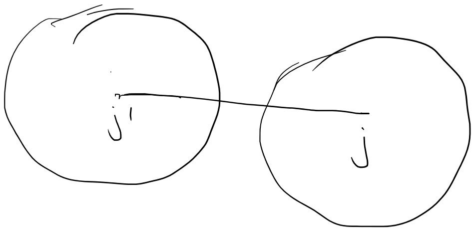
Total # open centers $\leq 2k$.

Proof

Claim (1) :-

If j got added to J^*, then dist of j to nearest center $= 0 \leq D_j$ ✓

If j didn't get added, there must exist a j' st



$$d(j, j') \leq 2D_j + 2D_{j'}$$
$$\leq 4D_j \quad ✓$$

So dist to nearest open center $\leq 4D_j$

Pf of claim 2 :

In each B_j, $\sum_{i \in B_j} y_i^* \geq \frac{1}{2}$

Moreover $B_{j_1} \cap B_{j_2} = \phi$ for $j_1, j_2 \in J^*$

So, simply sum over all $j \in J^*$

$$\sum_{j \in J^*} \sum_{i \in B_j} y_i^* \leq \sum_{i=1}^{n} y_i^* \leq k$$

↑ LP constraint

$$\sum_{j \in J^*} \sum_{i \in B_j} y_i^* \geq \sum_{j \in J^*} \frac{1}{2} = \frac{|J^*|}{2}$$

$$\Rightarrow \boxed{|J^*| \leq 2k}$$

Bi-Criteria Approximation Algorithms

Given an instance of k-Median, with optimal cost = OPT, we can efficiently find a solution which opens $2k$ centers and has cost $\leq 4\,OPT$

$\leq 4\,OPT$

How do we improve to a pure k-Median solution?

I D E A

Focus on J^*, pair them up in J^*, so that in each pair $\sum y_i^* \geq 1$ and then handle each pair

O(1)-Approximation for k-Median

15 March 2021 10:01

Recap

1) Solve LP
2) Get (x^*, y^*) as solution
3) Define $D_j = \sum_i d(i,j)\, x^*_{ij}$
4) Define $B_j = \{ i : d(i,j) \le 2D_j \}$
5) $y^*(B_j) = \sum_{i \in B_j} y_i^* \ge \frac{1}{2} \quad \forall j$
6) J^* is the near-independent far-away points in J.
7) $\forall j \notin J^*, \exists j' \in J^*$ st $d(j,j') \le 4D_j$ and $D_{j'} \le D_j$
8) $|J^*| \le 2k$
9) $\forall j, j' \in J^*, \quad B_j \cap B_{j'} = \phi$

From $1 \to 8$, we got a 4-approximation

1. which opens $2k$ cluster centers
2. Today, we'll try ~~to~~ make it a genuine k-clustering which opens $\le k$ centers while being slightly worse in cost.

Idea

- Pair up points in J^*.
- Open one center in each pair

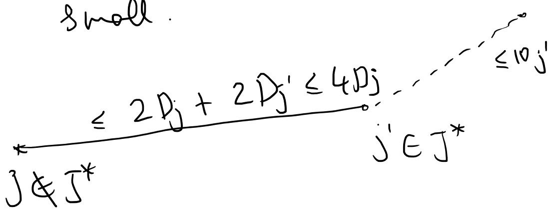
$|J^*| \le 2k \Rightarrow$ # centers open $\le k$. :)

- How do we analyze the cost?
- How to pair points in J^*?
- How to decide which center to open in a pair?

a pair ?

$\boxed{\text{We'll try to ensure the connection cost of each point } j \in J^* \text{ is at most, say, } 10 D_j.}$

$\Rightarrow$ Connection cost of points not in J^* is also small.



$\Rightarrow$ conn. cost of $j \leq$

$$2D_j + 2D_{j'} + 10 D_{j'}$$
$$\leq 14 D_j.$$

$\Rightarrow$ It suffices to show that points in J^* have low connection cost.

Q| How do we choose k centers to ensure that points in J^* have low conn cost ($\propto D_j$)

PAIRING ALGORITHM

- Among J^*, choose the closest pair of points say (j_1, j_2) and match them.
- Remove j_1, j_2 from J^* and repeat

---

pairs + (singleton, If left) $\leq k$.

For now, let's assume no singleton left.

Can handle it easily later.

Can handle it easily later.

We'll come up with a randomized selection procedure which ensures

① Each point is chosen as a center with prob y_i^*

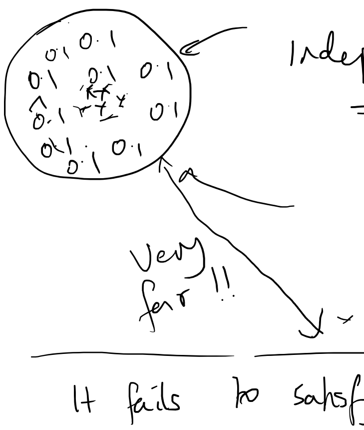
② Total # of centers opened $\leq k$

③ For each matched pair (j_1, j_2), we definitely open ≥ 1 center from among the points $B_{j_1} \cup B_{j_2}$.

SATISFYING just ① is easy.

Each i will Independently choose itself as a center w.p y_i^*.

As a sol^n idea, not great because there could be a region in space where we don't open any center



Independent rounding $\Rightarrow (1-0.1)^{10} \approx 1/e$ prob. not opening any center in this cloud

---

It fails to satisfy ② & ③ also.

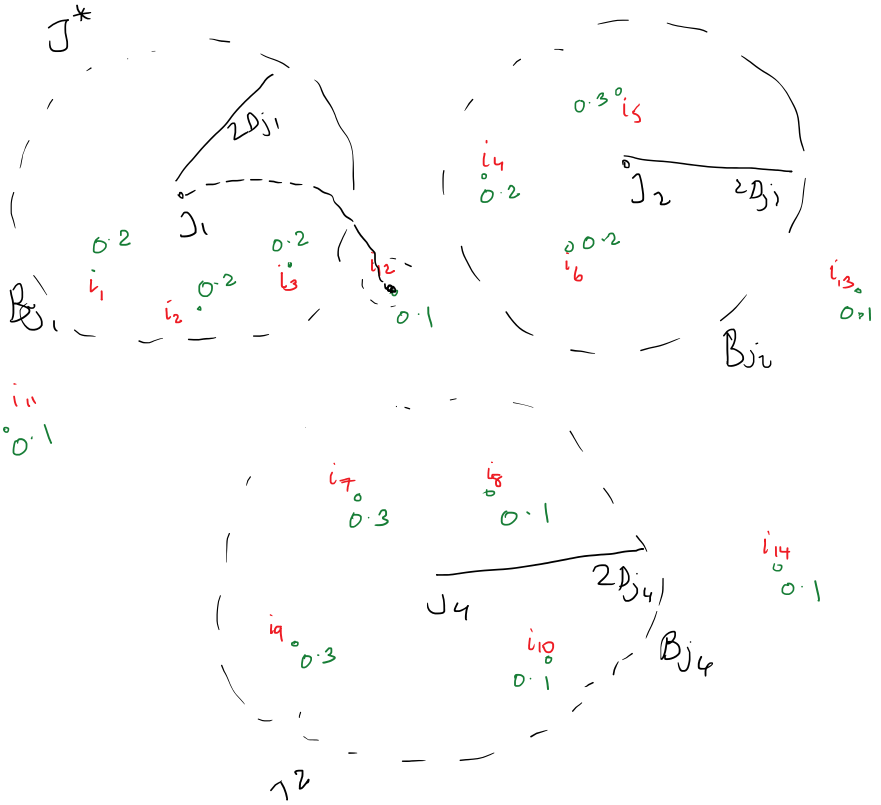
In fact ③ precisely tries to address the issue of completely missing local regions in space, which is the drawback of independent rounding.

α - point rounding / dependant rounding

- Nice properties of prob $\propto y_i^*$

along with satisfying extra constraints.

Selection Procedure Illustration

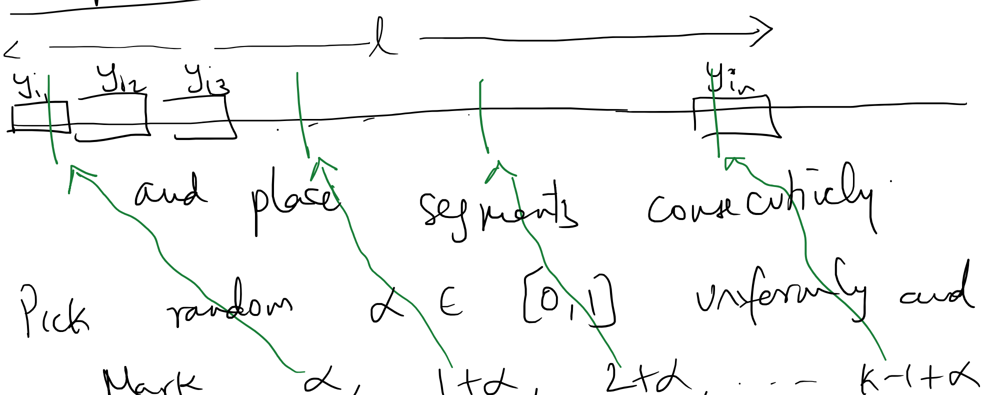


Example in picture above

j_1 connects to $i_1, i_2, i_3, i_{12}, i_{11}, i_4$

$\forall i$, we want to associate y_i length segment on the real line, and put them consecutively

Attempt ①: Order arbitrarily $i_1, i_2, \dots i_n$



and place segments consecutively

Pick random $\alpha \in [0,1]$ uniformly and Mark $\alpha, 1+\alpha, 2+\alpha, \dots k-1+\alpha$

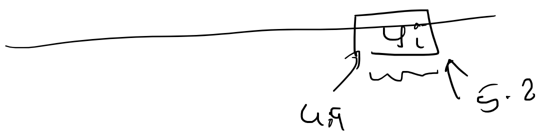
Because $\sum y_i = k$, length of line segment $l = k$.

If a dart intersects a segment y_i, open a center at i.

Nice properties:-

$\forall i$, $\Pr[i \text{ is opened as center}]$

First dart crosses i, only if $0 \le \alpha \le y_i$, which has probability y_i,



This is true for any i,

$\Pr[i \text{ is chosen as center}] = y_i$

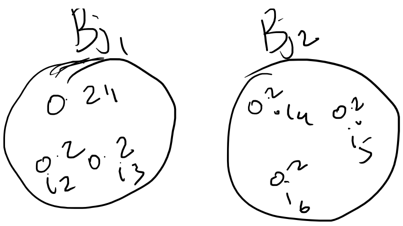
(2) ✓ # centers $= k$

because $(k+1)^{th}$ dart will lie outside the system.

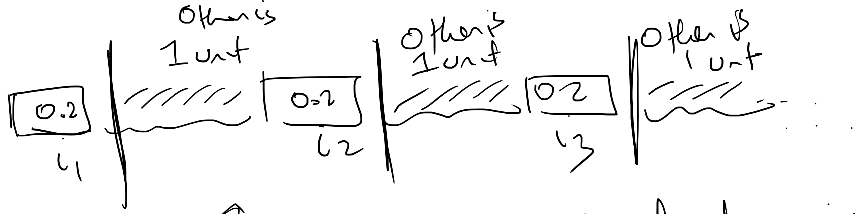
(3) X doesn't satisfy the property that in a "local region", there is one open center

(e.g) $\forall (j, j') \in M$, we may not open any center in $B_{j_1} \cup B_{j_2}$

B_{j_1} B_{j_2}



1



If $\alpha > 0.2$, all darts miss B_{j1} and B_{j2}

So, need to preserve locality in some manner.

Goals:

1. Ensure B_j is contiguous for each j
2. Ensure B_{j_1} and B_{j_2} are contiguous for each $(j_1, j_2) \in M$
3. Ensure other close points to j are contiguous for each j

for each i not in any B_j for $j \in J^*$,
ensure it is contiguous with its nearest $j \in J^*$

$$O_j = \left\{ i : d(i,j) \leq d(i,j') \ \forall j' \in J^*, j' \neq j \right\}$$

No ties in distances

lets assume that we break ties arbitrarily.

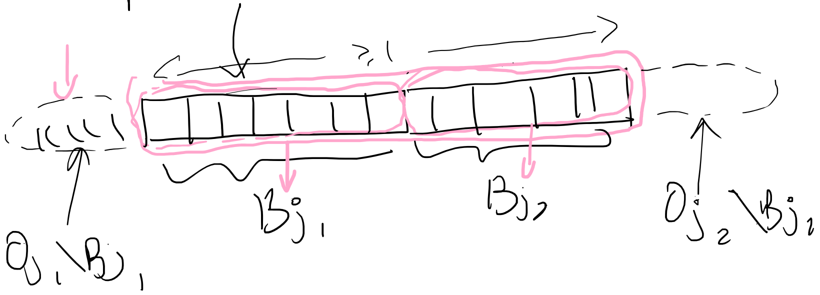
No ties in distances } we break ties arbitrarily.

(ii) $d_1 \neq d_2$ for all pairs

(3) $\forall j$, want O_j appears consecutively

Will this work?

Take pair $(j_1, j_2) \in M$



repeat for all pairs

Now since $y(B_{j_1}) + y(B_{j_2}) \geq 1$

we will definitely open one center in each "pair".

with same dart throwing algorithm.

17/03/2021

Algorithm

- Recall defn of B_j, O_j, J^*, D_j, x^*, y^*
- Recall Pairing M.
- Place the points on a line and do the "α-point rounding"

Ensure that for each pair,

B_{j_1} is contiguous

B_{j_2} is contiguous

$O_{j_1} \setminus B_{j_1}$ contiguous with B_{j_1}

$O_{j_2} \setminus B_{j_2}$ contiguous with B_{j_2}

- Open all centers which are crossed by the darts.

Lemma ①

Each i is opened with prob y_i^*

Lemma ②

For any pair $(j, j') \in M$, at least one point is opened from $B_{j_1} \cup B_{j_2}$ with probability 1.

<—— length ≥ 1 ——>

$\underbrace{\quad\quad\quad}_{B_{j_1}} \; \underbrace{\quad\quad\quad}_{B_1}$

because $y^*(B_{j_1}) + y^*(B_{j_2}) \geq 1$

and $B_{j_1} \cap B_{j_2} = \phi$

✓ gap b/w any 2 darts $= 1$

ENSURES "LOCALITY PRESERVING ROUNDING".

Lemma ③

For each point $j \in J^*$, let

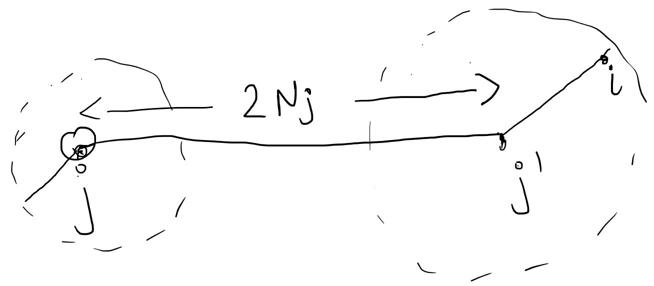
$N_j = \frac{1}{2} \min_{j' \in J^*} d(j, j')$ be half

$N_j = \frac{1}{2} \min_{\substack{j' \in J^* \\ j' \neq j}} d(j, j')$ be half the dist. to nearest other pt in J^*

Then, there is always an open center within $6 N_j$

Proof

Let's focus on j and j' – its nearest other pt from J^*



Either $(j, j') \in M$ or not

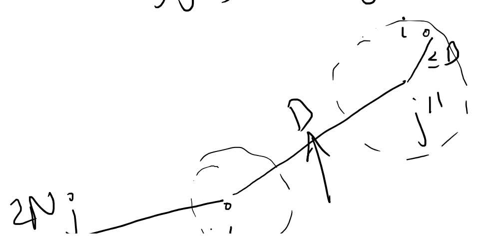
Case ①: If $(j, j') \in M$.

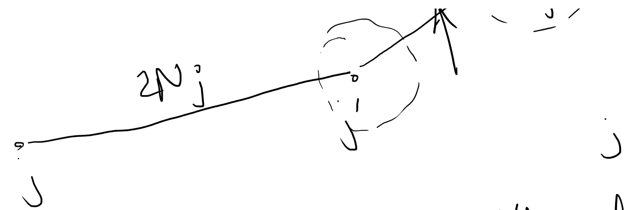
There is open center i among $B_{j_1} \cup B_{j_2}$

$\Rightarrow d(j, i) \leq 4 N_j$

Case ② $(j, j') \notin M$.

$\Rightarrow \exists\, j''$ st $d(j', j'') \leq 2N_j$ and $(j', j'') \in M$.





But again, there will always be an open center in $B_{j'} \cup B_{j''}$

$$d(j, i) \le 2N_j + D + D$$
$$\le 6N_j \quad (\Delta^{le} \text{ inequality}).$$

Rest of Analysis :-

If for some j, $D_j \geq 0.1 N_j$ then such j's are very happy (D_j = LP connection cost)

(Their real connection cost ≤ 60 their LP connection cost)

$\boxed{\text{Real problem happens if } D_j \text{ is much smaller than } N_j}$

$\downarrow$

Let's analyze D_j :-

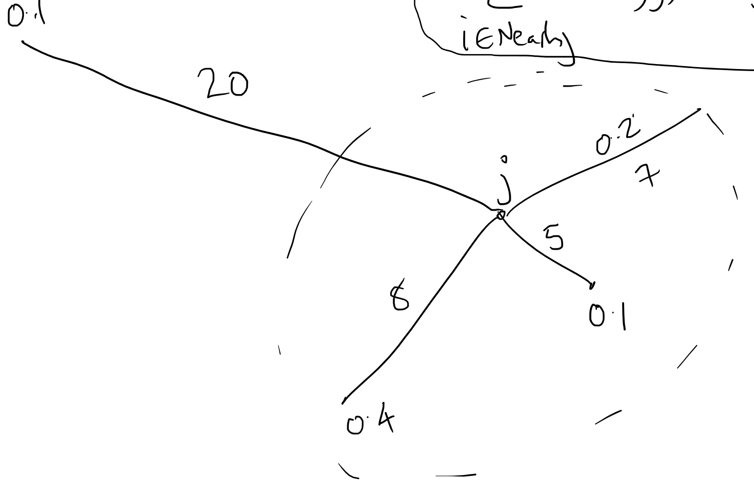
$$D_j = \sum_i d(i,j)\, x^*_{ij}$$

$$= \sum_{\substack{i : d(i,j) \\ < N_j}} d(i,j) \cdot x^*_{ij} + \sum_{\substack{i : d(i,j) \\ \geq N_j}} d(i,j)\, x^*_{ij}$$

$i : d(i,j) < N_j$ $\quad$ $i : d(i,j) \geq N_j$

$$\boxed{D_j \geq \sum_{i : d(i,j) < N_j} d(i,j)\, x^*_{ij} + N_j \sum_{i : d(i,j) \geq N_j} x^*_{ij}}$$

$$= \sum_{i \in Nearby} d(i,j)\, x^*_{ij} + N_j \Big(1 - \sum_{i \in Nearby} x^*_{ij}\Big)$$

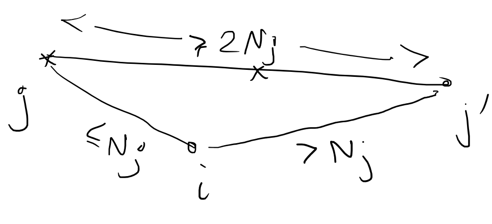


$5 * 0.1 + 7 * 0.2$
$+ 8 * 0.4$

What's the expected connection cost of this point j, in our rounding scheme?

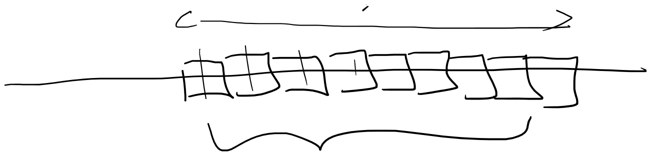
Notice : all the points i s.t $d(i,j) < N_j$ belong to O_j

for any other j'



Now, we ensured that such points can be placed contiguously on the line.

All points $i : \{d(i,j) < N_j\}$ are placed contiguously.



$i : d(i,j) < N_j \Leftarrow$ Nearby Points

Expected cost of j's connection

$= \Pr(\text{One of the nearby pts chosen}) \cdot E[\text{distance} \mid \text{nearby points chosen}]$

$+ \Pr(\text{Nearby pt not chosen}) \, E[\text{distance} \mid \underbrace{\text{Nearby pt not chosen}}]$

$$= \left(\sum_{i : d(i,j) < N_j} y_i^*\right) \left[\frac{\sum_{i : d_{ij} < N_j} d_{ij} \, y_i^*}{\sum_{i : d(i,j) < N_j} y_i^*}\right] + \left(1 - \sum_{i : d_{ij} < N_j} y_i^*\right) \cdot 6 N_j$$

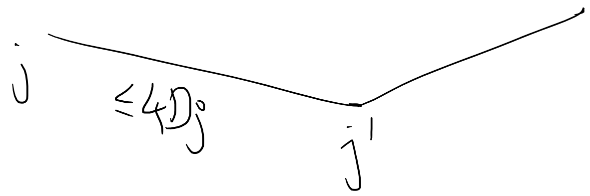
$\leq 6 D_j$

Corollary

$\forall j \in J^*$, $E[\text{connection cost of } j] \leq 6 D_j$

Corollary

$\forall j \notin J^*$, $E[\text{conn. cost of } j] \leq 10 D_j$



$\exists\ j'$ st $d(j, j') \leq 4D_j$ &

$D_{j'} \leq D_j$

$\Rightarrow E[\text{conn. cost of } j] \leq 4D_j + E[\text{conn cost of } j']$

$\leq 4D_j + 6D_{j'}$

$\leq 10 D_j$

Takeaway

① $O(1)$ for K Median

② Respecting hard constraints $\sum y_i \leq k$ during rounding is hard, we need "dependant rounding" ideas

③ α-point rounding is a good way to ensure this.

Next 2-3 lectures, yet another algo. for k-Median which gets rid of ② in a clever way.

k-Median via Lagrangean Relaxation

19 March 2021 12:07

Given metric (X, d) $|X| = n$ points

$d(\cdot)$ is distance function (metric)

- $d(i,j) + d(j,k) \geq d(i,k)$ $\forall i,j,k \in [n]$
- $d(i,j) = d(j,i)$
- $d(i,i) = 0$

Choose k points as centers and assign each point in X to nearest center to minimize total "assignment distance"

(ie) $\sum_{j=1}^{n} d(j, S)$ where $d(j,S) = \min_{i \in S} d(i,j)$

and $|S| = k$

Recall LP

Total Assignment distance $\leftarrow$ $\text{Min} \sum_j \sum_i d(i,j) x_{ij}$

every point is assigned $\leftarrow$ $\sum_i x_{ij} \geq 1 \quad \forall j$

Center must be open $\leftarrow$ $x_{ij} \leq y_i \quad \forall i,j$

$\sum y_i \leq k$

$$\sum y_i = k$$

Total k centers

$$\left.\begin{array}{l} x_{ij} \geq 0 \\ y_i \geq 0 \end{array}\right\}$$

Method of Lagrangean Relaxation

Idea:

We had a lot of difficulty in rounding the LP solution to preserve $\sum y_i \leq k$ (allowing violations was much easier)

Instead, let us push this constraint to the objective function

---

New LP: LP2

$$\min_{x,y} \sum_j \sum_i d(i,j)\, x_{ij} + \lambda\Big(\underbrace{\sum y_i - k}\Big)$$

$$\text{LP2}\left\{\begin{array}{lll} \sum_i x_{ij} \geq 1 & & \forall j \\ x_{ij} \leq y_i & & \forall i,j \\ \left.\begin{array}{l} x_{ij} \geq 0 \\ y_i \geq 0 \end{array}\right\} & & \forall i,j \end{array}\right.$$

$\forall \lambda \geq 0,$

$$\text{OPT(LP2)} \leq \text{OPT(LP)}.$$

In LP2, we can effectively ignore the $-\lambda k$ constant in the obj (It is the same for all x, y solutions).

$\boxed{\text{LP3}}$ is LP2 without $-\lambda k$ term

LP3

$$\text{Min} \sum_j \sum_i d(i,j)\, x_{ij} + \lambda \sum y_i$$

$$\sum_i x_{ij} \geq 1 \quad \forall j$$

$$x_{ij} \leq y_i \quad \forall i,j$$

$$x_{ij} \geq 0$$

$$y_i \geq 0$$

$$\boxed{\text{OPT(LP3)} \leq \text{OPT(LP)} + \lambda k}$$

Called the FACILITY LOCATION PROBLEM.

Open a set S of centers and

assign points to nearest center,

but instead of asking $|S| \leq k$, we add a cost of λ for each open center.

obj. fn = $\min \sum_j d(j, S) + \lambda |S|$.

HOPE

(1) To find an "easy" Apx algo for facility location

(2) Maybe for suitable λ, the algo actually opens k centers

$\Downarrow$

$\boxed{Q}$ Do (1) & (2) $\Rightarrow$ Apx Algo is good for k-Median?

$\boxed{Ans}$ This is almost true, need a little more guarantee from the approx algo for Facility Locations.

FACILITY LOCATION OBJ. FN

$\swarrow$ CONN. COST (DISTANCE)

$\searrow$ FACILITY OPENING COST (λ)

$\nearrow$ 2^{nd} cost is what came from the ...

k-Median constraint

Algo A is a C-LAGRANGEAN approximation for FACILITY LOCATION if

CONN.COST(A) + C.FACILITY COST(A)

$\leq$ C.[CONN(OPT) + FAC(OPT)]

without the C in the LHS, it is a traditional C-approximation.

[Recall: these are minimization problems, $C \geq 1$].

WHY does ① & ② along with C-Lagrangean Algo $\Rightarrow$ good algo for k-Median?

CONNCOST(A) + C.λ.k $\leq$ C [CONNCOST(k-MedianOPT) + λk]

Plugged in the k-Median OPT as a feasible soln for Facility Location OPT.

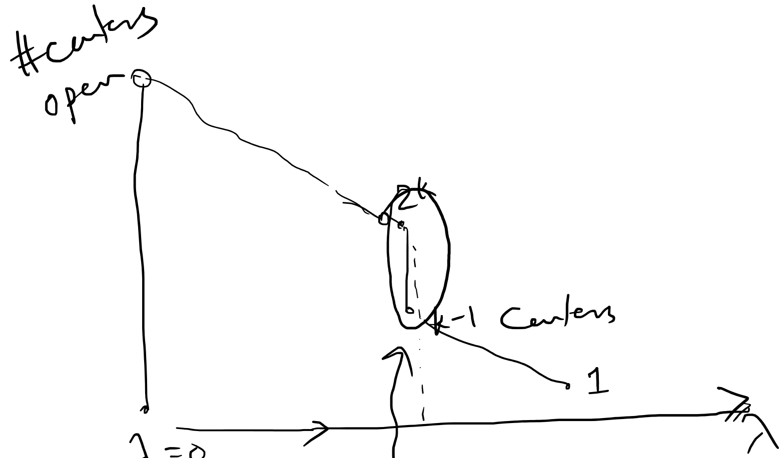
for Facility Location OPT.

$$\boxed{\begin{array}{c}\text{CONN Cost}(A) \leq C \cdot \text{CONN Cost}(k\text{ Median OPT}) \\ \text{and } A \text{ opens } k \text{ centers (from (2))}\end{array}}$$

$\Downarrow$

For the suitable λ, A is a C-approx. for k-Median

$\Downarrow$



What could happen in reality is that

as we keep increasing λ, there <u>need not exist any</u> point λ where Algo opens exactly k centers.

