The Set Cover Problem: LP-Rounding f Approximation

08 February 2021 09:59

Universe of elements $[n]$ elements U

Collection of subsets of U

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

Goal:

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

meaning

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

Feasible soln: $X = \mathcal{S}$

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

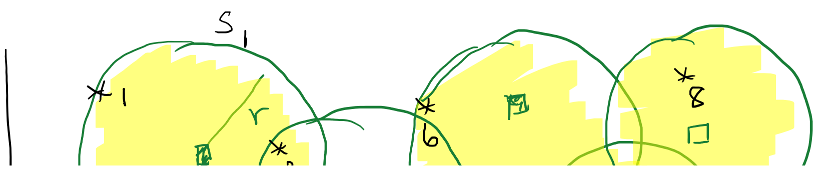
Objective fn

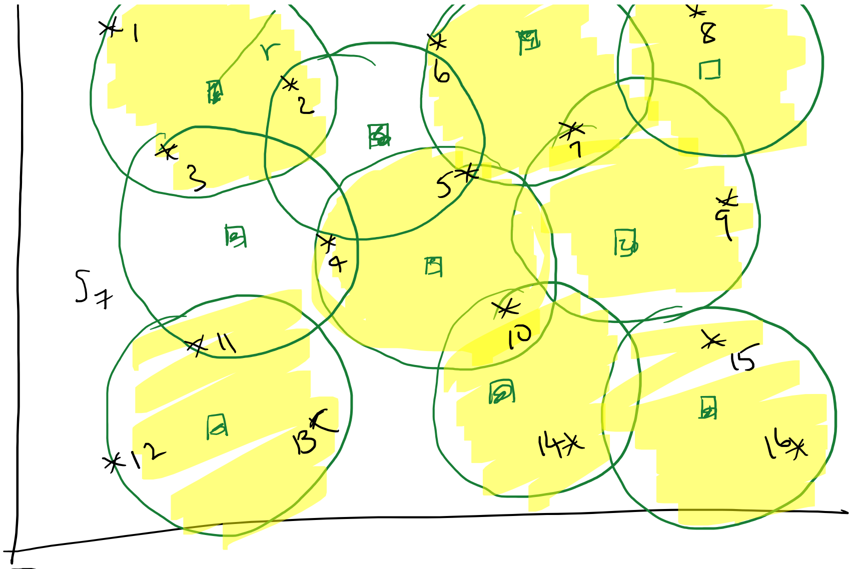
Find X of min. cardinality $|X|$

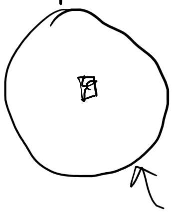
(ie) pick as few sets to cover the universe

EXAMPLE

Geometric Disk Coverage





each  is a "set"

disc

each * is a neighbourhood in a city w/ ≥ 100 population

Goal

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

In this example

$$\mathcal{U} = \{ e_1, e_2, \ldots e_{16} \}$$

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

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

$$S_7 = \{ e_3, e_4, e_{11} \}$$ 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 (ie) 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, choose 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

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

m variables & n constraints

Fact ①

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

Let x^* denote the optimal solution

Lemma ①

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

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 an 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.

Set Cover: Primal Dual f Approximation

08 February 2021 10:56

Last class we saw an LP-based f-approximation

- Main drawback:

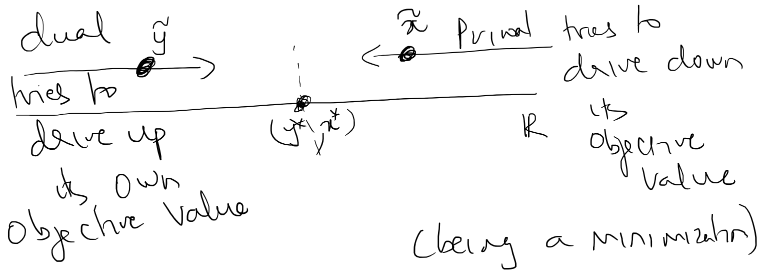
  LP^s, while efficient (polynomial time) are slow for large datasets.

- Search online: running time of best LP solver?

Today's lecture

Use LP^s conceptually to design faster f-Apx Algo.