(i.e. For $\lambda - d\lambda$, it opens $\geq k$ centers and $\lambda + d\lambda$ it opens $< k$ centers

could very well happen but there's a very nice way to deal with it

Remains to do

1. Design C-LAGRANGEAN algo for Facility Location

Next 1-2 lectures

2. Design the "combiner procedure" for $\lambda - d\lambda$ and $\lambda + d\lambda$ problem.

26-03

Any qns about 1?

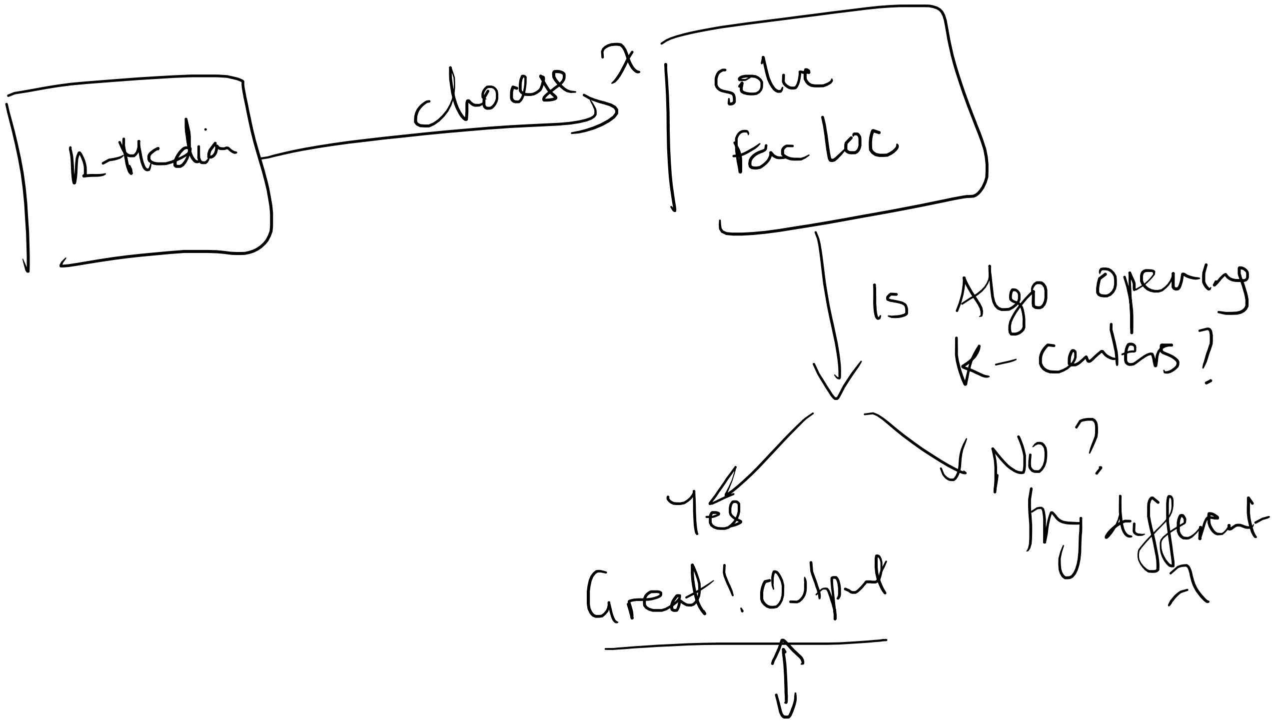
We'll assume black-box access to algorithm which has

$$\text{CONN-Cost} + 3\,\text{FAC.COST} \leq 3\,\text{OPT}(\text{FAC.LOC}).$$

How do we use this for k-Median approximation?

Idea

Given a k-Median problem, lets choose an "ideal λ" and solve the Facility location instance.



Why is this sol^n good for k-Median?

Here is where we use the
3-Lagrangean-Approx

If $\exists\ \lambda$ where Algo opens k centers,

then

$$\text{CONN Cost}(\text{Alg}) + 3\lambda k \le 3(\text{OPT}(FL))$$
$$\le 3\left(\text{CONNCost}(\underset{\text{OPT}}{k\text{-Median}}) + \lambda k\right)$$

$\Rightarrow$

$$\text{Conn Cost}(\text{Alg}) + \cancel{3\lambda k} \le 3\,\text{Conn Cost}(k\text{-Median OPT}) + \cancel{3\lambda k}$$

$$\Rightarrow \quad \text{Conn Cost}(\text{Alg}) \le 3\,\text{Conn Cost}(k\text{-Median OPT}) \quad \boxtimes$$

Great, but maybe no λ is "good" where we open exactly k centers

As we keep increasing λ, Algo opens fewer centers.

Maybe discrete jump occurs

$k_1 > k$

$k_2 < k$

$\lambda_- < \lambda < \lambda_+$

$\lambda_- \approx \lambda \approx \lambda_+$ but algo jumps from opening k_1 centers ($> k$) to k_2 centers ($< k$).

Think of

$\lambda_- = \lambda_+ = \lambda$, say.
(limiting case)

(k_1 sol^n) $\rightarrow$ sol^n $= S_1$, Cost $= C_1$

(k_2 sol^n) $\rightarrow$ sol^n $= S_2$, Cost $= C_2$.

$$C_1 + 3\lambda k_1 \leq 3(\mathrm{OPT} + \lambda k) \quad \text{— (1)}$$

$$C_2 + 3\lambda k_2 \leq 3(\mathrm{OPT} + \lambda k) \quad \text{— (2)}$$

COMBINER Procedure

k_1 sol^n is cheap but infeasible
k_2 sol^n is feasible but expensive
{ but, their average is feasible & cheap }

Let $p = \dfrac{k - k_2}{k_1 - k_2}$.

Consider

p (1) $+ (1-p)$ (2), and see what it gives?

$p C_1 + (1-p) C_2 + 3\lambda\left[p k_1 + (1-p) k_2\right]$

$$pC_1 + (1-p)C_2 + 3\lambda\left[pk_1 + (1-p)k_2\right] \leq 3(OPT + \lambda k).$$

$$\underbrace{pC_1 + (1-p)C_2} + 3\lambda\left[\frac{kk_1 - k_1k_2 + k_1k_2 - kk_2}{k_1 - k_2}\right] \leq 3(OPT + \lambda k)$$

$$pC_1 + (1-p)C_2 + 3\cancel{\lambda k} \leq 3OPT + 3\cancel{\lambda k}$$

So, if we choose $p = \dfrac{k - k_2}{k_1 - k_2}$

(*) so that

$$p \cdot k_1 + (1-p)k_2 = k \text{ [wtd Avg is } k \text{]}$$

Then Corr. wt Avg of cost

$$pC_1 + (1-p)C_2 \leq 3 \cdot OPT$$

In argument above we used $\lambda_- = \lambda - d\lambda$ for (1) and $\lambda_+ = \lambda + d\lambda$ for (2)

but think of $\lambda_- = \lambda_+$ since
$d\lambda \to 0$.

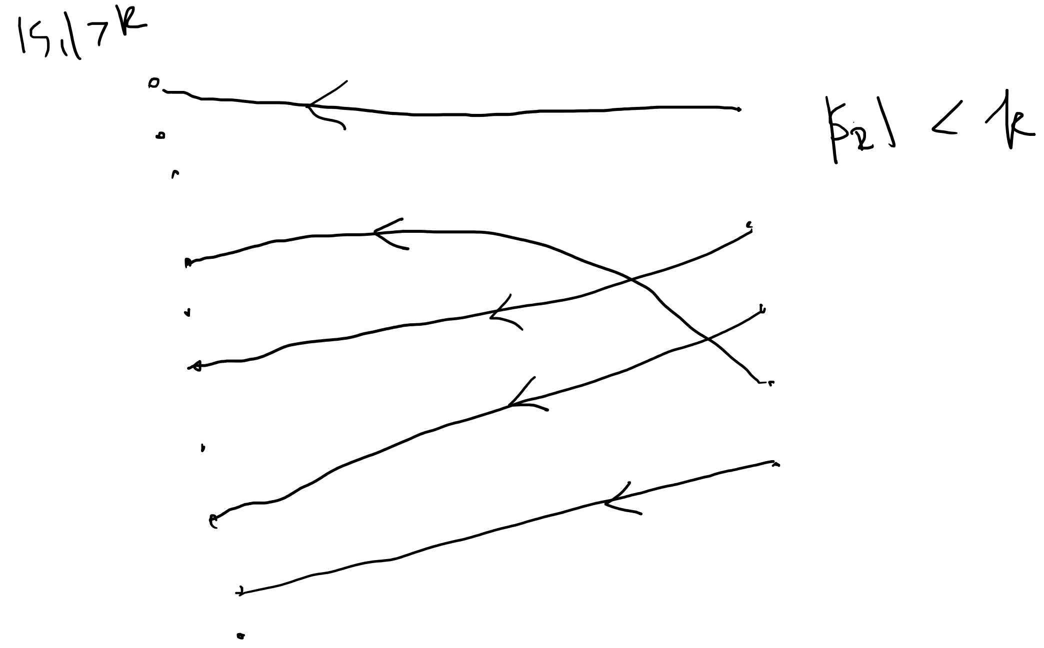
A Randomized Combiner Process :-

Idea :

Open centers in S_1 wp 'p' and
centers in S_2 wp '(1-p)' but
ensure that we open only
k centers wp 1.

If we open S_1 wp p and
S_2 wp $(1-p)$, then we are
infeasible wp p since
$|S_1| > k$.

We 'identify good back-ups' from
S_1 which we open instead
of all of S_1.



For each $i \in S_2$, let $\eta(i) \in S_1$ denote the closest point in S_1 to i.

(i.e) $\eta(i) = \underset{i' \in S_1}{\text{argmin}}\ d(i, i')$

Algo

Lets assume $\eta(i_1) \neq \eta(i_2)\ \forall\ i_1, i_2 \in S_1$

$\downarrow$

Proof works even if they collide, this is just to simplify the discussion.

COMBINING PROCEDURE :-

- With probability $(1-p)$, Open all of S_2
- Else, with prob p, we open $\eta(S_2)$.

Only open all mates

Step 1

After step 1, we open k_2 facilities with prob 1.

From rest of S_1 ($k_1 - k_2$ points)
Choose $k - k_2$ uniformly at random and open them. as centers.

$\Rightarrow$ After step 2, we open k centers w.p 1.

Sanity Check

Q: for $i \in S_2$, what is prob of i being selected?

Ans: $(1-p)$

Q: for $i \in S_1$, what is prob of i being selected?

Ans: p

Proof: if $i \in \eta(S_2)$ then it is p.

if $i \notin \eta(S_2)$ then

$$\Pr[i \text{ open}] = \frac{k-k_2}{k_1-k_2} = p.$$

LEMMA

Expected conn. cost of any pt j

$$\leq 2\left[p \cdot d_1(j) + (1-p)\, d_2(j)\right]$$

where d_1 & d_2 are j's cost in S_1 and S_2.

Corollary

Expected conn. cost of sol^n $\leq 2\left[p \cdot C_1 + (1-p) C_2\right]$

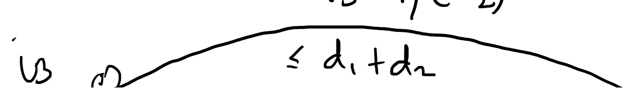
$\leq 2 \cdot 3 \cdot OPT$

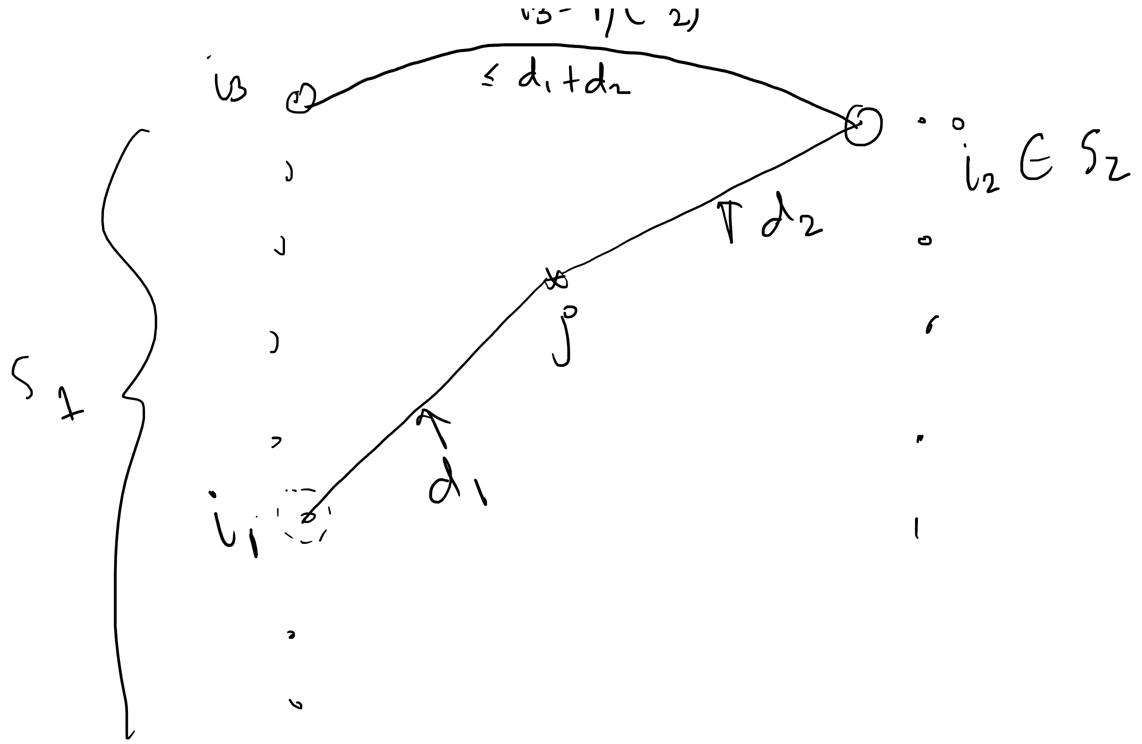
$= 6\, OPT$

$$\because \left.\begin{aligned} C_1 &= \sum_j d_1(j) \\ C_2 &= \sum_j d_2(j). \end{aligned}\right\}$$

Proof of Lemma.

$i_3 = \eta(i_2)$





Consider j and suppose its preferred conn. in S_1 is i_1, and S_2 is i_2.

and sps $i_3 = \eta(i_2)$ is the mate.

$$\boxed{d(i_2, i_3) \leq d_1 + d_2}$$

What does j connect to in our "mixed sol^n"?

If i_1 is chosen, j can connect to it.

(happens with probability p)

If i_1 is not ~~opened~~, we can check if i_2 is open.

"Check if i_2 is open"

If i_2 is not open, j can connect to i_3

for this "worst case analysis", lets assume that i_1 is not anybody's mate

$i_1 \notin \eta(S_2)$.

E[conn cost of j]

$= p d_1 + (1-p) \cdot \left[(1-p) \cdot d_2 + p(d_1 + 2d_2) \right]$

if i_1 is open

Open all of S_2

Open all of $\eta(S_2)$

$= p d_1 + (1-p) \left[d_2 + p d_1 + p d_2 \right]$

$= p d_1 + (1-p) d_2 + p(1-p)(d_1 + d_2)$

$\leq 2 p d_1 + 2(1-p) d_2$

Next Week

Johnson-Lindenstrauss Lemma.

Given n points X, distance function d and opening cost of $\lambda \geq 0$, choose a set S of centers to open and connect each point to nearest open center (incurring a cost of

$$d(j,S) = \min_{i \in S} d(i,j) \,)$$

to minimize

$$\lambda |S| + \sum_j d(j,S)$$

PRIMAL DUAL 3-(Lagrangean) Approximation.

If OPT soln has facility opening cost O^* and connection cost C^*,

and our solution has connection cost $\hat{C}$, and facility opening cost $\hat{O} = \lambda |\hat{S}|$ if we open centers at $\hat{S}$

α-approximation requires that

$$\hat{C} + \hat{O} \leq \alpha (C^* + O^*).$$

α-Lagrangean Approximation requires that

$$\boxed{\hat{C} + \alpha \hat{O} \leq \alpha (C^* + O^*)}$$

Useful to move from FL to k-Median by choosing λ parameter carefully.

LP(FL)

$$\text{Min} \sum_j \sum_i d(i,j) x_{ij} + \lambda \sum_i y_i$$

$\alpha_j)$ $\sum_i x_{ij} \geq 1 \quad \forall j$

$\beta_{ij})$ $y_i - x_{ij} \geq 0 \quad \forall (i,j)$

$x_{ij}, y_i \geq 0$

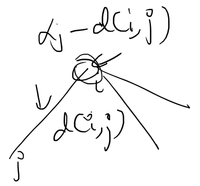
Dual (FL)

$$\text{Max} \sum_j 1 \alpha_j + \sum_{ij} 0 \beta_{ij}$$

$\alpha_j - \beta_{ij} \leq d(i,j)$

$\sum_j \beta_{ij} \leq \lambda$

$\alpha_i, \beta_{ij} \geq 0$



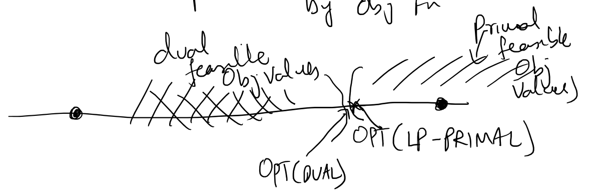
Think of α_j as amount of money j is raising for being connected.

β_{ij} = Amt of Money j is willing to put to opening a center at location i

$$\alpha_j = \beta_{ij} + d(i,j)$$

Recall: "take ≥ 0 linear combination, to maximize RHS, while ensuring all coefficients of primal variables are dominated by obj fn"

dual feasible values $\leq$ ////// $\leq$ Primal feasible Obj primal ... by obj fn"



Weak Duality THM

if (α, β) is dual feasible
and (x, y) is primal feasible,

then
$\text{Dual Cost}(\alpha,\beta) \leq \text{Primal cost}(x,y)$.

(ie) $\sum_j \alpha_j \leq \sum_i \sum_j d(i,j)\, x_{ij} + \lambda \sum_i y_i$

In particular,
(α,β) dual feasible $\Rightarrow$ $\boxed{\sum \alpha_j \leq OPT}$ (*)

$\Rightarrow$
if we find some good solution
with cost $\leq 3 \cdot \sum \alpha_j$, then
it will be a 3-approximation
due to (*)

PRIMAL - DUAL ALGORITHM ; STEP 1

Initialize $\hat{T} = \phi$ (no open facility)
Initialize $\alpha, \beta = 0$, and all clients are "unfrozen".

While ($\exists$ unfrozen clients)
- Increase $\alpha_j = \alpha_j + \varepsilon$ for suitably small ε for all unfrozen clients.
- If some $\alpha_j - \beta_{ij} = d(i,j)$ is tight for some $i \notin \hat{T}$, then also increase $\beta_{ij} = \beta_{ij} + \varepsilon$ to ensure (1) remains feasible
- if some facility constraint (2) becomes tight, (ie), $\sum_j \beta_{ij} = \lambda$, then add i to $\hat{T}$ (ie) open i temporarily) and freeze all clients j for which $\underbrace{\alpha_j - \beta_{ij} = d(i,j)}$ is tight.

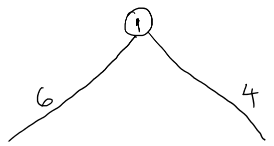
Recall Dual

Max $\sum_j \alpha_j$

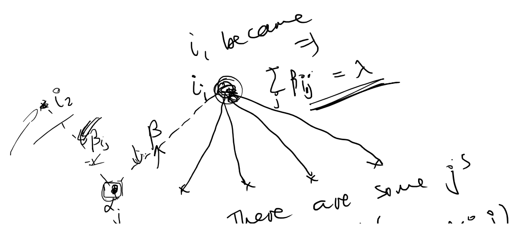
(1) $\alpha_j - \beta_{ij} \leq d(i,j) \quad \forall i,j$
(2) $\sum_j \beta_{ij} \leq \lambda \quad \forall i$
$\alpha_i, \beta_{ij} \geq 0$

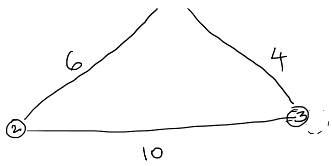
tight

Toy Example
3 points, distances marked in figure

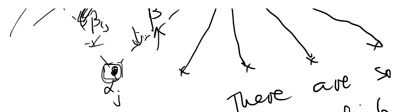


Suppose $\lambda = 12$





suppose ...



There are some j's for which $\alpha_j - \beta_{i,j} = d(i,j)$

None of these j's can $\uparrow \alpha_j$ (all frozen)

$\hat{I} = \phi$, all 3 pts are unfrozen.
all increase their α slowly @ same rate

At first step itself, $\alpha_i - \beta_{ii} = d(i,i)$ is tight because $d(i,i) = 0$

$\Downarrow$

$\beta_{i,i}$ increases jointly with α_i

Max $\Sigma \alpha_i$

$\alpha_1 - \beta_{11} \leq 0 \leftarrow$

$\alpha_1 - \beta_{21} \leq 6$

$\alpha_1 - \beta_{31} \leq 4 \leftarrow$ tight @ t=4

$\alpha_2 - \beta_{12} \leq 6$

$\alpha_2 - \beta_{22} \leq 0 \leftarrow$

$\alpha_2 - \beta_{32} \leq 10$

$\alpha_3 - \beta_{13} \leq 4 \leftarrow$ tight @ t=4

$\alpha_3 - \beta_{23} \leq 10$

$\alpha_3 - \beta_{33} \leq 0 \leftarrow$

tight @ t=6

Right at first step, β_{ii} grows like α_i

$\beta_{11} + \beta_{12} + \beta_{13} \leq 12 \leftarrow$

$\cancel{\beta_{31} + \beta_{32} + \beta_{33}} \leq 12$

@ t=7, $\beta_{11} = 7$, $\beta_{12} = 1$, $\beta_{13} = 3$

$\beta_{21} = 1$
$\beta_{22} = 7$
$\beta_{23} = 0$

$\beta_{31} = 3$
$\beta_{32} = 0$
$\beta_{33} = 7$

Think of α_j as money j is willing to raise to be connected

β_{ij} as the money j is willing to contribute to opening a facility at location i.

Lets think of $\varepsilon = 1$

t=0: All $\alpha = 0$, $\beta = 0$, $\hat{I} = \phi$

t=1 $\alpha_1 = 1$ $\alpha_2 = 1$ $\alpha_3 = 1$ $\beta_{ii} = 1$, $\hat{I} = \phi$

t=2 $\alpha_i = 2$, $\beta_{ii} = 2$, $\hat{I} = \phi$

t=3 $\alpha_i = 3$, $\beta_{ii} = 3$, $\hat{I} = \phi$

t=4 $\alpha_i = 4$, $\beta_{ii} = 4$, $\hat{I} = \phi$

t=5, $\alpha_i = 5$, $\beta_{ii} = 5$, $\beta_{13} = 1$, $\beta_{31} = 1$, $\hat{I} = \phi$

t=6: $\alpha_i = 6$, $\beta_{ii} = 6$, $\beta_{13} = 2$, $\beta_{31} = 2$, $\hat{I} = \phi$

t=7: $\alpha_i = 7$, $\beta_{ii} = 7$, $\beta_{13} = 3$, $\beta_{31} = 3$, $\beta_{21} = 1$, $\beta_{12} = 1$
$\hat{I} = \phi$

t=$7\frac{1}{3}$: $\alpha_i = 7\frac{1}{3}$, $\beta_{ii} = 7\frac{1}{3}$, $\beta_{13} = \beta_{31} = 3\frac{1}{3}$, $\beta_{21} = \beta_{12} = 1\frac{1}{3}$

facility 1 is tight

$\sum_j \beta_{ij} = 12$

$\Rightarrow$ can't increase any β_{ij} for all j

$\Rightarrow$ can't increase α_j for all j st $\beta_{ij} = d_{ij}$

$\Rightarrow$ freeze all such α_j.

(In this example, all 3 clients freeze at this point)

23/03

- Think of it as a continuous process (can be discretized easily).

- Few observations

- if $\beta_{ij} > 0$ for some (i, j) pair then j gets frozen due to i becoming frozen / added to $\hat{T}$ [tight constraint] $\alpha_j =$ (ie) $\beta_{ij} + d(i,j)$ $\sum \beta_{ij} = \lambda$.

(algo increases β_{ij} only when constraint becomes tight)

Next

if j

Then $d(i,j) \le \alpha_j$

Only those clients who can't increase their α anymore due to i freezing are frozen.

(ie) these clients have

$\alpha_j - \beta_{ij} = d(i,j)$ is tight

$\Rightarrow$ $\boxed{d(i,j) \le \alpha_j}$ since $\beta_{ij} \ge 0$

$\Rightarrow$ If we open all the facilities at $\hat{T}$ then the connection cost of all points (at the end)

Total Conn Cost to $\hat{T}$ $\le \sum \alpha_j \le$ dual OPT $\le$ OPT.

But What about total facility cost of opening $\hat{T}$?

Individually, each facility in $\hat{T}$ is a reasonable choice to open b/c $\sum_j \beta_{ij} = \lambda$

and so there are enough clients willing to share some money to open it.

But issue is collective,

· Some client could have $\beta_{ij} > 0$ to multiple facilities in $\hat{T}$.

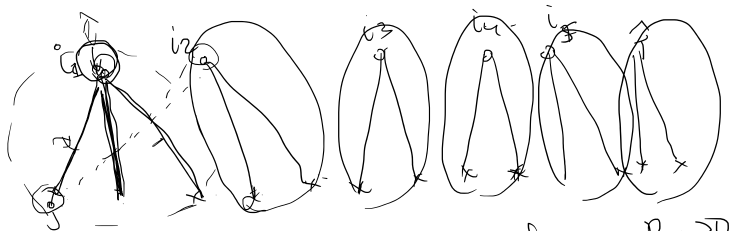
GOOD CASE

Sps $\forall$ points j, there is at most 1 facility $i \in \hat{T}$ for which $\beta_{ij} > 0$

Then I claim that overall it is a great solution

(10)

CONN Cost + $\lambda |\hat{T}| \leq$ OPT



from assumption

For each j, there is only one $\beta_{ij} > 0$ and it is the one which freezes j.

CONN Cost$(j, \hat{T})$ + $\lambda |\hat{T}|$

$$\sum_j \alpha_j = \sum_{\substack{\text{edges in} \\ \text{above graph}}} \left(d(i,j) + \beta_{ij} \right)$$

$$= \text{CONN cost}(j, \hat{T}) + \sum_{i \in \hat{T}} \left(\sum_{j \to i} \beta_{ij} \right)$$

Hence, CONN Cost + opening Cost $\leq \sum \alpha_j \leq$ OPT.

How to handle general case when some j is willing to put $\beta_{ij} > 0$ to multiple i's among $\hat{T}$

- Form a graph with vertices in $\hat{T}$
- Edges (i, i') iff $\exists j$ st $\beta_{ij} > 0$

& $\beta_{i'j} > 0$

- Pick a maximal independent set in this graph. $T_{ind} \subseteq \hat{T}$

Means No edges amongst T_{ind}

and $\forall$ $i \in \hat{T} \setminus T_{ind}$,

$\exists$ $i' \in T_{ind}$ (i, i') is edge.

$\hat{T}$ is all frozen centers for which $\sum_j \beta_{ij} = \lambda$

edge iff $\exists j$ st $\beta_{ij} > 0$ & $\beta_{i'j} > 0$

$\hat{T} = \{ i : \sum_j \beta_{ij} = \lambda \}$

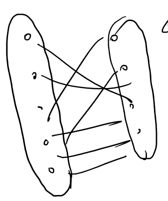
Given graph $G = (V, E)$

a set of vertices $I \subseteq V$

is "INDEPENDENT SET"

If $\forall$ $i_1, i_2 \in I$, there

is no edge $(i_1, i_2) \in G$



example, each side of a bipartite graph is an independent set.

In our case

T_{ind} is a "maximal Independent set"

(IE) can't add any other vertex to T_{ind} while preserving independence.

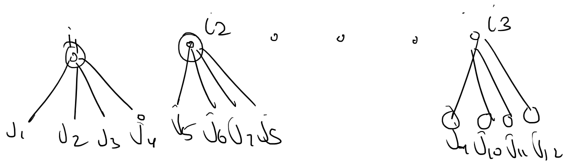
Overall Algo

- Run primal-dual process
- Build graph over $\hat{T}$ and choose maximal independent set
- Open facilities at T_{ind}, and connect all clients to nearest open facility.

ANALYSIS

Firstly, for T_{ind},

the total cost of opening T_{ind} is small.



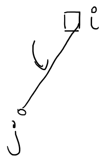
$$\lambda |T_{ind}| = \sum_{i \in T_{ind}} \left(\sum_{j : \beta_{ij} > 0} \beta_{ij} \right)$$

j's are disjoint !!

In particular,

$$\lambda |T_{ind}| \leq \sum_j \alpha_j$$

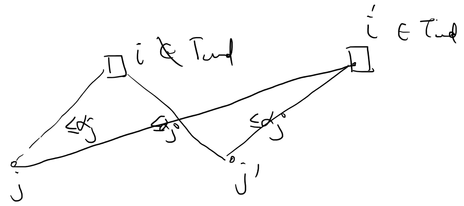
What about connection cost ?



Consider client j and suppose it is frozen b/c of i

& $i \in T_{ind}$

Then $d(i,j) \leq \alpha_j$ ✓



Recap 24-03-2021

Facility Location

Given points X, $|X| = n$, distance metric d, and opening cost $\lambda \geq 0$,

Open "centers/facilities" at $S \subseteq X$ to minimize

$$\sum_{j \in X} d(j, S) + \lambda |S|.$$

Formulated Primal-Dual and use dual to infer a solution.

Min – | Max $\sum \alpha_i$

Min
$$\sum_j \sum_i d_{ij} x_{ij} + \sum_i \lambda y_i$$
$$\sum x_{ij} \geq 1 \quad \forall j$$
$$y_i - x_{ij} \geq 0 \quad \forall i, j$$
$$x_{ij}, y_i \geq 0$$

Max
$$\sum \alpha_j$$
$$\alpha_j \leq d_{ij} + \beta_{ij} \quad \forall i, j$$
$$\sum_j \beta_{ij} \leq \lambda \quad \forall i$$

Intuition: α_j is money j is willing to invest for its overall happiness.

For any i, α_j breaks up into $d(i,j) + \beta_{ij}$

β_{ij}: Share j is willing to contribute for opening of center @ i.

Algo

① Keep raising all α_j (for unfrozen pts) ~~(at)~~ all points till to nearest open centers in T_{temp}

② If necessary, raise β_{ij} @ same rate.

③ If facility ($\sum_j \beta_{ij} = \lambda$) is tight for some i, freeze that i, open "Temp facility" (i.e.) add i to $\hat{T}$.
and freeze all clients which have tight constraint $\alpha_j = \beta_{ij} + d_{ij}$

④ When all clients frozen, form graph $G = (\hat{T}$, conflict edges)
(i.e.) (i, i') is an edge iff $\exists$ some j with $\beta_{ij} > 0$ & $\beta_{i'j} > 0$.

⑤ Pick a maximal independent set and output that soln (T_{ind})

For ~~Theorem~~ 3-Approx, we need to show: ~~3-Lagrangean~~ Approx.

$\hat{C}$ This Algo $\hat{O}$: $\le 3(OPT + FL)$

For $\hat{C}$ lagrangean approximation, we need to show total opening cost $\le \lambda |T_{ind}|$

$\hat{C}$ = total conn cost

$\hat{O}$

$$\boxed{\hat{C} + 3\hat{O} \le 3(OPT - FL).}$$

Proof

Consider T_{ind} and let us break up all the points in X into 2 sets:

G = good pts = $\{ j : \alpha_j - \beta_{ij} = d(ij)$ is tight for some $i \in T_{ind} \}$

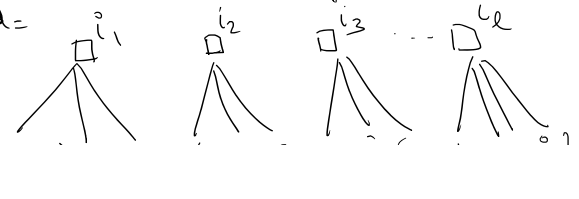
Notice: all pts j with $\beta_{ij} > 0$ for $i \in T_{ind}$ belong to G

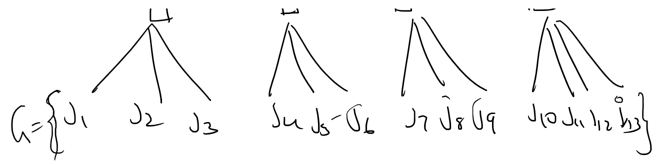
Moreover,

It can't happen that $\beta_{i_1 j} > 0$ & $\beta_{i_2 j} > 0$ for $i_1, i_2 \in T_{ind}$

If not, then there'll be an edge (i_1, i_2) so both can't be in independent set.

$T_{ind} = i_1 \; i_2 \; i_3 \; \ldots \; i_l$





Now,
$\forall j \in G$, let mate(j) be the $i \in T_{ind}$ for which $\beta_{ij} > 0$ (or if no such i exists, then pick any $i \in T_{ind}$ for which the constraint $\alpha_j - \beta_{ij} = d(i,j)$ is tight)

$\forall j$

$$\alpha_j = d(j, \text{mate}(j)) + \beta(\text{mate}(j), j)$$

Sum up over all $j \in G$

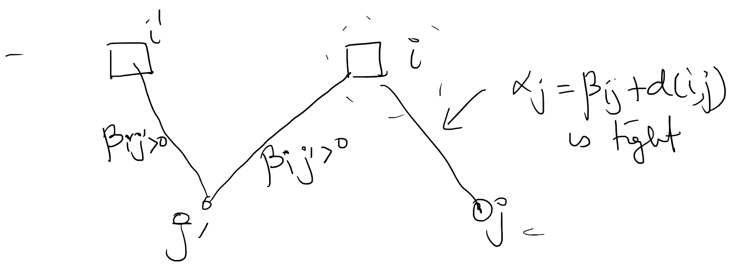
$$\sum_{j \in G} \alpha_j = \text{CONN COST}(G) + \underbrace{\sum_{j \in G} \beta(\text{mate}(j), j)}$$

$$= \quad '' \quad + \lambda |T_{ind}|$$

$$\boxed{3 \sum_{j \in G} \alpha_j = 3\,\text{CONN Cost}(G) + 3\lambda |T_{ind}|}$$

It remains to bound the bad points.

Lets look @ $j \notin G$
and let the facility which "froze it" be i



Clearly $i \notin T_{ind}$ else j would have been added to T_{ind}

$\Rightarrow \exists\ j', i'$ st

$$\beta_{i'j'} > 0, \quad \beta_{ij'} > 0$$

and $i' \in T_{ind}$

Claim:

① $d(i, j) \leq \alpha_j$

② $d(i, j') \leq \alpha_j$

③ $d(i', j') \leq \alpha_j$

$\Rightarrow \boxed{d(j, i') \leq 3\alpha_j}$

---

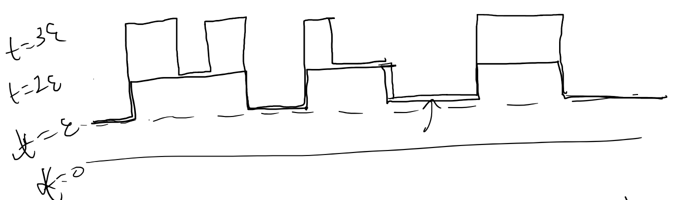
① is clear ($\alpha_j = \beta_{ij} + d(i,j)$ is tight). ✓

For ②

View the dual process as
"at time t, α_j for all unfrozen points $= t$"

and view $\beta_{ij} = \max(\alpha_j - d(i,j), 0)$

and if any $\sum_j \beta_{ij} = \lambda$ for some i,
freeze all j's for which
$\alpha_j = \beta_{ij} + d(i,j)$.



In this view, j froze at time α_j.
At this time, i gets frozen or was already frozen.
But definitely, since at this time, $\beta_{ij'} > 0$, j' must be frozen before j did
$\Rightarrow \alpha_{j'} \leq \alpha_j$

and because $\beta_{i'j} > 0$ & $\beta_{ij'} > 0$,

we get that

$d(i, j) \leq \alpha_{j'} \leq \alpha_j$

$d(i', j') \leq \alpha_{j'} \leq \alpha_j$

$\Rightarrow$ j has a good conn to T_{ind} of cost $\leq 3\alpha_j$

SUMMARY

$$3 \sum_{j \in G} \text{CONN Cost}(j) + 3\lambda |T_{ind}| \leq 3 \sum_{j \in G} \alpha_j$$

$$\sum_{j \notin G} \text{CONN Cost}(j) \leq 3 \sum_{j \notin G} \alpha_j$$

Overall

$$\text{Total CONN Cost} + 3\lambda |T_{ind}| \leq 3 \sum \alpha_j$$

$$\leq 3(\text{OPT-FL})$$

$\because \alpha$ is feasible dual soln.

Question :-

- We have a collection of N images $[1000 \times 1000]$ pixels.
- We are given a "query" image $[1000 \times 1000]$.
- Goal: quickly find out the "closest" image to the query image.

How?

High level Idea

- Transform each image into a vector in $1000^2 = 10^6$ dims.

$$d \left\{ \begin{array}{c} d \\ \begin{array}{|c|c|c|} \hline 0 & .2 & .1 \\ \hline .15 & .7 & .8 \\ \hline 0 & .5 & .4 \\ \hline \end{array} \end{array} \right. \rightarrow \overset{\longleftarrow D = d^2 \longrightarrow}{(0, .2, .1, .15, .7, .8, 0, .5, .4)}$$

- Same for query image
- Output the nearest vector to the query vector.

(say l_2 distance)

(i.e.) Given N vectors $v_1, v_2, \ldots, v_N \in \mathbb{R}^d$

and given query $q \in \mathbb{R}^d$

find $\boxed{\arg\min_{i=1}^{N} \|v_i - q\|_2^2}$

(APPROX.)

NEAREST NEIGHBOR SEARCH

Given N "base" vectors in D dimensions, pre-process and build a data structure.

So that when queries are presented, we can quickly retrieve the (approximate) closest vector to the query

---

Naive solution

When query is presented, compare distances to all base points and output the closest.

\# operations per query : $3 \cdot N \cdot D + N$

D subtractions, Multiplications, Additions per point

find MIN.

In our example $D = 10^6$.

Today's lecture

generic "dimension reduction" technique
which can bring down $10^6 \rightsquigarrow 10^3$
with very little "error".

JOHNSON - LINDENSTRAUSS LEMMA

Given n vectors in $\mathbb{R}^d$, $v_1, v_2, \ldots, v_n$
there is a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$
such that

$\forall\ u_i, u_j$

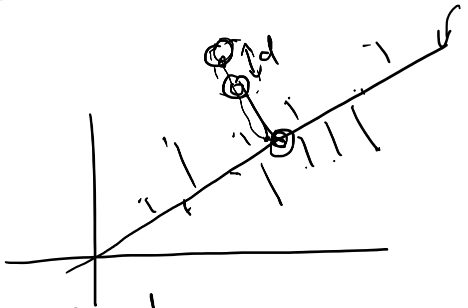
$$(1-\varepsilon) \le \frac{\|f(u_i) - f(u_j)\|_2}{\|u_i - u_j\|_2} \le (1+\varepsilon)$$

with $k = \Theta\left(\frac{\log n}{\varepsilon^2}\right)$.

Moreover we can find this efficiently
with good probability.

Goal for today

Idea: PCA ?
- Unclear
- It's more of a
- global error reduction



- we want pairwise preservation

Idea 2: what about just picking k random coordinates?

$v = (x_1, x_2, \dots, x_d)$ then $f(v) = (x_{i_1}, x_{i_2}, \dots x_{i_k})$

where $i_1, i_2, \dots i_k$ are randomly chosen from $[d]$.

Same $i_1, i_2, \dots i_k$ are used for all vectors.

[sample with replacement, to make analysis easy]

Good News

for any u and $v \in \mathbb{R}^d$

$$E\left[\|f(u) - f(v)\|^2\right] = \frac{k}{d}\|u-v\|^2$$

good, b/c then we can think of

$$\hat{f}(u) = \sqrt{\frac{d}{k}}\left(u_{i_1}, u_{i_2}, \dots u_{i_k}\right)$$

and then we'll have

$$E\left[\|\hat{f}(u) - \hat{f}(v)\|^2\right] = \|u-v\|^2.$$

Proof of Good News

Fix u, v and let $z = u - v$

$$\|z\|^2 = \sum_{i=1}^{d} (u_i - v_i)^2 = \sum_{i=1}^{d} z_i^2$$

Now, consider i_1 and focus on

$$\underset{i_1}{E}\left[(u_{i_1} - v_{i_1})^2 \right] = \sum_{l=1}^{d} \underbrace{\Pr[i_1 = l]}_{} \cdot (u_l - v_l)^2$$

$$= \frac{1}{d} \cdot \sum_{l=1}^{d} (u_l - v_l)^2$$

$$= \frac{1}{d} \|u - v\|_2^2$$

$$\Rightarrow E\left[\|f(u) - f(v)\|_2^2 \right] = \sum_{l=1}^{k} E\left[(u_{i_l} - v_{i_l})^2 \right]$$

$$= \frac{k}{d} \|u - v\|_2^2$$

If we can show some "concentration inequality" that $\|f(u) - f(v)\|$ is close to its average whp, then this scheme works!

and then union bound for all $1 \le i < j \le n$.

Q: Is this likely to have concentration?

A: Sadly, no ☹

Here's an example :-

$v_1 = (1, 0, 0, \dots 0)$

$v_2 = (0, 1, 0, \dots 0)$

$\vdots$

$v_n = (0, 0, \dots, 1)$.

Unless we pick one of 1 or 2 coordinate in f,

$\|f(u_1) - f(u_2)\| = 0$

$\Rightarrow$ even if $k < n-1$, we will always make huge error on some pair.

real distance $= \sqrt{2}V$

"our imagined" distance $= 0$

badness because most coordinates are same, few coords give all the distance.

How to "spread" this over all coords?

TRICK:

- Nothing sacred about standard basis.

Choose a random rotation $R \in \mathbb{R}^{d \times d}$

first rotate the data

$\boxed{\begin{array}{l} V_i \longrightarrow RV_i \quad \text{(still in d-dim)} \\ \text{then sample } k \text{ coords from } RV_i \end{array}}$

- Intuition, can be made rigorous
- can also be simplified greatly using gaussians.

$$\boxed{f(u) = \frac{1}{k} \underbrace{\overbrace{\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1d} \\ \vdots & & & \\ g_{k1} & g_{k2} & \cdots & g_{kd} \end{bmatrix}}^{d}}_{G} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u \end{bmatrix}}$$

09/04/2021

$$f(u) = \lambda \begin{array}{c} \\ k \updownarrow \end{array} \overbrace{\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1d} \\ g_{21} & \cdots & & g_{2d} \\ \vdots & & & \\ g_{k1} & & \cdots & g_{kd} \end{bmatrix}}^{\longleftarrow d \longrightarrow} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_d \end{bmatrix}$$

G = random $k \times d$ gaussian matrix where each g_{ij} is an independant gaussian r.v with mean 0 & variance 1.

i.e) $g_{ij} \sim N(0, 1)$.

λ will be chosen suitably later.

<u>THEOREM</u>

For suitable λ and k, this mapping preserves <u>all</u> pairs of distances upto $(1 \pm \varepsilon)$ factor.

<u>Q</u> How do we prove this?

We'll fix a pair of vectors u_i, u_j and let $z = u_i - u_j$

Next we show that

$$\Pr_{\text{random choice of } G}\left[\frac{\|f(u_i) - f(u_j)\|_2^2}{\|u_i - u_j\|_2^2} \notin (1-\varepsilon, 1+\varepsilon)\right] \leq \frac{1}{n^2}.$$

Then we just do union bound over all pairs u_i, u_j.

$$\Pr_G \left[\exists\, i, j \text{ which is not preserved to } (1 \pm \varepsilon) \right] \leq \frac{\binom{n}{2}}{n^2} \leq \frac{1}{2}.$$

With prob $\frac{1}{2}$, experiment was successful (ie) all pairs preserved!

If not, just repeat ☺

Need to show that a single pair is preserved w.h.p.

Fix u & v and let $z = u - v$

$$f(u) = \lambda \cdot G \cdot u$$
$$f(v) = \lambda \cdot G \cdot v$$

$$f(u) - f(v) = \lambda \cdot G(u - v) = \lambda G z = f(z).$$

Need to show that

$$1 - \varepsilon \leq \frac{\|f(z)\|_2^2}{\|z\|_2^2} \leq 1 + \varepsilon \quad \text{with prob} \geq 1 - \frac{1}{n^2}$$

Let $z = (z_1, z_2, \ldots, z_d)$.

and consider $\hat{z} = \frac{z}{\|z\|}$.

Then $z = \|z\| \cdot \hat{z}$.

Moreover,

$$f(z) = \lambda \cdot G \circ z = \lambda \|z\| \cdot G \hat{z}$$
$$= \|z\| \cdot f(\hat{z}).$$

So, it suffices to show that

$$(1-\varepsilon) \le \| f(\hat{z}) \|_2^2 \le (1+\varepsilon) \quad \text{w.p} \ge 1 - \frac{1}{n^2} \quad (*)$$

$$\Rightarrow \quad \|z\|^2 (1-\varepsilon) \le \|f(z)\|_2^2 \le (1+\varepsilon)\|z\|^2$$

Consider $\hat{z} = (\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_d)$

we know $\|\hat{z}\|_2^2 = 1 \Rightarrow \sum \hat{z}_i^2 = 1.$

Let's look at $f(\hat{z}) = \lambda \cdot G \circ \hat{z}$

$$= \lambda \begin{bmatrix} \sum_{j=1}^{d} g_{1j} \hat{z}_j \\ \sum_{j=1}^{d} g_{2j} \hat{z}_j \\ \vdots \\ \sum_{j=1}^{d} g_{kj} \hat{z}_j \end{bmatrix}$$

Each entry in $f(\hat{z})$ looks like

$$\lambda \cdot N\left(0, \sum_{j=1}^{d} \hat{z}_j^2\right) = \lambda N(0,1).$$

Moreover because g_{ij} and $g_{i'j'}$ are independent for all

independent for all
i, j, i', j',

these entries are themselves independent!

$$f(\hat{z}) = \lambda \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_k \end{pmatrix} \quad \text{where } F_i \sim N(0,1).$$

What is $E\left[\|f(\hat{z})\|^2\right] = \lambda \sum_{i=1}^{k} E\left[F_i\right]^2 = \lambda \cdot k.$

We want $f(\hat{z})$ to be close to 1.

$\Rightarrow$ set $\boxed{\lambda = \frac{1}{k}}$

$\Rightarrow$ $E[\underbrace{\|f(\hat{z})\|^2}] = 1.$

Let $Y = k\|f(\hat{z})\|^2$ be the random variable.

$$Y = \underbrace{\sum_{i=1}^{k} F_i^2}$$

Y is the sum of many independent random variables.

Moreover each F_i is a gaussian $N(0,1)$

$\boxed{\chi^2\text{-distribution}}$

χ^2 - distribution

{ Y actually sharply concentrates around its mean. }

Try to prove this using ideas from chernoff bounds proof

More or less,

Y behaves like $N(\mu(Y), \sigma^2(Y))$

$$E[Y] = \sum E[F_i^2] = k.$$

$$Var[Y] = \sum_{i=1}^{k} Var[F_i^2]$$

$$Var[F_i^2] = E[F_i^4] - E[F_i^2]^2$$

$$= E[F_i^4] - 1.$$

{ $E[F_i^4] = O(1)$ [constant] for gaussian distribution.

$\Rightarrow Var[Y] = Ck$ for constant C.

{ Y behaves like $N(k, Ck)$. }

Very big cheat, crude approx.

Basically, we'll use chernoff-like bounds on Y.

tail bounds on Y.

$$\Pr\left[\, |Y - E[Y]| \geq t \,\right] \leq \exp\left(\frac{-t^2}{\sigma^2}\right)$$

Can prove this type of inequality for Y.

$E[Y] = k$

$\sigma^2(Y) = c \cdot k$.

Set $t = \varepsilon \cdot k = \varepsilon E[Y]$

$$\Pr\left[\, Y \notin (1-\varepsilon, 1+\varepsilon) E[Y] \right] \leq \exp\left(\frac{-\varepsilon^2 k^{\cancel{2}}}{c \cdot \cancel{k}}\right)$$

$$= \exp\left(\frac{-\varepsilon^2 k}{c}\right).$$

So just set $k = \dfrac{2c \cdot \ln n}{\varepsilon^2}$

$$\Pr(\text{bad event}) \leq \exp\left(\frac{-\cancel{\varepsilon^2} \cdot 2\cancel{c} \ln n}{\cancel{c}\, \cancel{\varepsilon^2}}\right)$$

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

Whenever Y lies $\in [1-\varepsilon, 1+\varepsilon] E(Y)$

$\Longleftrightarrow \quad f(\hat{z}) = \frac{1}{k} Y$ lies in $[1-\varepsilon, 1+\varepsilon]$.

$\Longleftrightarrow \quad f(\hat{z}) = \frac{1}{k} \cdot Y$ lies in $[1-\varepsilon, 1+\varepsilon]$.

$\Rightarrow \Pr\Big(f(\hat{z}) \notin [1-\varepsilon, 1+\varepsilon] \Big) \le \frac{1}{n^2}$ for

$k = \frac{2c \ln n}{\varepsilon^2}$

and $\lambda = \frac{1}{k}$.

Locality Sensitive Hashing

12 April 2021 10:00

(c, r) - Approximate Near Neighbor search.

Motivation

- Again vectors in high dimensional space
- look to retrieve "closest" vector to given query vector.

Concrete Example :-

N documents, N is very large (all in english)

Represented as a $\{0,1\}^D$ vector for D very large.

$D =$ all distinct english words Ordered as $\{w_1, w_2, \ldots w_D\}$

$v = (v_1, v_2, \ldots, v_D)$

where $v_i = 0$ if i^{th} word is not in document

$= 1$ if i^{th} word is in doc.

Goal: build a data structure / Algorithm s.t when query vector q arrives, we quickly retrieve any nearby base vector.

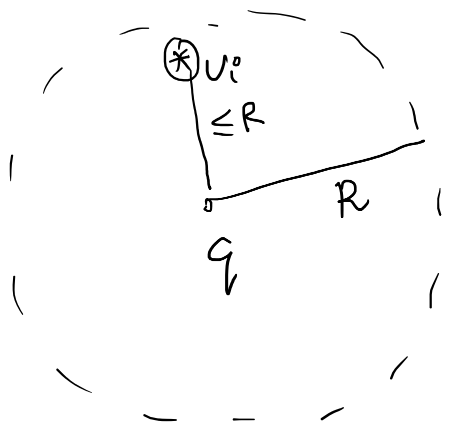
$$q = (q_1, q_2, \ldots, q_D)$$

Given N base vectors $v_1, v_2, \ldots, v_N$ and $q = (q_1, q_2, \ldots, q_D)$ arrives, and given parameter R, find any base vector which are at distance $\leq R$ from query q, if they exist.

Distance = Hamming distance

$$= |u - v|_1 = \sum_{j=1}^{D} |u_i - v_i|$$

$$= \#\text{indices where they differ}$$



<u>Naive approach</u>

- Compare distances to all base vectors
Output the closest.

Query Time = $O(\underline{N}D)$

Space of data structure = ND bits
(store all base vectors).

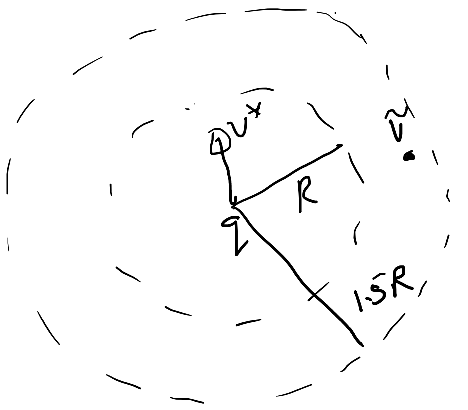
<u>QN</u>:

Can we get sub-linear dependence on N?
eg, even if we bring query time
from ND to $\sqrt{N} \cdot D$, that is
really significant in real life
N can be 10^9, 10^{12} even!!

Today + Tomorrow:
Can do such a thing if we're OK
with <u>Approximate</u> Near Neighbor search

Given q, target R,
- Algo can output <u>NO</u> if no base
point is distance $\leq R$
from q.

- Output <u>any</u> base point at distance $\leq CR$ from q, for some $C > 1$.



[Algo can output $\tilde{v}$ instead of v^*]

In this case $C = 1.5$

C is fixed up-front, is like "approximation factor of algo".

<u>THM</u> [Indyk-Motwani '90s]

(*) For above problem, can design C-Approx NNS algorithm with space $O(N^{1+1/c} + ND)$ but query time $= O(N^{1/c} D)$.

<u>Idea</u>: Design very generic technique called "LOCALITY SENSITIVE HASHING".

Imagine the following type of search Algo:-

Compute a hash function

Compute a hash function

$h : \{0,1\}^D \to [K]$ eg. where $K \ll N$

think of $K = \sqrt{N}$.

It maps each document vector to one of K buckets.

Similarly it can map query to a bucket.

<u>Algo</u>: ONLY look in the query bucket and output the closest base vector!

<u>Query time</u>: $\approx \frac{N}{K} \cdot D \approx \sqrt{N} \cdot D$ if $K = \sqrt{N}$.

h is a random hash
$\Rightarrow$ each bucket has $\approx N/K$ pts.

<u>Issue</u>

No reason why query's nearest neighbor is hashed to same bucket as q.

<u>Idea</u>:

What if hash fn 'h' is locality sensitive?

(ie) nearby points are in same bucket more often than far away points!

than far away points!

Then Maybe such an algo would work!

LSH

A hash family $\mathcal{H} = \{h_1, \ldots h_M\}$ is a (C, R, P_1, P_2) LSH for $P_1 > P_2$ if

for any 2 vectors x and y,

① $\Pr_{h \sim \mathcal{H}} [h(x) = h(y)] \geq P_1$ if $d(x,y) \leq R$

② $\Pr_{h \sim \mathcal{H}} [h(x) = h(y)] \leq P_2$ if $d(x,y) > CR$

Property ② ensures that buckets are small on avg, and

① ensures that near neighbors of query are in same bucket as q.

$\boxed{\text{From } (C, R, P_1, P_2) \text{ LSH to } (C,R) \text{ Algo.}}$

Idea:

① "boost" the gap between P_1 and P_2 by taking

(1) Boost the gap p, ...

by taking

$$g(x) = [h_1(x), h_2(x), \dots h_k(x)]$$

where $h_1, h_2, \dots h_k$ are k independent samples from $\mathcal{H}$.

What does g satisfy:

$\Pr[g(x) = g(y)] \geq p_1^k$ if $d(x,y) \leq R$

$\Pr[g(x) = g(y)] \leq p_2^k$ if $d(x,y) > cR$.

- This has driven down p_2^k $\Downarrow$
- Now, this also has driven down good case collision (p_1^k)
- Fix this by using L different hash functions

$g_1, g_2, \dots, g_L$ {independently for each g_i}

<u>Overall Algo</u> :-

(1) Compute $g_i = (h_{i1}, h_{i2}, \dots h_{ik})$ for $i = 1, 2, \dots L$.

(2) ... $g_i(\cdot)$ ... all $1 \leq i \leq L$ and

② Compute $g_i(v)$ for all $1 \le i \le L$ and all base vectors.

When query arrives

Compute $g_1(q)$, ..., $g_L(q)$.

For $i = 1, 2, \ldots L$

- Look at all base vectors with $g_i(v) = g_i(q)$.
- If we find any at distance $\le cR$, output and terminate

Need to analyze time and space complexity.

Space Complexity : $N \cdot D + O(N \cdot K \cdot L)$

hash values.

Query Time :-

$LK + L * \Big(E[\#$ far away pts which fall into bucket of $q] \cdot D$

hash fns g

dimension

Sort of "wasteful computation"

sort of
"wasteful computation"

$$\leq L \cdot N \cdot p_2^k \cdot D$$

$$E[\text{query time}] \leq L N p_2^k \cdot D$$

Now its all about parameter choosing

Choose K ~~st~~ $p_2^k = \frac{1}{N}$.

(ie)

$$K \log \frac{1}{p_2} = \log N$$

$$K = \frac{\log N}{\log 1/p_2}.$$

Next choose L so that q's near-neighbor is in the bucket with good probability

---

Let v^* be q's neighbor within distance $\leq R$ (if any).

What is the probability that it falls into one of the L buckets of q?

$Pr[v^*$ falls in q's bucket in one of the L hash fns$]$

$$\geq 1 - \prod_{i=1}^{L} (1 - p_1^k)$$

$$= 1 - (1 - p_1^k)^L$$

$(1-\varepsilon)^L$

$\approx 1 - \varepsilon L$

when ε is small

Need to formalize $\rightarrow \approx \underbrace{L \cdot p_1^k}$

Set L st

$$L \cdot p_1^k = 1$$

to ensure good success probability

$$L = \frac{1}{p_1^k}$$

$$= \left(\frac{1}{p_1}\right)^{\frac{\log N}{\log(1/p_2)}}$$

$$\boxed{L = N^{\frac{\log\left(\frac{1}{p_1}\right)}{\log\left(\frac{1}{p_2}\right)}}} \leftarrow \rho$$

Parameter $\rho = \frac{\log(1/p_1)}{} < 1$ is

$\log(1/p_2)$

the important parameter of LSH.

Summary

$$k = \frac{\log N}{\log 1/p_2} \; ; \quad L = N^{\rho}$$

Expected Query Time

$$= kL + O(\underbrace{N \cdot L \cdot p_2^k}\cdot D)$$

$$= \frac{\log N}{\log 1/p_2} \cdot N^{\rho} + O(N^{\rho} \cdot D).$$

{great if $\rho < 1$} ☺

Success Prob

if $\exists$ a near nbr, we find some approx NN with prob

$$\geq \left(1 - \left(1 - p_1^k\right)^L\right)$$

$$\geq 1 - e^{-p_1^k \cdot L}$$

$(e^x \geq 1 + x)$

$$\geq (1 - 1/e).$$ ∎

$(e^x \geq 1+x)$ $\geq (1 - 1/e)$. ∎

<u>THM</u>:

(C, R, P_1, P_2) LSH with $\rho = \dfrac{\log 1/P_1}{\log 1/P_2}$

implies a randomized (C, R)-ANNS algorithm

with success prob $\geq (1 - 1/e)$ of finding a near nbr @ distance $\leq CR$ (if there exists some nbr at dist $\leq R$)

and expected query processing time

$$= O\left(N^\rho \left(\frac{\log N}{\log 1/P_2} + D\right)\right)$$

$\leftarrow$ constant, can ignore

$\Rightarrow$ LSH type hash fns are good as long as ρ is <u>small</u> !! :)

<u>QN</u>

for vectors in $\{0,1\}^D$ with $d(x,y) = \sum_{i=1}^{D} |x_i - y_i|_1$ as distance,

What is a good LSH ?

Want

- If x, y are close by $(\le R)$
  $\Pr[h(x) = h(y)]$ is large.
- If x, y are far apart $(\ge cR)$
  $\Pr[h(x) = h(y)]$ is small.

Dirt Simple Idea

Pick a random coordinate d from $\{1, 2, \ldots, D\}$.

$$\boxed{h_d(x) = x_d}$$

$$\mathcal{H} = \{h_1, h_2, \ldots h_D\}$$

P_1, P_2 ANALYSIS

If $d(x,y) \le R$, (ie) they differ in at most R places,

$$\Pr_d[h_d(x) = h_d(y)] \ge \underbrace{\left(1 - \frac{R}{D}\right)}_{P_1}.$$

$$\Pr_d[h_d(x) = h_d(y)] \le \underbrace{\left(1 - \frac{cR}{D}\right)}_{P_2}.$$

$\leq d \cdot \underbrace{\qquad}_{P_2}$

if $d(x,y) \geq CR$

What is the ρ of this hash fn.

$$\rho = \frac{\log(1/P_1)}{\log(1/P_2)}$$

$$= \frac{\log\left(1/\left(1-\frac{R}{D}\right)\right)}{\log\left(1/\left(1-\frac{CR}{D}\right)\right)}$$

Need to be formalized

Assuming $R << D$,

we'll make some simplifications

a) $\frac{1}{1-\varepsilon} \approx 1+\varepsilon$

b) $\log(1+\varepsilon) \approx \varepsilon$

if $\varepsilon << 1$.

Hence $\rho \approx \frac{R\,\cancel{D}}{\cancel{D}\cdot CR} = \frac{1}{c}$.

Using this hash family gives us the (C, R) - Near Neighbor search problem with

space $O(N^{1+1/C} + ND)$ and

Query time $O(\underbrace{N^{1/C}} \cdot (D + \log N))$.

$\Downarrow$

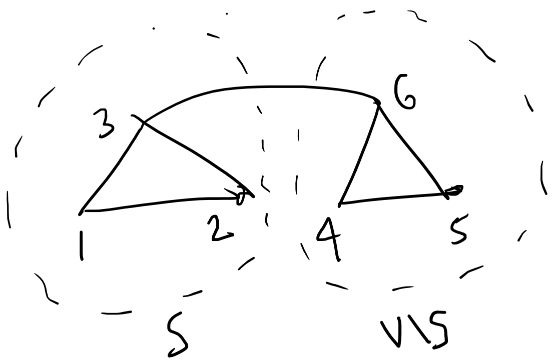
· Represents a significant milestone for "Nearest vector search" and is used in variety of applications today !!

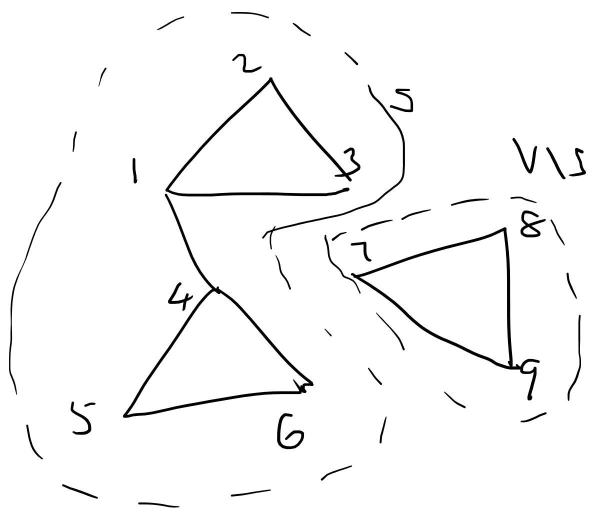
Min-Cut, Max-Cut and SDPs

19 April 2021 10:03

MIN CUT

{ Given $G = (V, E)$ unweighted, undirected graph, partition into $(S, V \setminus S)$ to minimize # edges "cut" (or) crossing $(S, \bar{S})$ }



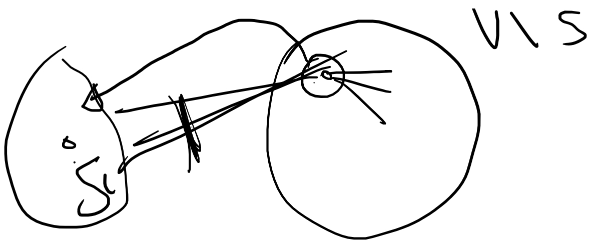


Q: How to solve it?

Idea ①: LP formulation?
↳ We'll get to this

Idea ②: Randomly put each vertex in S or $\bar{S}$
↳ UNCLEAR if any good!

Idea ③
Greedy: Include v from $V \setminus S$ with Max # edges to S.



Idea (4)

"Local Search"

If by shifting any v from S to $V \setminus S$ (or vice versa) the cut improves, do it.

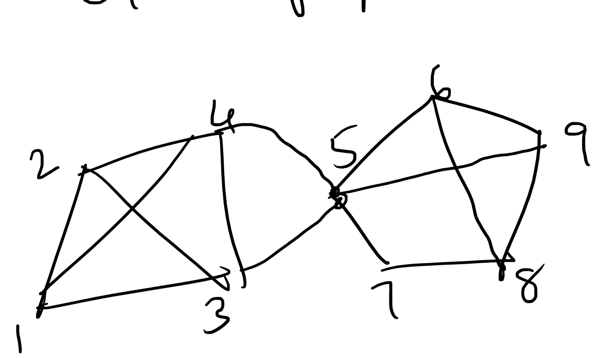
Idea (5)

Karger's Randomized contraction Algorithm :-

OPERATION

Graph Contraction ?

G = graph $G = (V, E)$



Let u & v be 2 vertices

Let u & v be 2 vertices

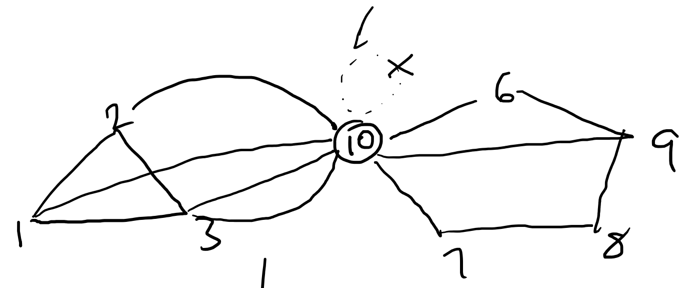
$G \setminus \{u, v\}$ = contraction of G by "fusing u & v"

Let's say we contract 4 & 5

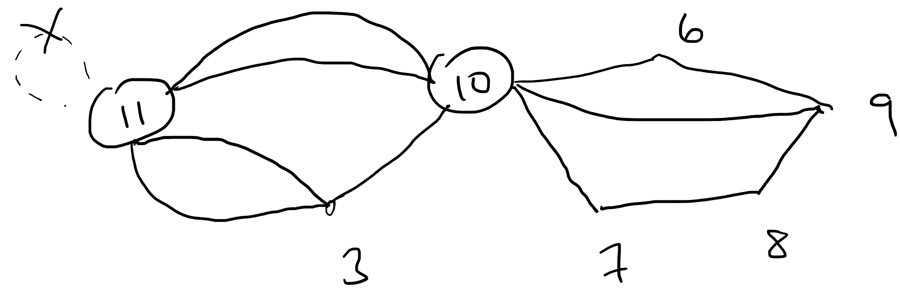
Let 10 be the contracted/fused vertex.

↓

Remove self-loop if any.



contract 1 & 2



Impt: retain duplicate/parallel edges, remove self-loop (if any)

KARGER ALGO

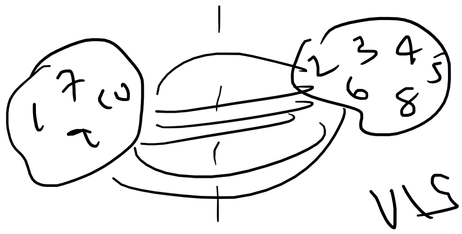
if G = 2 vertices = {u, v}

Min cut

= u on one side
v on other side

All parallel edges b/w u & v will be cut.

If $|G| > 2$ vertices:
Pick a uniformly random edge
and contract it !



S $\quad$ $V \setminus S$

← output of random contraction algorithm

THM:
with reasonable prob $\left(\geq \frac{1}{n^2}\right)$,
the resulting cut is MINIMUM !!

Overall
repeat $\Omega(n^2 \log n)$ times and pick the best

↓
Min CUT is in P [poly-time solvable]
Let's look @ LP formulation [idea ①]

<u>Variables</u>

$X_u = 0/1$ to denote if $u \in S$ or $V \setminus S$

What about edges? How to determine if edge is cut or not?

Y_{uv} for every edge $(u,v) \in E$

<u>Want</u> $Y_{uv} = 0$ if e is not cut
$\quad\quad\quad 1$ if e is cut.

$$\text{Min} \sum_{(u,v) \in E} y_{uv}$$

$$0 \le X_u \le 1 \qquad \forall (u,v)$$

Want to encode
$Y_{uv} \geq \mathbb{1}\{X_u \neq X_v\}$
with linear constraint

<u>Attempt</u> $Y_{uv} \geq |X_u - X_v|$.

is Actually a linear constraint because it is equivalent to

$$\begin{cases} Y_{uv} \geq (X_u - X_v) \\ Y_{uv} \geq -(X_u - X_v) \end{cases}$$

$$\begin{cases} y_{uv} \geq (x_u - x_v) \\ y_{uv} \geq -(x_u - x_v). \end{cases}$$

$\boxed{LP}$ $\quad \text{Min} \sum_{(u,v)\in E} y_{uv}$

$$\left.\begin{array}{l} x_u \geq 0 \\ x_u \leq 1 \end{array}\right\} \forall u \in V$$

tries to capture $y_{uv} \geq \mathbb{1}(x_u \neq x_v)$.

$$\left.\begin{array}{l} y_{uv} \geq (x_u - x_v) \\ \quad\;\; \geq -(x_u - x_v) \end{array}\right\} \forall u, v.$$

Lemma ①

LP is a valid relaxation of Min Cut problem.

$\Rightarrow$ if (x^*, y^*) is an optimal sol^n, then $\sum_{uv\in E} y^*_{uv} \leq \text{OPT(Min Cut)}$.

Now, we'll need to "round" it, since it might be fractional.

Idea ① Use threshold (0.5)

if $x_u \geq 0.5$, put in S

< 0.5 put in $V \setminus S$

Idea ② Think of x_u as probability of being in S.

We'll use Idea ③, a combination of ① & ②.

Algo:

① Choose random α between 0 & 1.

② For all $u \in V$,
put u in S if $x_u^* \geq \alpha$.

Need to show
$E[\#\text{ cut edges}]$ is small.

let

$Z_{u,v}$ = random variable
= 1 if u, v is cut
= 0 otherwise.

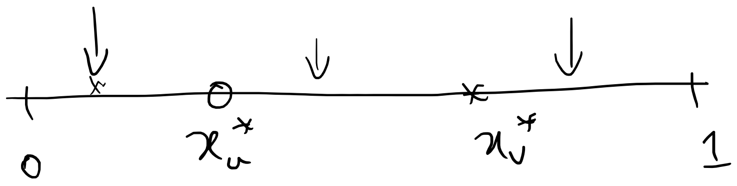
$E[\#\text{ edges cut}]$

$= E\left[\sum_{u,v \in E} Z_{uv}\right]$

$= \sum E[Z_{uv}]$ ← linearity of

$(u,v) \in E$ Expectation

What's $E[Z_{uv}]$?



$$E[Z_{uv}] = \Pr[(u,v) \text{ is cut}]$$

$$= \Pr[u \,\&\, v \text{ are on diff sides}]$$

$$\underset{\alpha}{=} \Pr[\alpha \in (x_u^*, x_v^*)]$$

$$= |x_u^* - x_v^*|.$$

$$\leq y_{uv}^* \quad (\text{from LP constraint}).$$

$$\Rightarrow \underset{\alpha}{E}[\#\text{ edges cut}] \leq \sum_{(u,v)\in E} y_{uv}^* \leq \text{OPT(Min Cut)} \quad \boxed{\text{☒}}$$

$\Rightarrow$ It's random, can we make it determinisi[...]

OPT = 5

Algo output $(S_1, V \setminus S_1)$ w.p. 0.25.   4

$(S_2, V \setminus S_2)$ w.p. 0.5   6

~~CANT~~ ...   4

~~CAN'T HAPPEN~~! $(S_3, V \setminus S_3)$ w.p. 0.25 4

$\Rightarrow$ All events have to yield value 5

Issue Pointed Out:

All x_u can be the same

LP will have value 0

How to fix this?

ANS:

"Guess 2 vertices u_L, u_R in OPT Min Cut" which are on different sides,

and enforce $x_{u_L} = 0$, $x_{u_R} = 1$.

in the LP

Solve many different LPs with different choices of u_L & u_R and output the non-degenerate one

TOMORROW:

Max Cut, Max # edges crossing cut.

Max Cut

20 April 2021 09:03

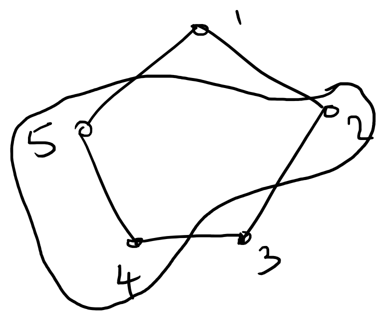
Min Cut is in P (exact polytime algorithm)

- What about MaxCut?
- Ans: NP-complete, resort to Approximation Algorithms

Qn: Given $G = (V, E)$, partition $V = S \dot\cup \bar{S}$ such that $|E(S, \bar{S})|$ is maximum.

where $E(A, B) = \{ (u, v) \in E \text{ s.t. } u \in A, v \in B \}$ for disjoint sets A, B

Examples



What is the Max Cut?

edges cut $= 4$

- Why is this Max Cut?

Because if $\exists$ cut with value 5, then it cuts all edges

$\Rightarrow$ Graph is bipartite

G here has an odd cycle.

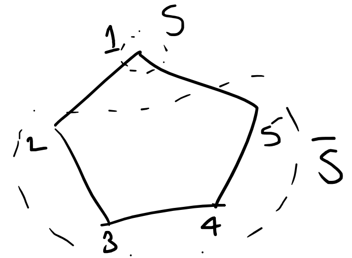
Max Cut for bipartite graphs is easy

just need to recover the partition.

Q: what about general graphs?

Idea (1): Local Search.

- Start with an arbitrary $(S, \bar{S})$ partition.
- If by moving a vertex from one side to the other, we can increase # edges cut, we'll do it.



Example.

$S_1 = \{1\}$ $\bar{S}_1 = \{2, 3, 4, 5\}$

edges cut = 2.

Lets swap 3 over.

$S_2 = \{1, 3\}$ $\bar{S_2} = \{2, 4, 5\}$

edges cut = 4.

"LOCALLY OPTIMAL, NO SINGLE SWAP CAN IMPROVE OBJECTIVE VALUE"

How good is this Algo for general graphs?

ANS: Quite good! Final solution $(S, \bar{S})$

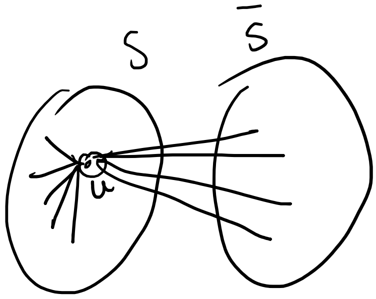
<u>ANS</u> : Quite good! Final solution $(S, \bar{S})$ cuts $\geq \frac{m}{2}$ edges of $|E| = m$.

<u>Proof</u> :

Let $(S, \bar{S})$ be the final soln.

$\forall$ u, # edges incident to u crossing the partition

$\geq$ # edges incident to u within u's side.



$\Rightarrow$ u can swap

$\Rightarrow 2$(# edges from u crossing the partition) $\geq \delta(u)$

$$\Rightarrow \boxed{\text{\# edges from } u \text{ crossing } \geq \frac{\delta(u)}{2}}$$

SUM OVER ALL u

$$2|E(S, \bar{S})| \geq \frac{\sum \delta(u)}{2}$$

$$= m.$$

$$\Rightarrow \boxed{|E(S, \bar{S})| \geq m/2 \geq \frac{OPT}{2}}.$$

{ $\frac{1}{2}$ - approximation }

... ization problems,

$\frac{1}{2}$ - approximation

For maximization problems,
C approx $\Rightarrow$
$\text{Val(Alg)} \geq C \cdot \text{Val(OPT)}$.
for $C \leq 1$ (larger C is better)

Q: Can we do a better analysis?

Ans: I think not ☹ (try showing?).

If we're trying to show that local search can do $\geq C \cdot m$ for $C > \frac{1}{2}$, that may not be possible.
(complete graph).

Ans ②: Are there graphs where OPT-Cut very close to m, but local search close to $m/2$?

Was long standing open problem to beat factor $\frac{1}{2}$ for Max Cut.

Seminal work [Goemans - Williamson] which try to characterize Max Cut using linear programming.

which ... using linear programming.

FUNNY: (LP will have infinitely many constraints) but still "solvable" in poly-time

---

$$\text{Max} \sum_{(u,v)\in E} y_{uv}$$

$$0 \le x_u \le 1 \quad \forall u$$

for Min Cut we used this: $y_{uv} \ge |x_u - x_v| \quad \forall u,v$

expressible as linear constraint

- LP can cheat in above formulation
- All y_{uv} can be set to one!

whereas we need

$$\text{Max} \sum y_{uv}$$

$$y_{uv} \le |x_u - x_v| \quad \forall u,v$$

$$0 \le x_u \le 1 \quad \forall u$$

Problem ☹ Not a linear constraint

$y_{uv} \leq \max\left(x_u - x_v \;,\; -(x_u - x_v)\right)$.

is not linearly expressible

whereas

$y_{uv} \geq \max\left(\;"\; , \;"\; \right)$

is expressible as

$y_{uv} \geq x_u - x_v$ and

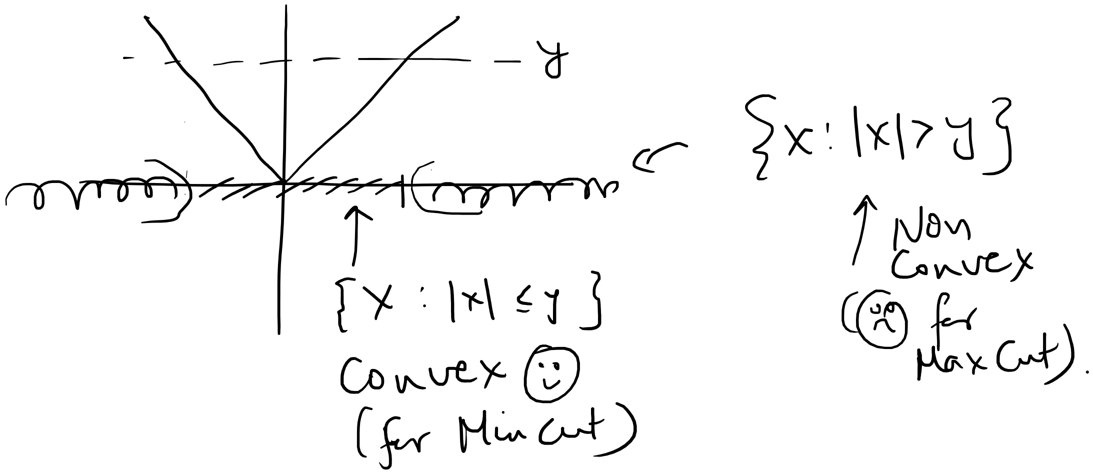
$y_{uv} \geq -(x_u - x_v)$.

Another way to see this issue :-

for any fixed y

$|x| \leq y$ is a convex set

$|x| \geq y$ is not a convex set.



Can we "express" $y_{uv} \leq |x_u - x_v|$

(or approximately)

using a linear program?

(or approx. ...)) using a linear program?

Hmmm...

Maybe not fully well, but lets also try $y_{uv} \le (x_u - x_v)^2$

This also captures $\mathbb{1}\{x_u \ne x_v\}$

Really, we're trying to capture $\mathbb{1}(x_u \ne x_v)$ using some nice constraints

To make life easier, let us use a slightly different notation.
don't think of x_u as 0/1 variables
Lets think of them as ± 1 variables.

Max Cut (exact formulation).

$$\begin{bmatrix} \text{Max} \sum_{(u,v)\in E} \frac{1}{2}(1 - y_{uv}) & \\ x_u \in \{\pm 1\} & \forall u \\ y_{uv} = x_u \cdot x_v & \forall u,v \end{bmatrix}$$

$x_u \in \{\pm 1\}$ is equivalent

$x_u \in \{\pm 1\}$ is equivalent to $x_u^2 = 1$, (ie) $Y_{uu} = 1$.

$$\Rightarrow \quad \text{Max} \sum_{(u,v)\in E} \frac{1}{2}(1 - Y_{uv})$$

$$\begin{cases} Y_{uu} = 1 & \forall u \\ Y_{uv} = x_u \cdot x_v & \forall u, v \end{cases}$$

Want to express this using linear constraints.

What sort of constraints do these Y_{uv}s satisfy?

Ex
we know $(x_u + x_v)^2 \geq 0$

$$x_u^2 + x_v^2 + 2x_u x_v \geq 0$$

$\Rightarrow$ we can enforce

$$Y_{uu} + Y_{vv} + 2Y_{uv} \geq 0$$

$$\Rightarrow \boxed{Y_{uv} \geq -1} \quad \because Y_{uu} = 1$$

Similarly

$$(x_u - x_v)^2 \geq 0$$

$$Y_{uu} + Y_{vv} - 2Y_{uv} \geq 0$$

$\boxed{Y_{uv} \leq 1}$ $\Leftarrow$ These are nice linear constraints in the Y variables in the y variables

Qn: what is a nice family of linear constraints we can impose on y_{uv}? (which the unknown sol^n satisfies)

SEMIDEFINITE PROGRAMMING

Consider <u>any</u> real vector $(c_1, c_2, \ldots, c_n)$
$c_i \in \mathbb{R}$

let's look at

$$\left(\sum c_i x_i\right)^2 \geq 0 \quad \leftarrow \text{true always}$$

$$\Rightarrow \sum_{i,j} c_i c_j x_i x_j \geq 0$$

ordered pairs (including $i=j$)

$\Rightarrow$ we can enforce

$$\boxed{\sum_{i,j} c_i c_j y_{ij} \geq 0}$$

Linear constraint on the y variables

Here is a relaxation for Max Cut

Vertices are labeled $1, 2, \ldots, n$.

$$\left\{ \begin{array}{l} \text{Max} \sum_{(i,j) \in E} \frac{1}{2}(1 - y_{ij}) \\ \forall i: \quad y_{ii} = 1 ; \\ \forall c = (c_1, \ldots c_n) ; \sum c_i^2 y_{ii} + \sum_i \sum_{j \neq i} c_i c_j y_{ij} \geq 0 \\ \forall (i,j) : y_{ij} = y_{ji} \text{ (ordered)} \end{array} \right\}$$

Linear Program with ∞ many constraints !!!

<u>Lemma ①</u>

If $(S, \bar{S})$ is any cut with cut value V then there is a feasible sol^n to above LP with obj value V.

$$\boxed{\text{This LP is actually called a semi-definite program}}$$

How to solve the LP if it has ① Infinitely many constraints?

② [Given a "claimed" $\{y\}$ solution, how do we even check if it is a feasible solution to LP?]

//

Idea for (2)

express y_{ij} values as a matrix

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & & & \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix}$$

(1) Then we can easily check

$y_{ii} = 1$ and

$y_{ij} = y_{ji}$ in n^2 time.

(2) How to check $\left(\sum c_i^2 y_{ii} + \sum_i \sum_{j \neq i} c_i c_j y_{ij}\right) \geq 0$

let's imagine c is a column vector $\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$.

(2) is equivalent to

$$c^T Y c \geq 0 \quad \forall c.$$

A real-symmetric matrix Y is called a "positive-semidefinite matrix" if

$$c^T Y c \geq 0 \quad \forall c \in \mathbb{R}^n$$

Checking ② $\iff$ Checking if Y is PSD or not.

THEOREM

Real symmetric Matrix Y is PSD:
[3 equivalent forms]

① $c^T Y c \geq 0 \quad \forall c \in \mathbb{R}^n$

② All eigenvalues of Y are ≥ 0

③ There exist vectors $w_1, w_2, \ldots w_n$ Such that $Y_{ij} = \langle w_i, w_j \rangle$.

Given $A \in \mathbb{R}^{n \times n}$, (λ, v) is eigenpair if $\boxed{Av = \lambda v}$

Now back to checking if a given Y satisfies LP or not?
(simply check if all eigen values ≥ 0)

Checking feasibility is OK, but how to solve this LP [called a semidefinite program]?

$$\text{Max} \sum \tfrac{1}{2}(1 - y_{ij})$$

$(i,j) \in E$

can be avoided because PSD will imply

$$y_{ii} = 1 \quad \forall i$$
$$y_{ij} = y_{ji} \quad \forall (i,j)$$

$$Y \succcurlyeq 0 \quad [Y \text{ is a PSD matrix}]$$

- equivalent to $\forall c \; c^T Y c \geq 0$
- equivalent to all eigs of $Y \geq 0$
- equivalent to $\exists$ w_i vectors st $Y_{ij} = \langle w_i, w_j \rangle$.

Beautiful Technique for solving Very Large LPs

Ellipsoid Method / Separation Oracle

Given an LP, and a "claimed" solution for the LP,

If there is an algorithm which can efficiently tell if the claimed sol^n is feasible or not

then we can actually solve the LP efficiently!!

and "almost optimally" (Negligible error)

... (#variables)

(Negligible error)

Efficient = poly (# variables) · time of checker

~~OK~~ We have solved the LP.
Gotten ourselves a Y matrix and
w_i vectors st $Y_{ij} = \langle w_i, w_j \rangle$.

$\boxed{\text{How to get a cut from this?}}$

Next class !

ROUNDING ALGO :-

Let Y^* be the optimal SDP soln.

$$\text{SDP Obj} = \sum \tfrac{1}{2}(1 - Y^*_{ij})$$

From Y^*, (because $Y^* \succeq 0$ is PSD)
we can recover vectors
$w_1^*, w_2^*, \ldots, w_n^* \in \mathbb{R}^n$
such that $Y^*_{ij} = \langle w_i^*, w_j^* \rangle$.

Moreover these are unit vectors

b/c $1 = Y^*_{ii} = \langle w_i^*, w_i^* \rangle = \|w_i^*\|^2$

NOTE: If w_i^* vectors were in 1 dimensional space, they would be ± 1 scalars and $Y^* = x_i^* \cdot x_j^*$ would hold

$$\text{SDP objective} = \frac{1}{2} \sum_{(i,j) \in E} (1 - \langle w_i^*, w_j^* \rangle).$$

SDP objective $= \frac{1}{2} \sum_{(i,j) \in E} (1 - \langle w_i^*, w_j^* \rangle)$.

all w_i^* are unit vectors.

Real goal in ROUNDING

How to convert high-dimensional vectors onto 1 dimension while roughly preserving inner products / distances?

Idea Random projection!

2 view points of this algo (equivalent)

VIEW ①

Pick $\tilde{g} \in \mathbb{R}^n$: $(g_1, g_2, \ldots, g_n)$ random gaussian vector

Let $x_i = 1$ if $\langle w_i^*, g \rangle \geq 0$

$= -1$ otherwise.

Geometric View

b/c $\tilde{g}$ is spherically symmetric, it is same as choosing random hyperplane

choosing random hyperplane and partitioning

Normal hyperplane $\langle x, \tilde{g} \rangle = 0$

Overall Algo

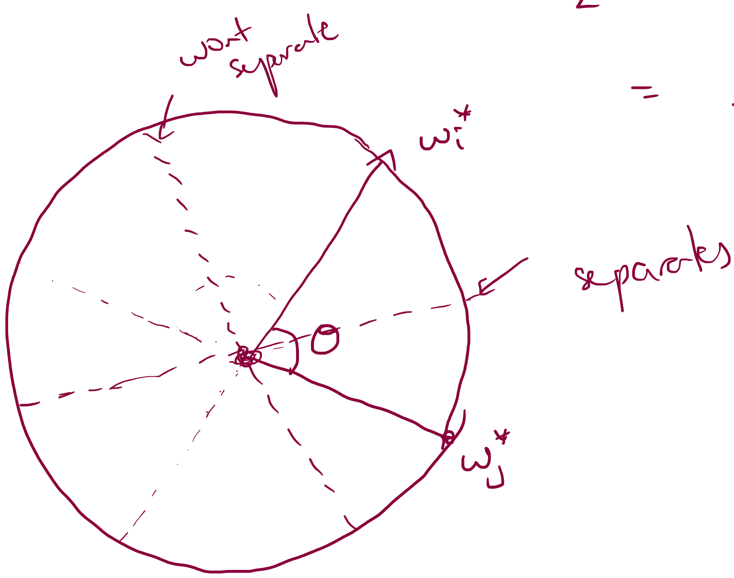
- Solve SDP optimally
- From Y^*, recover $w_1^*, w_2^*, \ldots w_n^*$
- Sample random gaussian vector $\tilde{g} = (g_1, g_2, \ldots, g_n)$
- Set $X_i = 1$ if $\langle w_i^*, \tilde{g} \rangle \geq 0$, -1 else
- Output $S = \{ i : X_i = 1 \}$ as cut.

Analysis:

Fix (i,j): SDP is cutting this edge to extent of

$$\frac{1}{2}\left(1 - \langle w_i^*, w_j^* \rangle\right).$$

$$= \frac{1}{2}\left(1 - \cos\theta\right).$$



Our algo cuts (i,j) whenever the random hyperplane separates w_i^* &

Our algo cuts (i,j) [if] random hyperplane separates w_i^* & w_j^*

QN: If we choose a random hyperplane, what's the probability that it separates w_i^* and w_j^*

$$\Pr[(i,j) \text{ is cut}] = \frac{\theta}{\pi}.$$

① Our prob. of cutting edge $= \frac{\theta}{\pi}$

② SDP obj. $= \frac{1}{2}(1-\cos\theta)$

ANALYSIS QN:

How small can ① be compared to ② ?

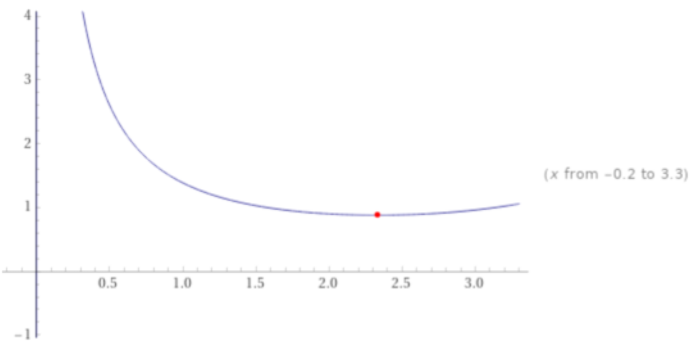
minimize	function	$\dfrac{2 \times \frac{x}{\pi}}{1-\cos(x)}$
	domain	$0 \le x \le \pi$

Global minimum:

More digits | Exact form | Step-by-step solution

$$\min\left\{\frac{2x}{\pi(1-\cos(x))} \,\middle|\, 0 \le x \le \pi\right\} \approx 0.878567 \text{ at } x \approx 2.33112$$

Plot:



$\Rightarrow$ ① ≥ 0.878 ② $\forall\, \theta$.

$\Rightarrow$ (1) $\geq$ 0.878 (2) $\forall\ \theta$.

$$\Rightarrow E[\#\text{ cut edges}] = \sum_{(i,j)\in E} \Pr[(i,j) \text{ is cut}]$$

$$\geq \sum_{(i,j)\in E} 0.878 * \frac{1}{2}(1-\cos\theta_{ij})$$

$$= 0.878 \sum_{(i,j)\in E} \frac{1}{2}(1-\langle w_i^*, w_j^*\rangle)$$

$$= 0.878 \sum_{(i,j)\in E} \frac{1}{2}(1-y_{ij}^*).$$

$$= 0.878\ \underline{\text{SDP OPT}}.$$

---

(4) Back to how to solve the SDP?

Ellipsoid Algorithm :-

Black box way to convert feasibility to optimization

SEPARATION ORACLE

Given a very large LP/SDP and a $\tilde{y}$

A is a sep. oracle if it can efficiently tell if $\tilde{y}$ is feasible for LP/SDP OR prove why $\tilde{y}$ is not feasible, (ie) output some constraint of LP/SDP which $\tilde{y}$ does not satisfy.

$\tilde{y}$ does not satisfy.

Ellipsoid Algo

If $\exists$ a separation oracle, then the said LP/SDP can be optimally* solved (up to any desired accuracy)

Q: What's the separation oracle for Max cut?

Ans: Just the eigen solver! :)

Given the $\tilde{Y}$ matrix, compute all eigen values and eigenvectors

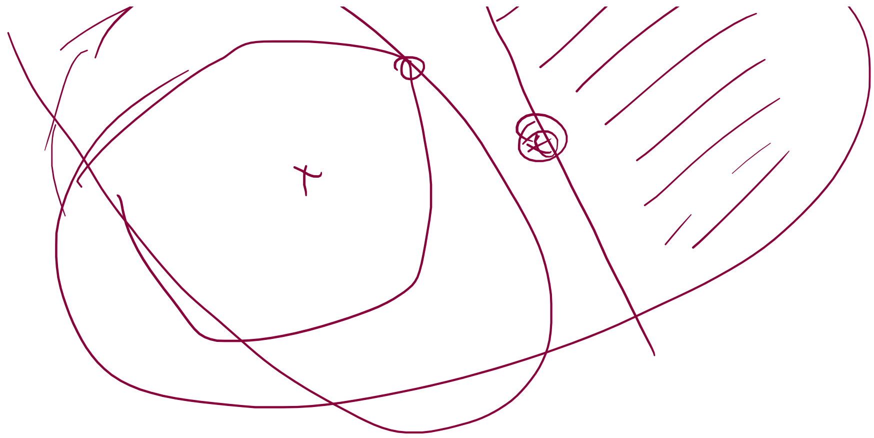
if all λ_i are ≥ 0, say feasible (also if all $\tilde{Y}_{ii} = 1$),

if $\exists$ $\lambda_i < 0$, with eigenvector v_i,

Output violated constraint $\boxed{v_i^T Y v_i \geq 0}$

This is enforced in the LP, but the given $\tilde{Y}$ matrix won't satisfy

$$v_i^T \tilde{Y} v_i = v_i^T \lambda_i v_i = \lambda_i \|v_i\|^2 < 0 \quad \blacksquare$$



Lots of data, lots of "analysis" one can do.

Eg:
{ Understand causality b/w smoking and lung disease. }

↓
Population wide studies.

Q:
{ How to conduct useful population-wide studies / "data analysis" without compromising individual privacy }

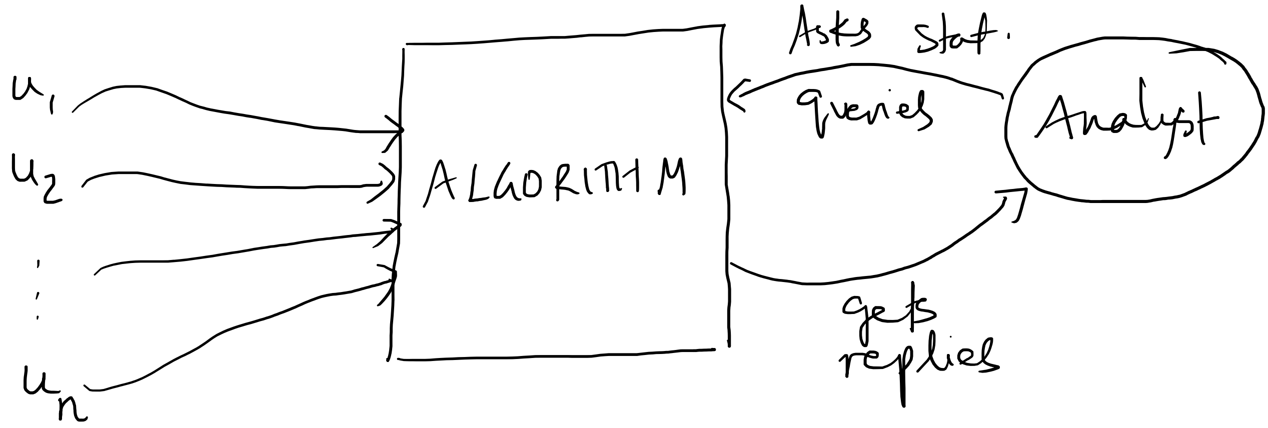
↓
Means users are incentivized to join the study
[Collectively we can learn something, but individually don't lose anything]

{ different from "CRYPTOGRAPHY" }
which is like a vault + key.

{ Here we want to contribute data but the analyst learns
... " , " }

but the analyst learns
nothing "private"

INITIAL ATTEMPT AT MODELING PRIVACY



Users provide data

Attempt ①

Want the analyst to not learn anything new about any individual.

Natural Issue

You will learn something new from the answer of the query.

Example

Example

Imagine each u_i = (# left feet, # right feet)

- Analyst Asks Avg(# left ft − # right ft)
- Reply (likely) = 0

⇒ Analyst "learns" that u_i has equal # of Left & right ft.

(Maybe need few more queries to get Min(# left ft)
Min(# right ft), etc.

- Why is this definition vacuous?

A: what we learnt is something "global".
Nothing specific about the particular individual

Q: How do we capture "Individual Privacy"?

[The earlier definition of privacy is broken]

DMNS − Dwork, McSherry, Nissim, Smith

[Differential Privacy]

[What if the Algorithm guarantees that the answer to the query is "almost" the same regardless of whether u_1 (or any fixed individual) is present in the input or not?]

The analyst can't even distinguish if u_1 was part of the study or not, so how can he/she learn anything about u_1's data?

The Differential Privacy Model

Input: Database X of 'n' rows, each corr to a user, fn f.

Algo takes X and outputs $\tilde{f}(X)$

[it can be scalar output, vector output, etc]

f is the statistical query

$\tilde{f}$ is the output.

$\tilde{f}$ is the output.

$\tilde{f}(x)$ can be some "noisy / approximate" response to $f(x)$.

If x and x' are two databases which differ in a single row, then

want $\boxed{\tilde{f}(x) \text{ "}\approx\text{" } \tilde{f}(x')}$ ①

guarantees privacy.

<u>and</u>

② $|f(x) - \tilde{f}(x)|$ is "small" for all x

guarantees utility of study.

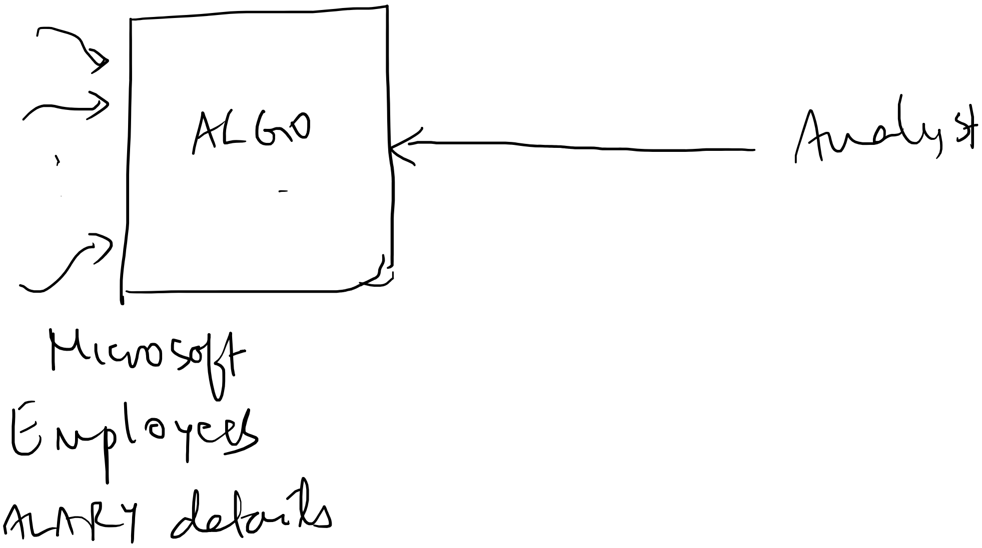
Need to formalize "$\approx$" meaning and "small"

{ Just ① is easy to satisfy:
Output $\tilde{f}(x) = 0$ always.
Full privacy, No utility }

{ How to get both together? }

There are many simple queries for which we need to introduce <u>noise</u>, else we break privacy.

<u>Ex ①</u>



Analyst asks " count # people with Salary ≥ 2M $ "

Sps Algo replies ①.

{Then analyst can learn that CEO's salary ≥ 2M $.}

Output will be 0 if CEO doesn't take part in survey.

take part in survey.

$\Rightarrow$ CEO's privacy is compromised acc. our definition

---

Is this just a 1 vs 0 issue?

Perhaps not.

Sps Answer = 1000.

and tomorrow, the answer is 1001.

Maybe Microsoft hired a new employee,

$\Rightarrow$ {likely that this person has salary $\geq 2M\$$.}

we may learn something about the new individual.

$\boxed{\text{So our Model / definition captures those issues well}}$

$\Rightarrow$ we must add some noise to $f(x)$ so that

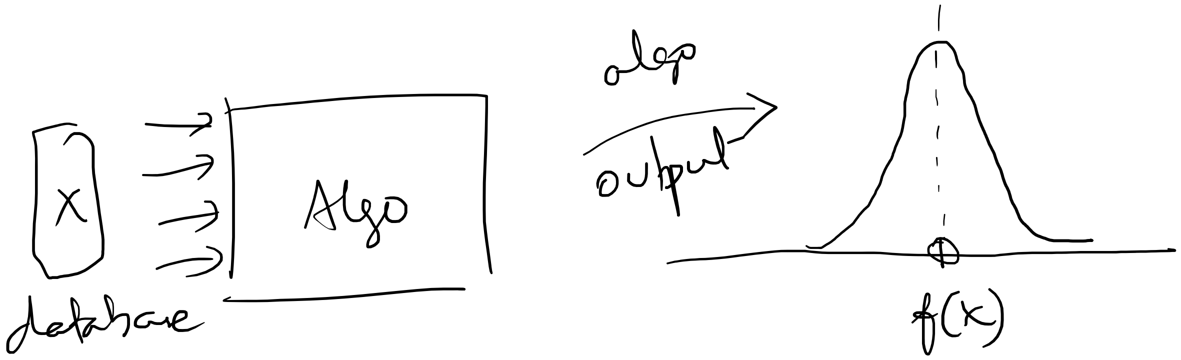
$\tilde{f}(x) \approx \tilde{f}(x')$ for all

d. having .

$\tilde{f}(x) \approx \tilde{f}(x')$ for all neighboring datasets.

Model Formally

$\boxed{\text{ALGO}}$ is randomized, adds noise acc. distribution



Distribution of $\tilde{f}(x)$.

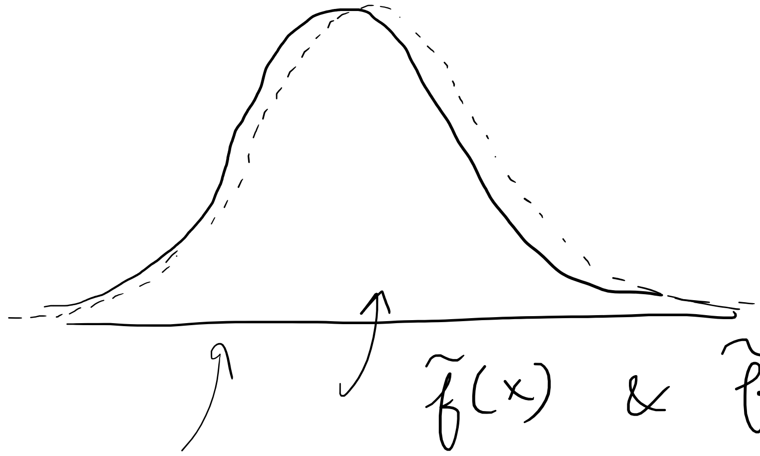
Error of Algo

$$\underset{\text{random choices}}{E}\left[\left(\tilde{f}(x) - f(x)\right)^2\right]$$

Privacy Requirement :-

$\forall$ x, x' differing in one row, want distributions to be nearly identical.

nearly identical.



$\tilde{f}(x)$ & $\tilde{f}(x')$ distribution.

$\forall\, x, x'$, and for all subsets R of possible outputs of $\tilde{f}$,

$$\boxed{\Pr\left[\tilde{f}(x) \in R\right] \le \underbrace{e^{\varepsilon}}_{\text{like } (1+\varepsilon)} \Pr\left[\tilde{f}(x') \in R\right]}$$

ϵ - Differential Privacy.

<u>Goal</u>:

[How to answer queries with ϵ - DP, with MIN. ERROR?]

↓

<u>TOMORROW</u>

Such a scheme for "counting" queries

Counting Queries

30 April 2021 11:59

Database X

Users	Salary
u_1	s_1
u_2	s_2
$\vdots$	
u_n	s_n

Query = "Count # users with salary $\geq v$"

$f(X)$

What should a good $\tilde{f}(X)$ be ?

Want ① $\tilde{f}(X)$ close to $f(X)$

② $\tilde{f}(X)$ close to $\tilde{f}(X')$ for all neighboring X'.

<u>Idea</u>

- Add noise to real answer.

Compute $f(X)$, output $f(X) + R$

R is some suitable

where R is some suitable noise.

eg $\left\{ \begin{array}{l} R \sim N(0, \sigma^2) \text{ for suitable } \sigma \\ \text{will give privacy with} \\ \text{low error for suitable parameters} \end{array} \right\}$

<u>Our Algo</u> [Technical Difference]

Add noise z w.p $\propto e^{-\frac{|z|}{\sigma}}$

(in contrast gaussian noise has prob $\propto e^{-z^2/\sigma^2}$)

"LAPLACIAN DISTRIBUTION"

$$f_R(z) = \frac{1}{2\sigma} e^{-\frac{|z|}{\sigma}}$$

<u>Check</u>

① $\displaystyle\int_{-\infty}^{\infty} f_R(z)\, dz = 1$

② $\displaystyle\int_{-\infty}^{\infty} z\, f_R(z)\, dz = E[R] = 0$

③ $\displaystyle\int_{-\infty}^{\infty} z^2 f_R(z)\, dz = E[R^2] = 2\sigma^2$

$$\int_{-\infty} z^2 f_R(z)\,dz = E[R^2]$$

eg, Integration by parts

If we add noise acc. Laplacian(σ), What is the squared error like?

$$\tilde{f}(x) = f(x) + R$$

$$\Rightarrow \text{error} = E\left[\left(\tilde{f}(x) - f(x)\right)^2\right]$$

$$= E\left[R^2\right] \qquad \text{where } R \sim Lap(\sigma).$$

$$= 2\sigma^2$$

Want to set σ sufficiently large to get desired privacy.

$\forall$ x, x' differing in a row, and any subset S of output values

Want

$$\Pr\left[\tilde{f}(x) \in S\right] \leq e^{\varepsilon} \Pr\left[\tilde{f}(x') \in S\right]$$

Privacy Requirement.

We'll instead set noise such that

We'll infact set noise such that the pdf of Algo output for X and X' are <u>very similar</u>.

Fix an output value 't'.

Let $f_{Alg}(X, t)$ = PDF of Alg outputting t on input X

$$= \frac{1}{2\sigma} \exp\left(- \frac{|f(X) - t|}{\sigma} \right)$$

Similarly,

$$f_{Alg}(X', t) = \frac{1}{2\sigma} \exp\left(- \frac{|f(X') - t|}{\sigma} \right)$$

$$\Rightarrow \quad \frac{f(X,t)}{f(X',t)} \leq e^{-\frac{|f(X) - f(X')|}{\sigma}}$$

$$\leq e^{-\frac{1}{\sigma}}$$

So we can set $\sigma = \frac{1}{\varepsilon}$.

$\forall x, x', t$ neighboring databases

$$\boxed{\frac{f(x,t)}{f(x',t)} \leq e^{\varepsilon}} \leftarrow \text{Privacy}$$

And $E\left[\left(\tilde{f}(x) - f(x)\right)^2\right] \leq \frac{2}{\varepsilon^2}$

Utility

Only thing we used in proof is how much f can change from $x \rightsquigarrow x'$.

SENSITIVITY of fn.

$$\Delta_f = \max_{\substack{x,\, x' \\ \text{differing} \\ \text{in one} \\ \text{row}}} |f(x) - f(x')|$$

Noise will simply depend on Δ_f by setting σ appropriately to ensure ϵ-Privacy.

SUMMARY

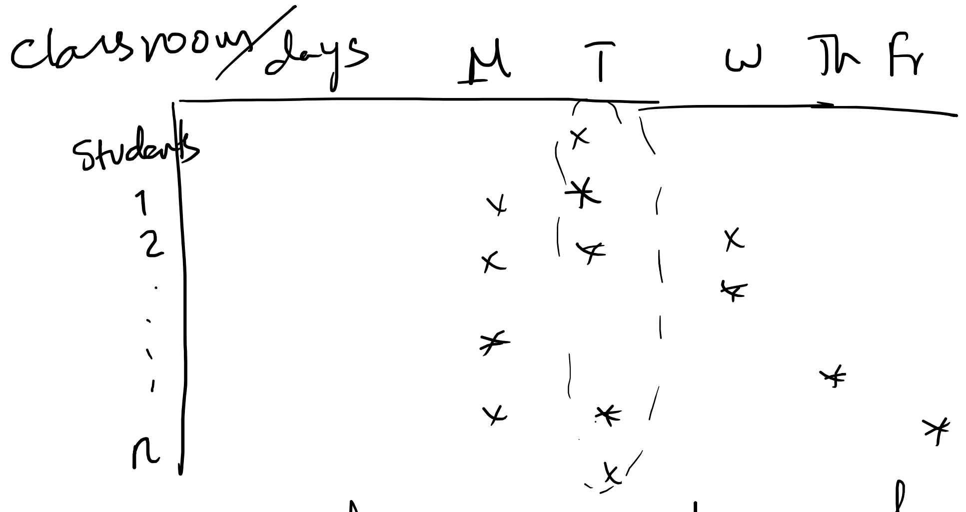
Simple scheme which works not just for counting queries, but for any low-sensitivity function any low-sensitivity function

Algo is called "Laplace Mechanism"

Similarly, Gaussian Mechanism also exists (w/ gaussian noise).

In above example, query answer was a numerical value. what if its not?

Example



Each student has a preference for when to conduct exam.

We want to select "Most popular day" in a differentially private manner.

Q1: - Cant simply add noise, (Meaningless)

Q1: - Cant simply add noise, (Meaningless)
Q2: - How to measure utility of a scheme.

Privacy is easy to ~~extend~~

$$\frac{\Pr[\text{Alg selects Monday for } X]}{\Pr[\text{" " " " } X']} \leq e^{\varepsilon}$$

Similarly for each other day.

$\boxed{\text{How to get utility ?}}$

Want output to be a popular day if not the MOST popular day.

Idea

For each possible output (days in our example)

	Day 1	2	3	4	5
compute # people who prefer	n_1	n_2	n_3	n_4	n_5

Output day 'i' as answer with prob $= e^{\varepsilon \cdot n_i}$

$$\overline{\sum e^{\varepsilon n_i}}$$

$\Rightarrow$ Intuitively popular days are more likely to be output.

Privacy + Error Analysis

For any day i and inputs X and X'

$$\frac{\Pr[\text{Alg selects } i \text{ for } X]}{\Pr[\text{Alg selects } i \text{ for } X']}$$

$$= \frac{e^{\varepsilon n_i(x)}}{\sum_j e^{\varepsilon n_j(x)}} \cdot \frac{\sum_j e^{\varepsilon n_j(x')}}{e^{\varepsilon n_i(x')}}$$

$$= \underbrace{e^{\varepsilon(n_i(x) - n_i(x'))}} \cdot \frac{\sum e^{\varepsilon n_j(x')}}{\sum e^{\varepsilon n_j(x)}}$$

Since X and X' differ in a single row,

each $-1 \leq n_i(x) - n_i(x') \leq 1$

for all i.

Overall, $\dfrac{\Pr[\text{Alg output } i \text{ on } X]}{\Pr[\text{Alg output } i \text{ on } X']} \le e^{\varepsilon} \cdot e^{\varepsilon} = e^{2\varepsilon}$.

Satisfies 2ε- Differential Privacy.

What about error?

Let database has n people

and suppose $n_1 = OPT$ is the day with largest count.

Ideally: Want Alg to output a day with count close to n_1

Let's calculate

$\Pr[\text{Alg outputs a day with count} \le n_1 - t]$

Let's fix a day i with count $\le n_1 - t$

$$\Pr[\text{Alg outputs this day}] = \frac{e^{\varepsilon n_i}}{\sum_j e^{\varepsilon n_j}}$$

to [Alg outputs ... $\frac{\ldots}{\sum e^{\varepsilon n_j}}$

$$\leq \frac{e^{\varepsilon(n_1 - t)}}{e^{\varepsilon n_1}} \leq e^{-\varepsilon t}.$$

So for $t = \frac{\log(n/\delta)}{\varepsilon}$

Thus probability is $\leq \delta/n$

$\Rightarrow$ Union bound over all bad days gives

Pr[outputting a bad day] $\leq \delta$

$\Rightarrow$

① with good probability, Algo chooses a day with score $\geq$ OPT $- \frac{\log(n/\delta)}{\varepsilon}$

[Very useful if OPT $\gg \frac{\log n}{\varepsilon}$]

AND

② Preserves 2ε- Privacy of users.

EXPONENTIAL MECHANISM.

Brief Intro to Hardness of Approximation

03 May 2021 10:01

Q: What is the class NP?

Defn: defined for <u>decision problems</u>

e.x is given a graph, is it 3-COLOURABLE?

Here $\mathcal{L} = \{ G : G \text{ is 3-colourable} \}$

$\left[\text{NP is a class of languages which admits a poly-time non-deterministic turing machine.} \right]$

In contrast P is the class of languages identifiable with deterministic poly-time turing machine

Equivalent viewpoint of the class NP in a "prover - verifier language"

NP is the class of languages L st

$\exists$ deterministic polynomial time Algo (VERIFIER)

st

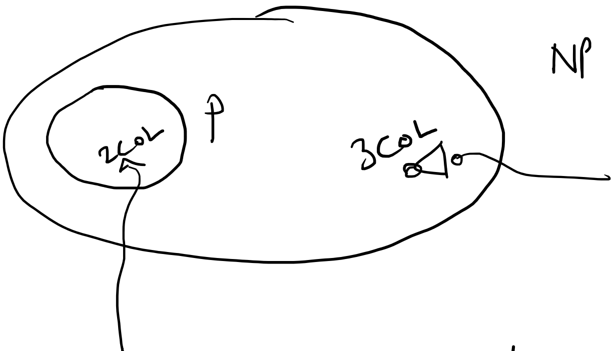
$\forall x \in L$, $\exists$ proof $\pi(x)$ of poly($|x|$) bits

such that

$$V(x, \pi(x)) = 1$$

$\forall x \notin L$, no proof should make V accept

(i.e. $\forall \pi(x)$, $V(x, \pi(x)) = 0$)



Given graph, is it 3-colourable?

2COL: Given a graph, is it 2-colourable?

$L = \{$ 2-colourable graphs $\}$

$\boxed{\text{Is } P = NP}$ Most longstanding problems in complexity theory.

{Widely believed that $P \neq NP$.}

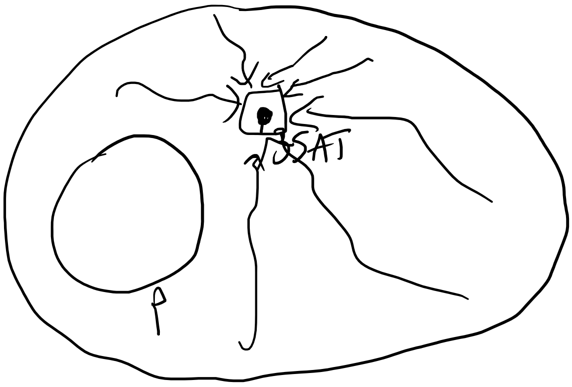
Cook-Levin Theorem [1971]

3SAT is NP-Complete.

Given a CNF where each clause is an OR of 3 literals (x_i or $\bar{x}_i$)

check if it is satisfiable or not.

When is a problem NP-complete?



A problem Q is NP Complete if

a) it is in NP

b) every other problem in NP can be reduced to Q in poly-time.

Meaning,
if we discover a poly-time algo for 3SAT, then we can obtain poly-time algos for any problem in NP by using Cook-Levin Thm.

This notion of NP completeness gave a way of showing how "hard" some decision problems are.

3COL is NP-complete
How would we show something like this?

a) 3COL $\in$ NP [Proof is the coloring]
b) Every other problem in NP can reduce to 3COL in poly-time?

(b) seems tricky, but its not due to Cook-Levin simply reduce 3SAT ~~to~~ 3COL in polytime

(ies)
given I of 3SAT, output a graph $G(I)$ in poly-time such that

I is satisfiable $\rightleftharpoons$ G is 3-colorable.

Great! We have a way of characterizing "hardness" of decision problems by showing NP-completeness.

For longest time, it was not clear how to extend these ideas / notions to optimization / approximation-type problems. ☹

Ex:
We know 3-SAT is NP-complete.
Let's look at Max-3SAT:
given an instance, find the

assignment which satisfies
max #clauses ?

- Can't admit poly-time algos (if $P \neq NP$)

Can it admit a PTAS ?

(ie) $\forall$ constant $\varepsilon > 0$, can we get a $(1-\varepsilon)$-
approximation algorithm which
runs in poly-time ?

Similarly can ask for
Graph colouring.

Given graph G, colour it with
fewest colours.

What's the best approximation algo for
this ? Is there a
$(1+\varepsilon)$ - approximation ? Is there
a 2-approximation ?

Another side of the Story.

How to analyze the class NP from
the perspective of the verifier ?

How all can we restrict the verifier and
still admit the NP-class ?

(long story for the motivation)

One attempt [ALMSS '98]

Probabilistically Checkable Proof $\boxed{PCP}$

PCP[r(n), s(n)] is the class of all languages L s.t. $\exists$ poly-time verifier V

$x \in L \Rightarrow \exists$ proof $\pi(x)$ of size poly(|x|) such that

V probes only s(|x|) bits of the proof and accepts with prob 1.

$x \notin L \Rightarrow \forall$ proofs, , V rejects after checking s(|x|) bits of the proof with prob $\geq \frac{1}{2}$.

The verifier V is restricted to probe only s(n) bits of the proof.

Hence we give it some power to toss r(n) random coins.

So V can look at x and the $r(n)$ coins to determine which $s(n)$ bits to probe.

$PCP[r(n), s(n)]$

poly time, randomized verifier which tosses $r(n)$ coins

$x \in L \Rightarrow$ $\exists$ proof $\pi(x)$, V probes $s(n)$ bits of proof π and accepts w.p 1

$x \notin L \Rightarrow \forall$ proof π, V rejects wp $\geq 1/2$.

REMARKABLE THM [ALMSS '98]

$$NP = PCP[O(\log n), O(1)] \quad !!!$$

$\Rightarrow$ Can check if $x \in L$ by looking at only constant bits of the proof!

Has beautiful connections to hardness of approximations

Opened the door to studying if 3SAT, COL have PTAS, etc.

PCPs and Hardness of Approximations

04 May 2021 09:20

Goal: Hardness of Approximation of Max3SAT

Prior to PCPs, was not known if
Max 3SAT admits a $(1-\varepsilon)$-approximation
for any constant $\varepsilon > 0$.
(ie) PTAS ?

Emergence of PCPs $\Rightarrow$ PTAS NOT POSSIBLE

There is some fixed $\varepsilon > 0$ for which it
is NP-hard to approximate Max3SAT
to a factor better than $(1-\varepsilon)$.

Subsequent improvements to PCP Machinery
$\forall\ \varepsilon > 0$, it is NP-hard to get a
$\left(\frac{7}{8}+\varepsilon\right)$-approx to Max 3SAT.

And we know trivial $\frac{7}{8}$-approximation.

Q: What does it mean?

A: If you design a poly-time $\left(\frac{7}{8}+\varepsilon\right)$-approx for $\varepsilon > 0$
then we can design a poly-time
algo for 3SAT.

PCP view of the NP class allows us to go from such approx-hardness to classical NP-completeness.

Q: How are we going to show APX-hardness of Max 3SAT?

Idea: create an intermediate decision problem and show it is NP-complete.

Problem

Gap $3SAT_{c,s}$:

Given an instance $\mathcal{I}$ of 3SAT, output

1. Yes if there is an assignment satisfying $\geq c$ fraction of clauses
2. NO if NO assignment can satisfy $\geq s$ fraction of clauses
3. YES/NO if intermediate values.

Ex: for $c = 1$, $s = 0.9$

Gap $3SAT_{1, 0.9}$:

Intuitively easier than regular 3SAT, where we need to give, ...

correct answer in all regimes of satisfiability.

Using PCP Thm, we can show that (★) $Gap\,3SAT_{1,1-\varepsilon}$ is NP-complete for some constant $\varepsilon > 0$.

Using (★), easy to show $(1-\varepsilon)$-hardness of Max 3SAT.

Sps $\exists$ A which is a $(1-\varepsilon)$-approx for Max 3SAT

Given I of $Gap\,3SAT_{1,1-\varepsilon}$, run A on I.
if it finds an assignment satisfying $\geq (1-\varepsilon)$ fraction of clauses,
Output YES
- Else output NO

<u>Todo</u>
Show that $Gap\,3SAT_{1,1-\varepsilon}$ is NP-complete

<u>Proof</u>
From PCP Theorem + reduction
We know that 3COL is NP-complete.

We'll reduce it to Gap $3SAT_{1,1-\varepsilon}$.

Traditional reductions don't give such "gaps"

PCP view comes to our rescue.

$$\boxed{3COL} \xrightarrow[\text{reduction}]{\text{poly-time}} \boxed{\text{Gap } 3SAT_{1,1-\varepsilon}}$$

We'll use the PCP view of 3-coloring.

$\exists$ verifier V which tosses $O(\log n)$-random coins, and looks at $O(1)$ bits of the proof to decide if $G \in 3COL$.

- if $G \in 3COL$, $\exists$ proof st $\Pr_{coins}[V \text{ accepts}] = 1$
- if $G \notin 3COL$, $\forall$ proofs $\Pr_{coins}[V \text{ accepts}] \leq 1/2$.

Our reduction will create an instance of Gap $3SAT_{1,1-\varepsilon}$ as follows :-

Imagine that proof provided is of the form

$x_1, x_2, x_3, \ldots \ldots x_N$

where $N = \underline{\text{poly}(n) \cdot \text{proof size}}$

Verifier tosses $O(\log n)$ - coins

C_1	C_2	$\cdots$	C_r		probe		truth-table
$0,$	$0,$	$\cdots,$	0	$\to$	$\underbrace{x^1_1, x^1_2, \ldots, x^1_5}$	$\to$	0 / 1
$\vdots$					$\vdots$		
$0,$	$1,$	$1, 1,$	1	$\to$	$x^k_1, x^k_2, \ldots x^k_5$	$\to$	0 / 1

We'll form a 3CNF from these truth tables.

We'll convert each truth table to a collection of clauses. (small #)

For example

x_1	x_{10}	x_{15}	x_{20}	output of V
0	1	1	1	0
1	0	1	0	0
1	1	1	1	0

otherwise accept.

Verifier is checking

$$(\bar{x}_1 \wedge x_{10} \wedge x_{15} \wedge x_{20}) \vee$$

$$\text{reject} \Leftarrow (x_1 \wedge \bar{x}_{10} \wedge x_{15} \wedge \bar{x}_{20}) \vee (x_1 \wedge x_{10} \wedge x_{15} \wedge x_{20})$$

$$\text{accept} \Leftarrow (x_1 \vee \bar{x}_{10} \vee \bar{x}_{15} \vee \bar{x}_{20}) \wedge (\bar{x}_1 \vee x_{10} \vee \bar{x}_{15} \vee x_{20}) \wedge (\bar{x}_1 \vee \bar{x}_{10} \vee \bar{x}_{15} \vee \bar{x}_{20})$$

[Can use some small # auxillary variables to make it a 3CNF.]

In total, we'll generate poly(n) poly(s) clauses but s is constant.

This is the desired Gap 3SAT instance I

Final part of proof:

$G \in$ 3COL, then all clauses of I are satisfied

$G \notin$ 3COL, then $\leq (1-\varepsilon)$-fraction of clauses of I are satisfied

---

PCPs are intimately tied to hardness of Approx

How to show hardness of Apx?
For optimization problem P (let's say Maximization)
Create a Gap $P_{c,s}$ problem (decision problem)
& show it is NP-complete
$\Rightarrow$ NP-Hardness of factor $\left(\frac{s}{c}\right)$ for problem P.

Try to find c, s such that $\frac{s}{c}$ is as small as possible

This gap is related to accept prob. of verifier in PCP theorem.

"Parallel Repetition theorem"

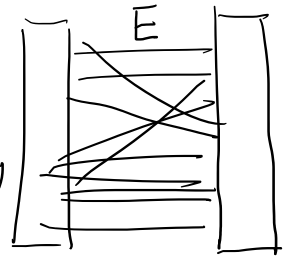
lots of research to trade-off
r(n), s(n), accept probability

Culminated in a very neat abstraction called label cover problems.

equivalent form of PCP Thm + Parallel Repetition
where verifier probes only 2 locations of the proof.
[Catch: proof is no longer bit string, its over a larger alphabet]

Max LabelCover Problems:

Given a bipartite graph $G = (U, V, E)$ and a "projection function" $f_{u \to v} : [L] \to [K]$ on each edge $(u,v) \in E$,



goal

Pick labels for each vertex (ie)

$\ell(u) \in [L]$ for all $u \in U$

$\ell(v) \in [K]$ for all $v \in V$

U ↑ label set $= [L]$ V label set $= [K]$

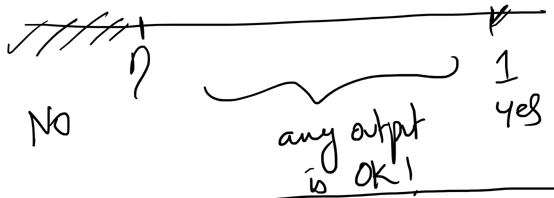
Max # edges for which labels are aligned wrt f.

(ie) $\boxed{f_{u \to v}(\ell(u)) = \ell(v)}$

Think of K and L as being constants

Gap Label Cover$_{1, \eta}$

Given an instance of label cover, can we decide if ∃ labelling which can satisfy <u>all</u> <u>edges</u> vs if all labellings can satisfy $\leq \eta$ fraction of <u>edges</u>?



THEOREM:

For <u>any</u> constant $\eta > 0$, ∃ constant K, L bounded by poly$(1/\eta)$

Gap Label Cover ... is ... Moreover

K, L bounded by poly$(1/\eta)$
Such that the GapLabelCover$_{1,\eta}$ is NP-complete.

Moreover, this holds even when $|U| = |V|$ and graph is regular

Many, if not most, hardness of approx. results come with the GapLabelCover as starting point.

$$\text{GapLabelCover}_{1,\eta} \xrightarrow{\text{reduction}} \text{GapP}_{c,s}$$

will get us some hardness for problem P.

Example Application (Hardness of Max-Coverage)

<u>Problem</u>:
Given a set system $(U, \mathcal{S})$ where
U is universe of elements and
$\mathcal{S}$ is collection of subsets of U,
and given parameter k, <u>goal</u>:
choose k sets from $\mathcal{S}$, say
$S_1, S_2, \dots, S_k \in \mathcal{S}$ to maximize
$\left|\bigcup_{i=1}^{k} S_i\right|$ (ie the # elts covered by them).

<u>Algorithmic Ideas</u>:-
① greedy algo: for $t = 1, 2, \dots, k$.
choose set which covers max # uncovered elts

② LP - rounding?

③ local search: Start with k random sets, and swap in a new set for an existing set if coverage improves

$\boxed{\text{THM: Greedy Algo is a } \left(1-\frac{1}{e}\right)\text{- approximation}}$

THM
For any fixed $\varepsilon > 0$, it is NP-complete to design a poly-time $\left(1-\frac{1}{e}+\varepsilon\right)$-approximation

PCP + Label Cover viewpoint

THM 2
For any fixed $\varepsilon > 0$, it is NP-complete to design a poly-time $(\ln n - \varepsilon)$-approx for set cover. (Min # sets to cover all elts)

Recall: greedy algo is $\ln n$ - approx.

Next Lecture
Briefly outline the redn. from GapLabelCover $\rightarrow$ Gap Max Coverage. to show slightly worse factors of $\left(\frac{3}{4}+\varepsilon\right)$ hardness of Approx.

Given an instance $\mathcal{I} = \{G = (U, V, E), \{f_{u \to v}\}\}$ of GapLabelCover problem, we'll create an instance $\mathcal{I}' = \{X, \mathcal{S}, k\}$ of GapCoverage s.t.

$$(*)\ \begin{cases} \text{if} \quad Opt(\mathcal{I}) = 1 & \Longrightarrow \quad Opt(\mathcal{I}') = 1 \\ \quad\ \ Opt(\mathcal{I}') \leq \eta & \Longrightarrow \quad Opt(\mathcal{I}') \leq \frac{3}{4} + \varepsilon. \end{cases}$$

Here
Opt = fraction of satisfied edges

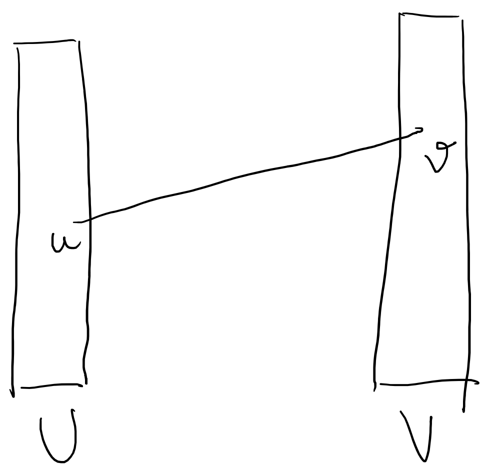
Here
Opt = fraction of covered elements.

From this,

NP-completeness of Gap Label Cover, we can get NP-completeness of Gap Coverage.

$\Rightarrow \left(\frac{3}{4}+\varepsilon\right)$ - hardness of approx for Max Coverage.

Todo
Come up with $\mathcal{I}'$ and prove $(*)$ for it.



From $\mathcal{I}$, we want $\mathcal{I}'$ such that there is some correspondence between assigning a label ℓ to u and picking a set in $\mathcal{I}'$.

We'll create $\mathcal{I}'$ such that

there is a set $S_{u,\alpha}$ for all $u \in U$, $\alpha \in [L]$

and similarly

$S_{v,\beta}$ for all $v \in V$, $\beta \in [K]$

In total

$$\begin{aligned} \#\text{sets} &= |U| \cdot L + |V| \cdot K \\ &= |U|(L+K) \\ &= n(L+K) \end{aligned} \quad \Bigg\} \because |U| = |V| = n \text{ in input graph.}$$

Next, we'll create some sort of association between edges in $\mathcal{I}$ with elements of $\mathcal{I}'$.

For each edge $(u,v) \in G$, we'll create a number of elements.

2^K elements corresponding to K-bit strings

We'll refer to these elements as

$e_x^{(u,v)}$ where (u,v) is edge
x is a K-bit

How many elements in $\mathcal{I}'$ have we created? string.

$\#$edges. $2^k = d \cdot n \cdot 2^k$

$\because$ graph is d - regular.

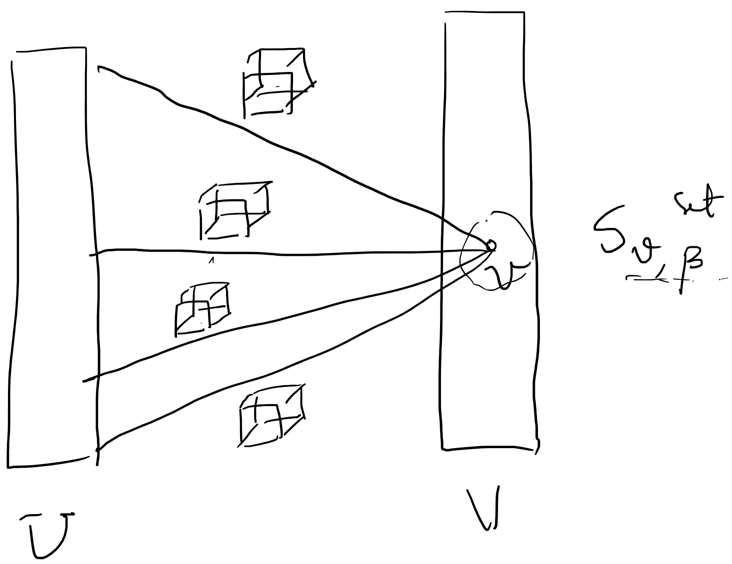
We've defined $\mathcal{X}$ (univers of elts) in $\mathcal{I}'$

" " $\mathcal{S}$ (coll. of sets).

Last: When does a set $S_{u,\alpha}$ cover an element?

lly, when does a set $S_{v,\beta}$ cover an element?

Let's start with $S_{v,\beta}$.



RULE 1: Firstly $S_{v,\beta}$ can only cover the elements on the

edges incident to v.

$\Rightarrow |S_{v,\beta}| \leq d \cdot 2^k$ (trivial bound)

Moreover,

consider edge (u,v) and all elts $e_x^{(u,v)}$ for $x \in \{0,1\}^k$

RULE 2

Let's make $S_{v,\beta}$ cover all elts where the β^{th} bit of $x = 1$.

In particular

$$e_x^{(u,v)} \in S_{v,\beta} \text{ iff } x[\beta] = 1$$

(β^{th} bit of x).

$\boxed{S_{v,\beta} \text{ is simply all such elts}}$

$$\Rightarrow |S_{v,\beta}| = d \cdot 2^{k-1}$$

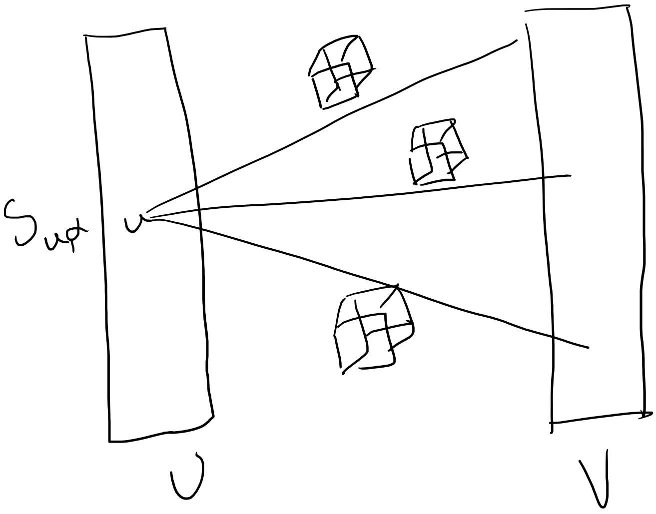
Similarly, sets for all $v \in V$, all $1 \leq \beta \leq K$.

Next we'll define sets for U

Consider $S_{u,\alpha}$ for $u \in U$, $\alpha \in [L]$.

$e_x^{(u,v)} \in S_{u,\alpha}$ iff $x[f_{u,v}(\alpha)] = 0$

(ie) the $f_{u,v}(\alpha)^{th}$ bit of $x = 0$.



$$|S_{u,\alpha}| = d \cdot 2^{k-1}$$

Also set k (in Gap k coverage) to be '2n'.

That defines I'. Why is it useful?

CLAIM ①

I is satisfiable, I' is fully coverable

(ie) $OPT(I) = 1 \Rightarrow OPT(I') = 1$

Proof.

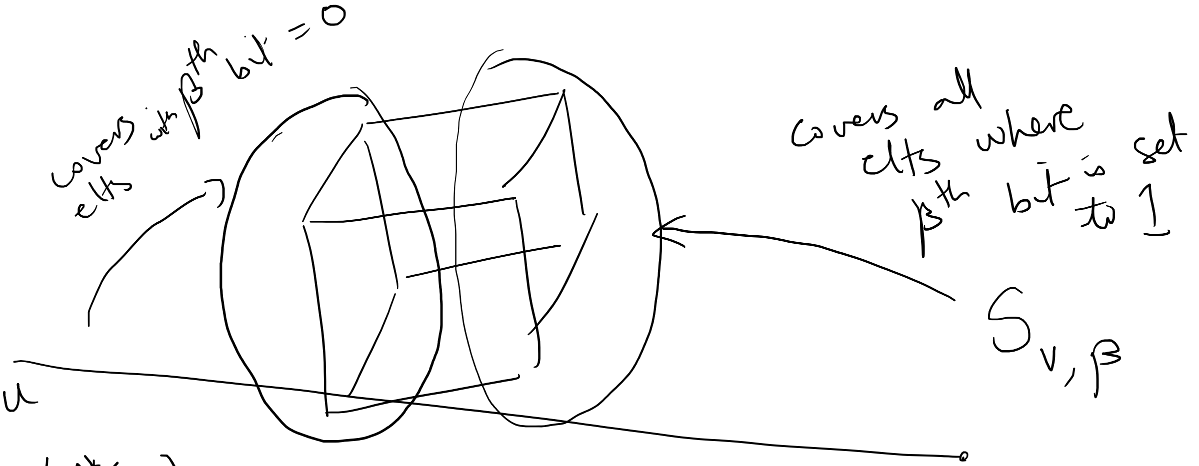
Let's consider the optimal labelling for I.

it can satisfy all edges.

(i.e.) $\ell^*(u)$, $\ell^*(v)$ are optimal labels
$u \in U$, $v \in V$

$\forall (u, v) \in G$, it holds that

$$\boxed{f_{u,v}(\ell^*(u)) = \ell^*(v)} = \beta$$
$(1 \leq \beta \leq K)$



$f_{u,v}(\ell^*(u)) = \beta$.

$\ell^*(v) = \beta$

Using this labeling ℓ^*, can we get a good sol^n for I' (GapKCoverage).

We can pick one set per vertex acc. ℓ^* labeling.

$\forall u \in U$, choose $S_{u,\ell^*(u)}$ and

$\forall\, v \in V$, choose $S_{v, \ell^*(v)}$.

sets selected $= 2n = k$.

$S_{v,\beta}$ covers all $e_x^{(u,v)}$ st $x[\beta] = 1$

$S_{u, \ell^*(u)}$ covers all $e_x^{(u,v)}$ st $x[\beta] = 0$.

$\Rightarrow$ together they cover all elts ☺

Remains to show:-

$$\mathrm{Opt}(I) \le \eta \Rightarrow \mathrm{opt}(I') \le \tfrac{3}{4} + \varepsilon.$$

Instead we'll show

$$\mathrm{Opt}(I') > \tfrac{3}{4} + \varepsilon \Rightarrow \mathrm{opt}(I) > \eta.$$

Way we'll show it:-

If there is a good sol^n for gap coverage covering $\geq \left(\tfrac{3}{4} + \varepsilon\right)$ fraction of elts, then we can recover a good labeling satisfying $\geq \eta$ fraction of edges.

High-level sketch :-

Let S^* be a good cover covering $\geq (\frac{\beta}{4}+\varepsilon)$ fraction of elts.

$$|S^*| = k = 2n.$$

Sps
Cheating assumption

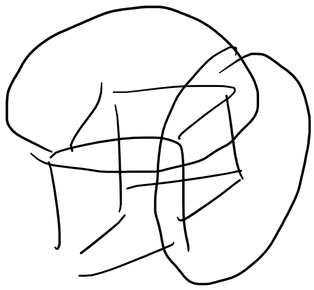
S^* is such that it picks one set per vertex of the graph G of I.

Then we ask : Can we get a good labeling for I.

If S^* contains $S_{u,\alpha}$, then assign label $\ell^*(u) = \alpha$

IIIly If S^* contains $S_{v,\beta}$ then assign label $\ell^*(v) = \beta$.

Then claim : $\geq 4\varepsilon$ fraction of edges need to have a satisfied labeling.



Any badly labeled edge can cover only $\leq 3/4$ fraction of elts.

But since S^* covers $\geq \left(\frac{3}{4}+\varepsilon\right)$ fraction of elts, there must be a good # of satisfied edges ☺

$$(1-\delta)\cdot\frac{3}{4} + \delta\cdot 1 = \frac{3}{4}+\varepsilon$$

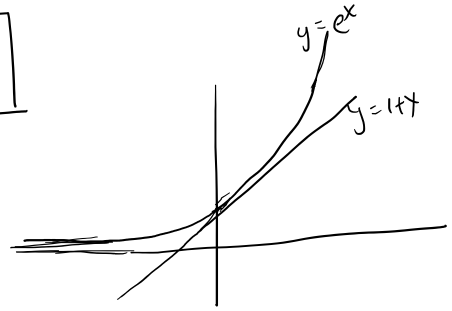
$$\frac{3}{4} + \frac{\delta}{4} = \frac{3}{4}+\varepsilon$$

$$\boxed{\delta = 4\varepsilon}$$

Useful Inequalities

09 February 2021 10:04

$$\boxed{1+x \leq e^x}$$



03/03/2021

- Cauchy-Schwarz Inequality
- Young's Inequality

Höl - der's Inequality

- Start with "online" load balancing/ Makespan

Cauchy schwarz

$$|\langle u, v\rangle| \leq \|u\|_2 \cdot \|v\|_2$$

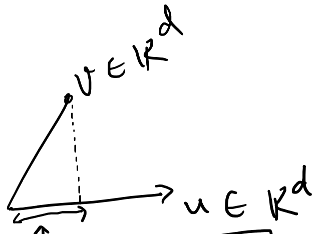
where $\|u\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_d^2}$

$$\langle u, v\rangle \equiv u \cdot v \equiv u^T v = \sum u_i v_i$$

$$\left(\sum u_i v_i\right)^2 \leq \left(\sum u_i^2\right)\left(\sum v_i^2\right)$$

Pictorial Proof

Assumed How to do Projections



$$\boxed{\frac{\langle u, v\rangle}{\|u\|} \leq \|v\|}$$

Young's Inequality :-

If p and $q \geq 0$ are st

$$\frac{1}{p} + \frac{1}{q} = 1 \quad \text{then}$$

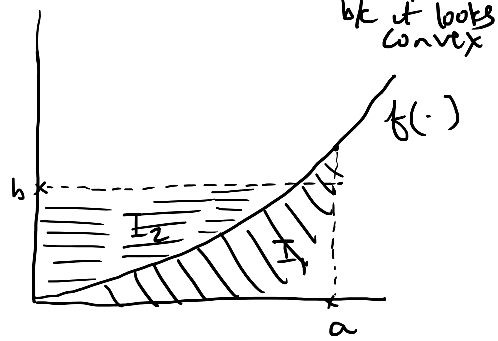
for all $a \geq 0$, $b \geq 0$,

$$\boxed{ab \leq \frac{a^p}{p} + \frac{b^q}{q}}$$

Proof

Let $f(x) = x^{p-1}$

↓ In example $p > 1$ b/c it looks convex



In geometric view,

$$ab = \text{Area of Rectangle}.$$

$$\leq I_1 + I_2.$$

Apply to $f(x) = x^{p-1}$

$$I_1 = \frac{x^p}{p}\Big]_0^a = \frac{a^p}{p}.$$

I_2 = Integral of the inverse function

$$y = x^{p-1}$$

$$y^{\frac{1}{p-1}} = x$$

$$f^{-1}(y) = y^{\frac{1}{p-1}}$$

$$I \quad \ldots \frac{1}{p-1} \Big]^b$$

$$I_2 = \left. \frac{y^{\frac{p}{p-1}}}{\frac{p}{p-1}} \right|_0^b$$

$$= \frac{b^{\frac{p}{p-1}}}{\frac{p}{p-1}} = \frac{b^q}{q}.$$

Recall

$\frac{1}{p} + \frac{1}{q} = 1 \Rightarrow q$ is exactly $\frac{p}{p-1}$

Hence we get $\boxed{ab \le \frac{a^p}{p} + \frac{b^q}{q}}$

For $p = q = 2$, this is famous

$2ab \le a^2 + b^2$ inequality

<u>Hölder's Inequality</u>

Can think of it as a generalization of Cauchy–Schwarz for l_p norm where $p \neq 2$.

Any vector $v \in \mathbb{R}^d$ has many <u>notions</u> of how "long" it is.

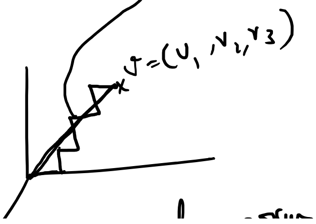
$\downarrow$

Each is called a norm.

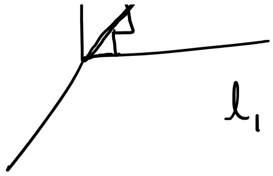
$v = (v_1, v_2, \ldots, v_d)$

l_2 norm $\|v\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_d^2}$

"distance as the bird would fly"



l norm (Manhattan Norm)



l_1 norm (Manhattan Norm)

$$= |v_1| + |v_2| + \cdots + |v_d|$$

l_p Norm $= \left(|x_1|^p + |x_2|^p + \cdots + |x_d|^p \right)^{1/p}$

Any function

$f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ is a NORM iff

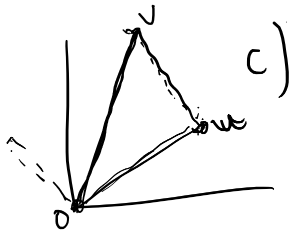
a) $f(\alpha \cdot v) = |\alpha| f(v) \quad \forall \alpha \geq 0, \ \forall v \in \mathbb{R}^d$

b) Δ^{le} inequality is true.

(ie) $\forall u \in \mathbb{R}^d, \ v \in \mathbb{R}^d$

$$f(v) \leq f(u) + f(v-u)$$

c). $f(v) = 0$ iff $v = 0$



EXERCISE

Verify that l_1, l_2 & l_p are indeed NORMS.

Hölders Inequality

Extension of Cauchy Schwarz for l_p NORMS.

In general, if

$v = (v_1, v_2, \ldots v_d)$

$u = (u_1, u_2, \dots u_d)$

Hölder

$$\boxed{\sum |u_i v_i| \leq \|u\|_p \cdot \|v\|_q}$$

if $\frac{1}{p} + \frac{1}{q} = 1$.

[One of the Motivations for calling $\|\cdot\|_q$ as the "dual Norm" of $\|\cdot\|_p$ whenever $q = \frac{p}{p-1}$.

Idea: let's try to simplify the problem without loss of generality

Can we assume that $\|u\|_p = 1$ & $\|v\|_q = 1$?

Yes, because

let $\hat{u}_i = \frac{u_i}{\|u\|_p}$, $\hat{v}_i = \frac{v_i}{\|v\|_q}$

Now, $\|\hat{u}\|_p = \|\hat{v}\|_q = 1$ (by scaling property)

$\Rightarrow$ If we show

$$\sum_i |\hat{u}_i \hat{v}_i| \leq 1 \quad (*)$$

$$\Rightarrow \sum \left| \frac{u_i}{\|u\|_p} \cdot \frac{v_i}{\|v\|_q} \right| \leq 1$$

$$\Rightarrow \sum |u_i v_i| \leq \|u\|_p \|v\|_q. \quad \blacksquare$$

Remains to show ⊛

Let's apply Young's Inequality inside each term

$$|\hat{u}_i \hat{v}_i| \leq \frac{|\hat{u}_i|^p}{p} + \frac{|\hat{v}_i|^q}{q}$$

Sum over i

$$\sum |\hat{u}_i \hat{v}_i| \leq \frac{\sum_i |\hat{u}_i|^p}{p} + \frac{\sum_i |\hat{v}_i|^q}{q}$$

$$= \frac{1}{p} + \frac{1}{q}$$

$$= 1. \quad \text{(dual norm)}$$

Stirling's Approximations

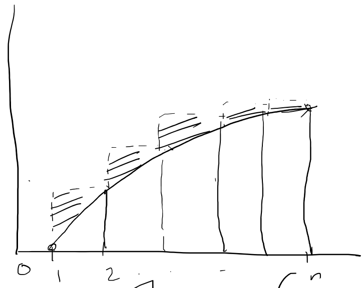
01 February 2021 09:38

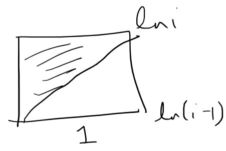
I $\boxed{n! \approx \sqrt{n}\cdot\left(\frac{n}{e}\right)^n}$

How?

$$Q = \ln(n!) = \sum_{i=1}^{n} \ln n$$

$$I = \int_1^n \ln x \, dx$$





$$I - Q = \text{shaded region} \approx \frac{1}{2}\left[\sum\left(\ln i - \ln(i-1)\right)\right]$$

$$\approx \frac{\ln n}{2}$$

$$\therefore \quad Q = I - \ln\sqrt{n}$$

$$= \left[x\ln x - x\right]_1^n - \ln\sqrt{n}$$

$$= n\ln n - n - \ln\sqrt{n}$$

$$= \ln\left(\sqrt{n}\cdot\left(\frac{n}{e}\right)^n\right)$$

$$\Rightarrow \quad n! \approx \sqrt{n}\cdot\left(\frac{n}{e}\right)^n$$

II $\boxed{\binom{n}{k} \approx \left(\frac{ne}{k}\right)^k}$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{\sqrt{n}\; n^n\; \cancel{e^k}\; \cancel{e^{n-k}}}{\sqrt{k}\; \cancel{e^n}\; k^k\; \sqrt{n-k}\cdot(n-k)^{n-k}}$$

$$\approx \left(\frac{n}{k}\right)^k \left[1 + \frac{k}{n-k}\right]^{n-k}$$

$$\approx \left(\frac{ne}{k}\right)^k \quad \boxtimes$$

Gaussian Tails

02 February 2021 14:37

$$Q(t) = \frac{1}{\sqrt{2\pi}} \int_t^\infty e^{-x^2/2}\, dx$$

$$Q(t) \le \frac{1}{\sqrt{2\pi}} \int_t^\infty \left(\frac{x}{t}\right) e^{-x^2/2}\, dx$$

$$= \frac{1}{t \cdot \sqrt{2\pi}} \int_t^\infty x e^{-x^2/2}\, dx = \frac{1}{t \cdot \sqrt{2\pi}} e^{-t^2/2}$$

$$= \frac{f(t)}{t}.$$

On the other hand

$$\left(1+\frac{1}{t^2}\right) Q(t) \ge \frac{1}{\sqrt{2\pi}} \int_t^\infty \left(1+\frac{1}{x^2}\right) e^{-\frac{x^2}{2}}\, dx$$

$$= \frac{1}{\sqrt{2\pi}} \left[-\frac{e^{-x^2/2}}{x} \right]_t^\infty = \frac{1}{\sqrt{2\pi}} \frac{e^{-t^2/2}}{t}$$

$$\Rightarrow Q(t) \ge \frac{t}{1+t^2} \cdot f(t)$$

$$= \frac{1}{t}\left[1 - \frac{1}{1+t^2}\right] f(t)$$

$$\ge \left(\frac{1}{t} - \frac{1}{t^3}\right) f(t)$$

Hence

$$\boxed{\left(\frac{1}{t} - \frac{1}{t^3}\right) f(t) \le Q(t) \le \frac{1}{t} \cdot f(t)}$$

Introduction to Linear Programming

01 February 2021 09:37

- Broad class of Problems can be modeled with linear programming

- n real valued variables $x_1, x_2, \dots, x_n$
  - m linear constraints over those variables

one linear objective function over them.

$\boxed{\text{either maximize (or) minimize Obj. fn. subject to all constraints}}$

$\boxed{\text{Useful in theory \& practice}}$

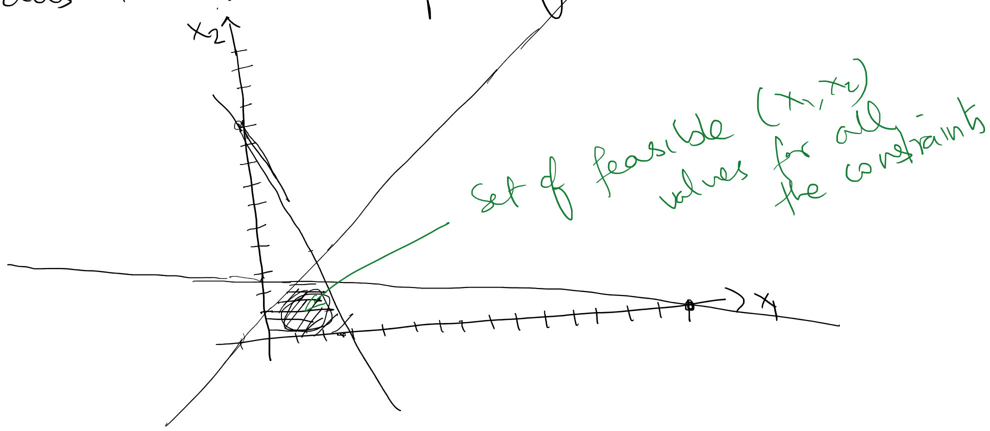
[Matousek
Understanding and Using
Linear Programming]

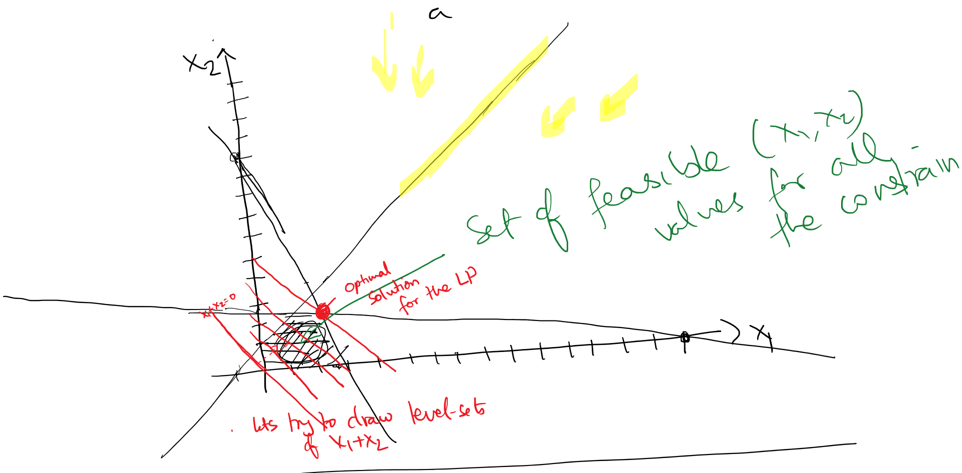
Max $x_1 + x_2$

$$x_2 - x_1 \leq 1$$
$$x_1 + 6x_2 \leq 15$$
$$4x_1 - x_2 \leq 10$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

{ LPs don't allow strict Inequalities }

How does the solution space of this look like?





General Form

$$\boxed{\begin{aligned} \min\ & c^T x \\ & a_1^T x \geq b_1 \\ & a_2^T x \geq b_2 \\ & \vdots \\ & a_m^T x \geq b_m \end{aligned}}$$

$$c^T x = \sum c_i x_i = \langle c, x \rangle = c \cdot x$$

$$c^T x = (c_1 c_2 \cdots c_n) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\sum c_i x_i$$

$$\Updownarrow$$

$$\boxed{\begin{aligned} \min\ & c^T x \\ & Ax \geq b \end{aligned}}$$

typically $m \geq n$

$A = m \times n$ matrix

rows of A correspond to the constraints

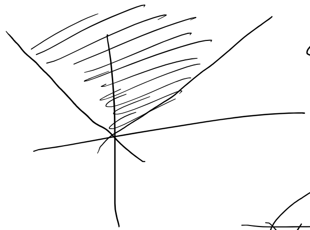
What does the soln space look like?

Intersection of 'm' halfspaces

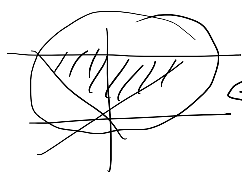
Polyhedron (Intersection of finitely many half spaces)

Polytope

Polyhedron which is also bounded



Polyhedron but not a polytope



polytope

Feasible set for any LP is

Feasible set for any LP is a polyhedron

How do we characterize optimal solutions of LP?

5th Feb 2021

$x_1, x_2, \ldots x_n \in \mathbb{R}$ are variables

$$\left\{ \begin{array}{ll} \max & c^T x \\ & Ax \le b \end{array} \right\}$$

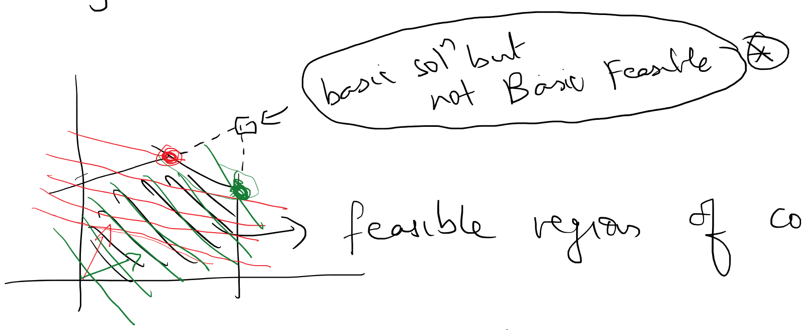
Optionally can separate out $x_i \geq 0$ type "non-negativity" constraints if they are present

Why study LPs.

1. Very general, can capture variety of problem
2. You can solve them efficiently

In theory & in practice

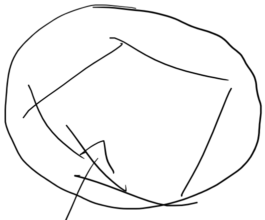
Why are we able to solve them so effectively?



feasible region of constraints,

For this lecture (and most of the course, & most applications)

Our feasible sets will be bounded



can be encapsulated inside some n-dimensional ball of finite radius

POLYTOPES
(bounded intersection of finitely many half-spaces)

LPs are nice to solve over polytopes because optimal solutions always occur at "corner points"

Q: How do we characterize a corner point?

Choose n out the m constraints and solve them @ equality

$$A_S \cdot x = b_S$$

notice the equality

$S \subseteq [m]$ of size n

$\max c^T x$
$m \updownarrow \; Ax \leq b$

$\left\{ \begin{array}{l} A \in \mathbb{R}^{m \times n} \\ b \in \mathbb{R}^m \\ c \in \mathbb{R}^n \end{array} \right.$

lets assume $m \geq n$

Sps $\det(A_S) \neq 0$

$\boxed{\text{then } \mathrm{sol}^n \text{ is } x^{(S)} = A_S^{-1} b_S}$

Such solution are called BASIC SOLUTIONS

Now, it might not satisfy the other constraints in $[m] \setminus S$

if $x^{(S)}$ additionally satisfies

all other constraints $\longrightarrow$ $A_{[m]\setminus S} \; x^{(S)} \leq b_{[m]\setminus S}$

$x^{(S)}$ is called a basic Feasible Solution

See (*) above for example of

See ⊛ above for example of basic soln which is not BFS.

{At most $\binom{m}{n}$ many BFS, can check in finite time & optimize}

There are algos which find the optimal BFS in poly time !!

Geometric Views of "Corner Points"

① VERTEX OF POLYTOPE P

$x \in P$ is a "vertex" if

$\exists$ objective function $c \in \mathbb{R}^n$

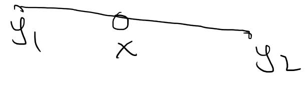
st $c^T x > c^T y \quad \forall\, y \in P,\ y \neq P$

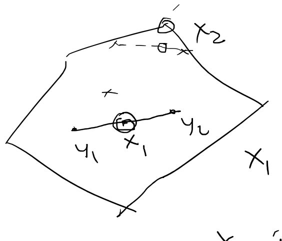
② Extreme Points of Polytope P

$x \in P$ is an extreme point

iff $\nexists\ y_1, y_2 \in P$, $y_1 \neq y_2$ st

$x = \alpha y_1 + (1-\alpha) y_2$ for some $\alpha \in [0,1]$



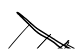


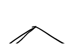
x_1 is not an extreme point

x_2 is an extreme point

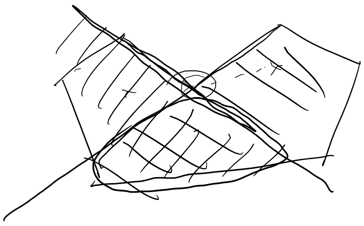
$$\boxed{\begin{array}{l}\forall \text{ polytope} \\ \text{Extreme Points} = \text{Vertices} = \text{BFS}^s\end{array}}$$

↑ Very useful THEOREM





Very useful THEOREM



← Such polygons will not arise as the feasible region of any LP

THM

Intersection of finitely many Halfspaces is convex
⇒ polytope P is convex

DUALITY

Useful way to understand optimal sol^ns of LPS without "optimizing the LP".

$$\text{Max} \quad 2x_1 + 3x_2$$

$$4x_1 + 8x_2 \le 12 \quad (1)$$

$$3x_1 + 2x_2 \le 4 \quad (2)$$

$$2x_1 + x_2 \le 3 \quad (3)$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Can we get good "upper bounds" on the optimal value of the LP without solving it?

Due to $x_1 \geq 0$ & $x_2 \geq 0$

$$2x_1 + 3x_2 \le 2x_1 + 4x_2 = \tfrac{1}{2}(4x_1 + 8x_2) \le 6$$

From constr (1)

⇒ Optimal Sol^n ≤ 6

Can we do better?

(1) + (3) gives

$$6x_1 + 9x_2 \le 15$$

$$\Rightarrow \boxed{2x_1 + 3x_2 \le 5}$$

$$\Rightarrow \boxed{2x_1 + 3x_2 \leq 5}$$

equivalently,

$\frac{1}{3}(1) + \frac{1}{3}(3)$

Since we're looking for upper bounds, we can try "dominating" the objective fn C by non-negative linear combination of the constraints, to get a best upper bound.

$$\max \quad c^T x$$

$$\left.\begin{array}{lll} y_1 & a_1^T x & \leq b_1 \\ y_2 & a_2^T x & \leq b_2 \\ \vdots & \vdots & \\ y_m & a_m^T x & \leq b_m \\ & x_1 \geq 0 & \\ & x_2 \geq 0 & \\ & \vdots & \\ & x_n \geq 0 & \end{array}\right\}$$

best such "upper bound" can be found by a linear program in itself.

Seeking multiplying factors $y_1 \cdots y_m \geq 0$

Coefficient of x_j in this combination

$$= \sum_{i=1}^{m} y_i \, a_{ij}$$

Want $\underbrace{c_j \leq \sum_{i=1}^{m} y_i \, a_{ij}}$ $\forall\, 1 \leq j \leq n$

$\Rightarrow \sum c_j x_j$ (for any feasible soln x)

$$\leq \sum b_i y_i$$

---

gives us DUAL program

$$\min \sum b_i y_i = b^T y$$

$$\boxed{\begin{array}{l} \max\ c^T x \\ Ax \leq b \\ x \geq 0 \end{array}} \underset{\text{dualize}}{\overset{\text{dualize}}{\rightleftarrows}} \boxed{\begin{array}{l} \min\ b^T y \\ A^T y \geq c \\ y \geq 0 \end{array}}$$

$Ax \le b$ $\quad$ ← dualize $\quad$ $Ay \geq c$
$x \geq 0$ $\quad\quad\quad\quad\quad\quad$ $y \geq 0$

Dual (Dual) = Primal

WEAK DUALITY THM

If x^* is optimal sol^n for primal &
y is <u>any</u> <u>feasible</u> sol^n for Dual

$$\boxed{c^T x^* \le b^T y}$$

{In form discussed $x \geq 0$ & we had $\le$ constraints}

Mechanical dual generator in most general form.

$$P = \max c^T x$$
$$a_i^T x \le b_i \quad \forall i \in I_1$$
$$a_i^T x = b_i \quad \forall i \in I_2$$
$$x_j \geq 0 \quad \forall j \in J_1$$
$$x_j \in \mathbb{R} \quad \forall j \in J_2$$

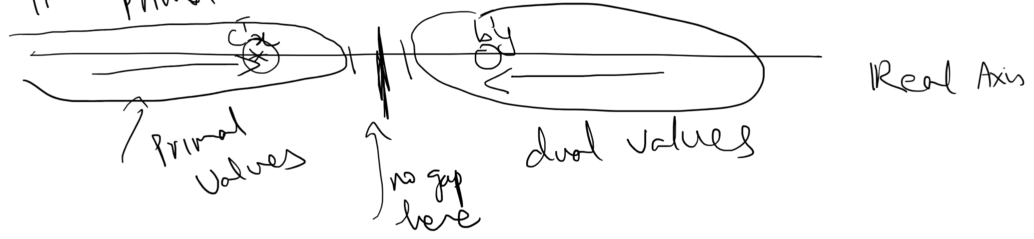
i^{th} row of A

$$D = \min b^T y$$
$$y_i \geq 0 \quad \forall i \in I_1$$
$$y_i \in \mathbb{R} \quad \forall i \in I_2$$
$$A_j^T y \le c_j \quad \forall j \in J_1$$
$$A_j^T y = c_j \quad \forall j \in J_2$$

j^{th} column of A

<u>Todo</u>: Practice duals for many LPs.

IF Primal is maximization & Dual is minimization



STRONG DUALITY THEOREM

Given P & D, One of 4 cases can occur :

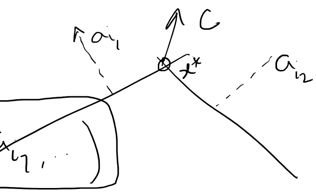
1. Either P & D are Infeasible
2. Either P is UNBOUNDED $\Rightarrow$ D is Infeasible
3. Either D is UNBOUNDED $\Rightarrow$ P is Infeasible
4. Both feasible & optimal values equal (ie) $\boxed{c^T x^* = b^T y^*}$

Proof of Strong Duality

let x^* be primal optimal for

$$\max \; c^T x \qquad Ax \le b$$

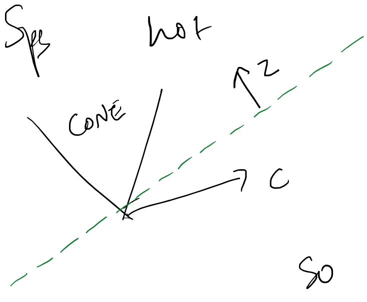
and let I be indices which satisfy tight inequality (ie) $a_i^T x^* = b_i \;\; \forall i \in I$



CLAIM :-

& $\boxed{c \in \text{Cone}(a_1, a_2, \ldots)}$

Sps not



$\exists$ hyperplane st

$z^T \cdot \text{CONE} \le 0$ &

$z^T c > 0$

so we can perturb

$x^* = x^* + \epsilon \cdot z$

and violate Optimality of x^*

Being in cone $\Rightarrow$

$c = \sum y_i^* a_i$

Set $y_i^* = 0 \;\; \forall i \notin I$ and so

$$A^T y^* = c$$

$$b^T y^* = \sum b_i y_i^* = \sum_{i \in I} y_i^* a_i^T x^*$$

$$= \langle x^*, \sum y_i^* a_i \rangle$$

$$= c^T x^* \quad \blacksquare$$

Some Useful probabilistic Inequalities
The study of random variables' behaviour

Markov's Inequality

If X is a non-negative random variable,
then $Pr[X \geq t\, E[X]] \leq \frac{1}{t}$

equivalently
$$Pr[X \geq v] \leq \frac{E[X]}{v}$$

Recall defn of $E[X] = \sum_v v \, Pr(X = v)$ or

$= \sum_{\substack{\text{all possible} \\ \text{outcomes } \omega}} val(X) \cdot Pr(\omega)$

also recall linearity of Expectation
$E[X_1 + X_2] = E[X_1] + E[X_2]$
regardless of (in)dependence of X_1 & X_2

How useful is Markov's Inequality
Sps we toss 1000 coins independently
each H wp $1/2$ &
T wp $1/2$

$X_i = \begin{cases} 0 & \text{if Tail} \\ 1 & \text{if Head} \end{cases}$ for i^{th} coin

$E[X_i] = \frac{1}{2}$

$E[X] = E[\Sigma X_i] = 500$

How likely / unlikely to get in excess of 750 ~~heads~~

Markov's Inequality gives

$Pr(X \geq 750) \leq \frac{500}{750} = \frac{2}{3}$.

Loose estimate (does not make use of fact that all X_i's are independent at all)

Proof of Markov's Inequality

$Pr[X \geq V] \leq \frac{E[X]}{V}$

Q. Can we make use of the independence of X_i's to get better estimates.

DETOUR

$E[X] = V$

$E[cX]$ for c constant $= cV$

$E[\cdot]$

but $E[X^2] \neq V^2$

Eg: $X = \begin{cases} -1 & w/p\ 1/2 \\ 1 & wp\ 1/2 \end{cases}$ $E[X] = 0$ but $E[X^2] = 1$

DEFINE

$$Var(X) = E\left[(X - E[X])^2\right]$$

$X = \begin{cases} 0 & wp\ 1/2 \\ 1 & wp\ 1/2 \end{cases}$ $E[X] = \frac{1}{2}$

$Var(X) = \frac{1}{4}$

$X - E[X] = \begin{cases} -1/2 & wp\ 1/2 \\ +1/2 & wp\ 1/2 \end{cases}$ $E[(X - E[X])^2] = \frac{1}{4}$

$$Var(X) = E\left[X^2 - 2\underbrace{X\,E[X]} + E[X]^2\right]$$

linearity of expectation $= E[X^2] - 2E[X] \cdot E[X] + E[X]^2$

$= E[X^2] - E[X]^2$

In 0/1 example $E[X^2] = 1/2$

$E[X]^2 = 1/4$

Let's go back to the coins example

(recall 1000 coins, trying to understand $Pr(\geq 750$ heads$)$)

$$X = \sum_{i=1}^{1000} X_i$$

let's make life easy a lot and consider slightly changed random variables

$Y_i = -1$ if TAIL for ith coin

$= 1$ if HEAD

$E[Y_i] = 0$

$Y = \sum Y_i$

Q: How are X & Y related?

A: $Y_i = 2(X_i - \frac{1}{2}) = 2X_i - 1$

$$\boxed{Y = 2X - n}$$

Trying to study deviations for $X \rightleftharpoons$ deviations for Y

e.g. $E[Y] = 0$

$\Pr(X \geq 750) = \Pr(Y \geq 500)$

$$\mathrm{Var}(Y) = E\left[(Y - E[Y])^2\right]$$

$$= E[Y^2]$$

$$= E\left[\left(\sum Y_i\right)^2\right]$$

$$= E\left[\sum Y_i^2 + \sum_{i \neq j} Y_i Y_j\right]$$

$$= \sum E[Y_i^2] + \sum_{i \neq j} \underbrace{E[Y_i Y_j]}$$

$\Pr(Y = y \mid X = x) = \Pr(Y = y)$ $\forall (x, y)$

If X & Y are "independent random variables"

$E[XY] = E[X] \cdot E[Y]$.

$$\boxed{= \sum^n E[Y_i^2] = n.}$$

$$\boxed{Var(Y) = \sum_{i=1}^{n} E[Y_i^2] = n}$$

$$Pr(Y \geq 500) \leq Pr(|Y| \geq 500) = Pr(Y^2 \geq 500^2)$$

$$\leq \frac{E[Y^2]}{500^2} \quad \text{MARKOV}$$

equal to Var(Y)

$$= \frac{1000}{500 \cdot 500}$$

$$= \frac{2}{500}$$

$$= \frac{1}{250}$$

by using "PAIRWISE independence" of X_i & X_j (Y_i & Y_j) we get much better bounds.

Chebyshev's Inequality

If X is any random variable,

$E[X] = \mu$

$Var[X] = E[X^2] - E[X]^2 = \sigma^2$

Then

$$Pr[|X - \mu| \geq t\sigma] \leq \frac{1}{t^2}$$

Proof: use Markov on R.V $Y = (X-\mu)^2$

WHY stop here?

These random variables (for the coins)
are not just pairwise independent

Any 3 of them are independent

In fact, Any subset of them are "

Can we use this fact ~~to~~
get better estimates?

Try to use
Markov's on
Y^{2K} for some good choice of k

$Y = \sum Y_i$ $\quad E[Y_i] = 0 \quad \forall i$

Y_i's are independent

Each Y_i : -1 wp $1/2$, 1 wp $1/2$ $\qquad E[Y] = 0$

$$\Pr(Y \geq 500) \leq \Pr(Y^{2k} \geq 500^{2k})$$
$$\leq \frac{E[Y^{2k}]}{500^{2k}}$$

Try to understand $E[Y^{2k}]$

$= E[(\sum Y_i)^{2k}]$

" be] dominated by

will be dominated by

$$\sum (Y_1^2 Y_2^2 \cdots Y_k^2)$$

$$\binom{n}{k} \cdot \frac{(2k)!}{\underbrace{2 \cdot 2 \cdots 2}_{2^k}}$$

$E[Y^{2k}]$ will be approximately $\binom{n}{k} \frac{(2k)!}{2^k}$ (*)

Stirlings Approximation For Factorials

$$\binom{n}{k} \approx \left(\frac{ne}{k}\right)^k$$

$$n! \approx \sqrt{n} \cdot \left(\frac{n}{e}\right)^n$$

(*) gives $\quad \frac{n^k \cdot e^k}{k^k} \cdot \frac{(2k)^{2k}}{e^{2k} \cdot 2^k}$

$$\frac{n^k \cdot \cancel{e^k}}{\cancel{k^k}} \cdot \frac{2^{2k} \cdot k^{2k}}{e^{2k} \cdot \cancel{2^k}}$$

$$= \left(\frac{2nk}{e}\right)^k$$

$$\Pr(Y \geq t \cdot \sqrt{n}) \leq \frac{E[Y^{2k}]}{(\ldots)^{2k}}$$

$$(t\sqrt{n})^{2k}$$

$$= \frac{\left(\frac{2nk}{e}\right)^k}{t^{2k} \cdot n^k}$$

$$= \left(\frac{2nk}{e n t^2}\right)^k$$

$$= \left(\frac{2k}{e t^2}\right)^k$$

I am free to choose k, we'll optimize to minimize RHS.

We'll just set $k = t^2$

$$\Pr(Y \geq t\sqrt{n}) \leq \left(\frac{2}{e}\right)^{t^2}$$

SUMMARY

Markov

$$\Pr(X \geq t\, E[X]) \leq \frac{1}{t}$$

Chebyshev

$$\Pr(|X - E(X)| \geq t\sigma) \leq \frac{1}{t^2}$$

Chernoff $\Pr(|X - E(X)| \geq t\sigma) \leq \exp(-t^2)$

X to decomposable as sum of indep RVs

$\Pr($ 750 heads $) \leq$ .0000

$Pr(\text{750 heads}) \leq .0000$

SUMMARY

If X is non-negative R.V

Markov's Inequality

$$Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda}$$

Multiplicative deviation from than mean $E[X]$

Chebyshev's Inequality

X is any R.V, with Expectation $E[X]$ & Variance $E[X^2] - E[X]^2 = \sigma^2$

$$Pr[|X - E[X]| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

Chernoff Bounds

If X is the SUM of independent & bounded random variables, the X concentrates sharply around its mean

Popular Forms

1. If $X = \sum X_i$ each $X_i \in [0,1]$

a) $Pr[X \leq E[X](1-\delta)] \leq e^{-\frac{E[X]\delta^2}{2}}$ $\forall\, 0 \leq \delta \leq 1$

b) $Pr[X \geq E[X](1+\delta)] \leq e^{-\frac{E[X]\delta^2}{2+\delta}}$ $\forall\, \delta \geq 0$

② if $X = \sum X_i$, where X_i's are independent random variables with $|X_i| \le M$

and $Var(X) = E[X^2] - E[X]^2 \le \sigma^2$

$$Pr\left[|X - E[X]| \geq T\right] \le \exp\left(\frac{-T^2}{\sigma^2 + \frac{MT}{3}}\right)$$

BERNSTEIN'S INEQUALITY

for some intuition,
sps we plug in $T = \lambda\sigma$
& we ignore $\frac{MT}{3}$ (often this will be small)

$$Pr\left(|X - E[X]| \geq \lambda\sigma\right) \underset{\approx}{\le} \exp\left(-\lambda^2\right)$$

Try these inequalities to see what they yield for the coins problem

TODO

(1000 coins, what is probability of ≥ 750 heads)
independent

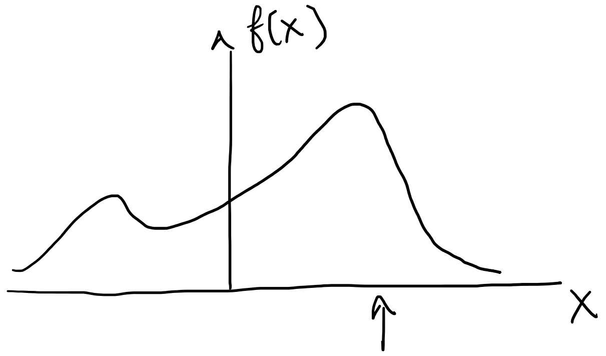
More on probability will be later —
Gaussian random variables.

Gaussian Random Variable

05 April 2021 10:01

Question

X is a random variable x over real values and has some distribution

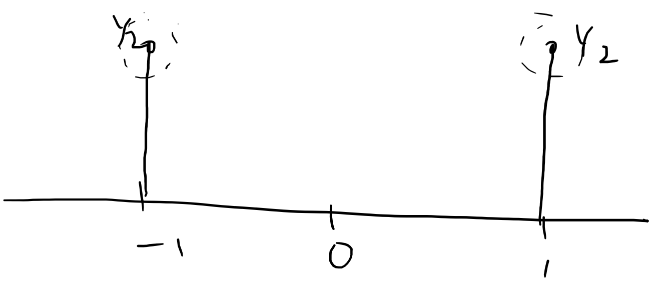


Suppose we consider $Y = X_1 + X_2$ where X_1 and X_2 are 2 independent copies of X.

What does the distribution of Y look like?

EXAMPLE

$$X = \begin{cases} +1 & \text{w.p } 1/2 \\ -1 & \text{w.p } 1/2 \end{cases}$$



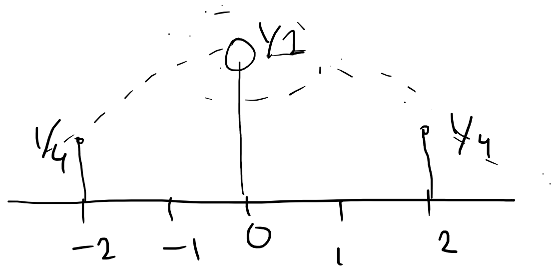
← bi-modal distribution (2 peaks).

$Y = X_1 + X_2$

$\Pr(Y=0) = \frac{1}{2}$

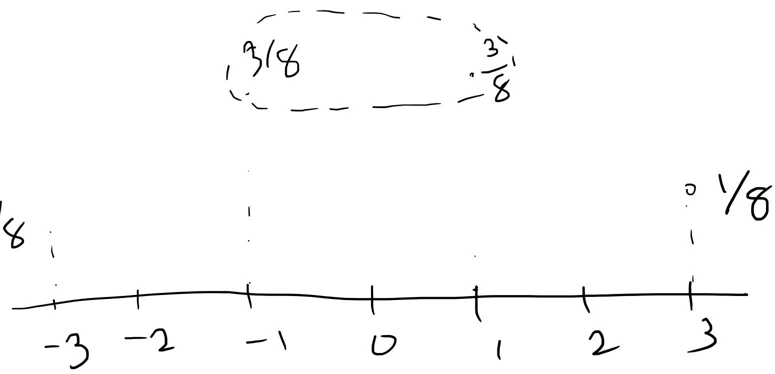
$\Pr(Y=1) = 0$

$\Pr(Y=2) = \frac{1}{4}$



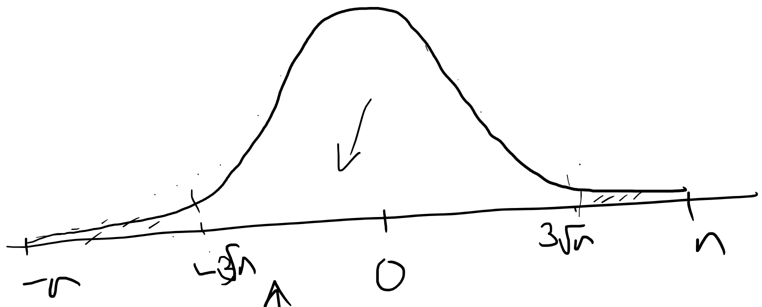
[distribution of Y is different from that of X.]

$Y = X_1 + X_2 + X_3$ (similarly)



looks different, but closer to having one peak

Eventually, you add enough times, you'll end up with the following type of distribution $Y = X_1 + X_2 + \dots + X_n$



Normal distribution

Moreover

This behaviour is not just for ± 1 R.V's.

Central Limit Theorem

Let X be any random variable.

Let $Y = X_1 + X_2 + \dots + X_n$ be n copies of X (independent).

then Y is almost distributed like a "Gaussian Distribution".

Y will have mean $E[Y] = n.E[X]$

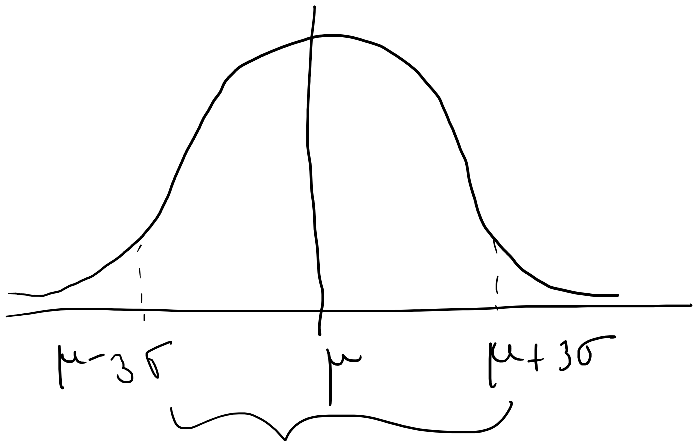
and variance $Var[Y] = \sum Var[X_i]$
$= n.Var[X]$.

DEFN:

$X \sim N(\mu, \sigma^2)$ is a gaussian

$X \sim N(\mu, \sigma^2)$ is a gaussian random variable with mean μ and variance σ^2 iff it has the following probability density function :-

$$f_X(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(t-\mu)^2}{2\sigma^2}\right)$$



Most of the probability $(> 95\%)$ is in the interval

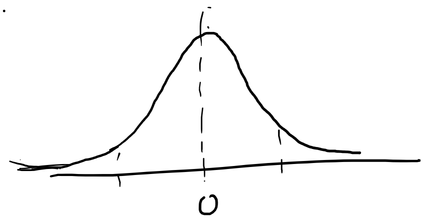
$$[\mu \pm 3\sigma].$$

Standard Normal Distribution

$X \sim N(0,1)$ Mean 0
Variance 1

$-t^2/2$

$$f_X(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$$



These $\exp(-\frac{t^2}{\sigma^2})$ type bound occurred in Chernoff bounds also

[Gaussians are deeply interconnected with these inequalities].

For any gaussian random variable

$$X = N(0, 1).$$

$$E[X] = \int_{-\infty}^{\infty} t \cdot \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \, dt$$

$$= \frac{1}{\sqrt{2\pi}} \left[e^{-t^2/2} \right]_{-\infty}^{\infty}$$

$$= 0$$

$$Var[X] = E[(X-\mu)^2] = E[X^2]$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} t^2 e^{-t^2/2} \, dt$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty} t^2 e^{-t^2/2} \, dt$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \underbrace{t}_{u} \underbrace{\left(t e^{-t^2/2} \, dt \right)}_{dv}$$

$$= \frac{1}{\sqrt{2\pi}} \left[uv \Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} v \, du \right]$$

$$= \frac{1}{\sqrt{2\pi}} \left[t \cdot e^{-t^2/2} \Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty} e^{-t^2/2} \, dt \right]$$

$$= \frac{1}{\sqrt{2\pi}} \left[0 + \int_{-\infty}^{\infty} e^{-t^2/2} \, dt \right]$$

$$= \int_{-\infty}^{\infty} \underbrace{\frac{1}{\sqrt{2\pi}} e^{-t^2/2} \, dt} = \underline{1}$$

because f is a pdf.

also (more importantly) because

$$\boxed{\int^{\infty} e^{-t^2/2} \, dt = \sqrt{2\pi}}$$

$$\boxed{\int_{-\infty} e^{-t^2/2}\, dt = \sqrt{2\pi}\,.}$$

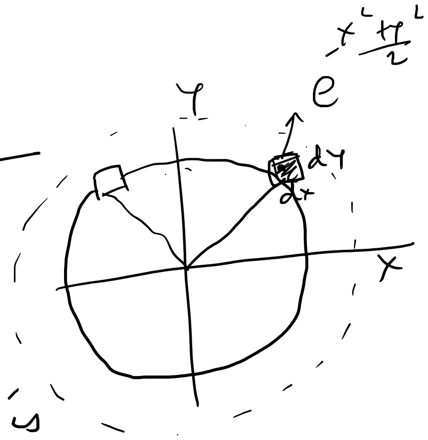
Proof

let $$\underline{I} = \int_{-\infty}^{\infty} e^{-x^2/2}\, dx$$

$$\underline{I} = \int_{-\infty}^{\infty} e^{-y^2/2}\, dy$$

$$\underline{I}^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{x^2+y^2}{2}}\, dx\, dy.$$

because it is spherically symmetric, (ie) the value of the integrand is invariant under rotation due to its form of $\exp\left(\frac{-(x^2+y^2)}{2}\right)$



We can use polar coordinates ☺

$$I^2 = \int_{r=0}^{\infty} \int_{\theta=0}^{2\pi} e^{-\frac{r^2}{2}}\, r\,dr\, d\theta$$

$$I = \int_{r=0} \int_{\theta=0} e$$

$$= 2\pi \int_{r=0}^{\infty} r e^{-r^2/2} dr$$

$$= 2\pi \left[-e^{-r^2/2} \right]_0^{\infty}$$

$$= 2\pi .$$

$$\Rightarrow \boxed{I = \sqrt{2\pi}}$$

- We have understood one gaussian well.
- what about sums of independent gaussians ?

$X_1 = N(0,1)$ and independent
$X_2 = N(0,1)$
what will the distribution of $X_1 + X_2$ look like ?

let $Y = X_1 + X_2$

$$E[Y] = E[X_1] + E(X_2) = 0.$$

$Var[Y] = Var[X_1] + Var[X_2] = 2.$

Distribution of Y ?

$\downarrow$

Y will also be gaussian !

— $N(0, 2)$

INTUITION
① (CLT intuition

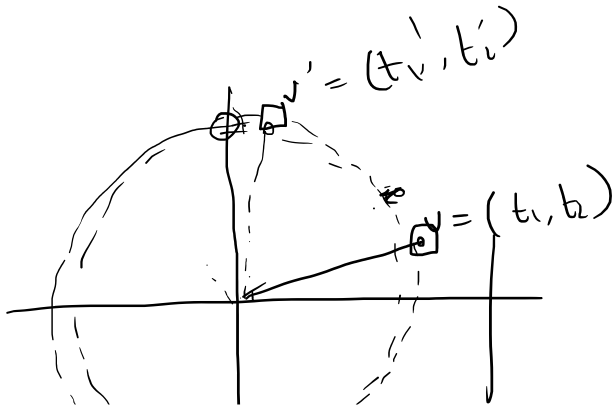
X_1 can be thought of as sum of many small RVs)

X_2 similarly

$\Rightarrow$ Y has gaussian form, with mean 0 and Var = 2.

INTUITION ②

lets look at joint distribution of (X_1, X_2)



$$f_{X_1,X_2}(t_1,t_2) = \frac{1}{2\pi} \exp\left(-\frac{t_1^2+t_2^2}{2}\right)$$

Spherical symmetry.

Joint distribution of (X_1, X_2) is the same as joint distribution of

$$\left(\frac{X_1+X_2}{\sqrt{2}}, \frac{X_1-X_2}{\sqrt{2}}\right)$$

rotation of (X_1, X_2) by 45°

$\Rightarrow$ distribution of $\left(\frac{X_1+X_2}{\sqrt{2}}\right)$ is identical to that of X_1

$= N(0,1)$

$\Rightarrow$ $\text{dist}(X_1+X_2) = \sqrt{2}\,N(0,1)$

$= N(0,2)$.

More generally

More generally

$X_1, X_2, X_3, \dots X_d$ are all $N(0, 1)$

then

$y = \sum a_i X_i$ is distribution as $\boxed{N\left(0, \sum a_i^2\right)}$

for scalars $a_1, a_2, \dots, a_d$.

Given a gaussian $X = N(0,1)$

for $v \geq 0$ $\Pr(X \geq v)$ is "tail estimate"

$$\Pr\left(|X| \geq v\right) = 2 \cdot \underbrace{\int_v^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \, dt}$$

$$2\left(\frac{1}{v} - \frac{1}{v^3}\right) f(v) \leq \Pr\left(|X| \geq v\right) \leq \frac{2}{v} f(v)$$

pdf @ v.