PRIMAL LP	DUAL LP
$\text{Min} \sum w_S x_S$	$\text{Max} \sum_{e \in U} y_e$
$\sum_{S : e \in S} x_S \geq 1 \quad \forall e \in U$	$\sum_{e \in S} y_e \leq w_S \quad \forall S$
$x_S \geq 0 \quad \forall S \in \mathcal{S}$	$y_e \geq 0 \quad \forall e$
Variables $x_S \; \forall S \in \mathcal{S}$	Variables $y_e \quad \forall e \in U$
Constraints for each elt $e \in U$	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 <u>any</u> feasible dual,

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

<u>In words</u>

$\boxed{\text{Any feasible dual sol}^n \text{ gives a good lower bound on OPT.}}$

<u>ALGORITHM</u>

Initialize $F = \phi$

↑ solution

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

<u>DUAL LP</u>

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

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

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

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)

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

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

<u>Observations</u>:-

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

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

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]

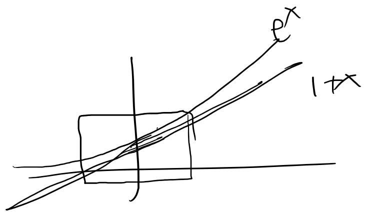
Fix e:

$$\Pr[\text{elt } 'e' \text{ is not covered in } 1 \text{ round}]$$

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

$$\le 1/e$$

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



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

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

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

to get

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

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

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

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

$= Pr\left[\begin{array}{l} e_1 \text{ is uncovered or } e_2 \text{ is uncovered} \\ \quad \text{or} \ \ldots \ e_n \text{ is uncovered} \end{array}\right]$

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

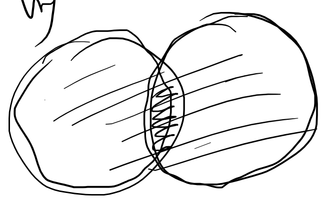
$\leq n \cdot \frac{1}{n^2}$

$= \frac{1}{n}$

$Pr(A \cup B) \leq$

$\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$ 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 $\leq 4\ln \sum w_S x_S^*$ 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.

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 $R \neq \emptyset$
  - 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.

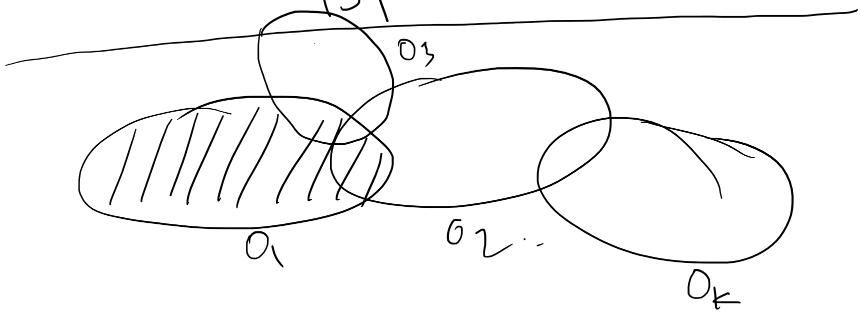
$$OPT = \sum_{S \in O^*} w_S$$

& suppose $|O^*| = k$

Claim: the first set alg includes satisfies

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

$$\frac{\cdots_S}{|S|} = \frac{\cdots}{n}$$



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

$= W_S$.

LEMMA

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

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

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

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

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

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

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

↓

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

↓

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

↓

f could be very large in general

↓

Useful when $f \gg \log n$ {

$O(\log n)$ - approximation (LP + Randomized Rounding)

↓

$\Theta(\log n)$ - approximation (greedy algo).

}

↓

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 $\frac{w_S}{|R \cap S|}$
  - update $R = R \setminus S$

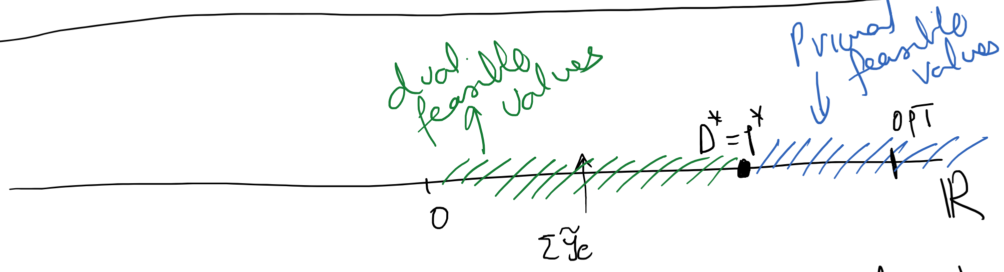
Recap LP Relaxation & Dual for Set Cover

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

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

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

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.

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

BACK to greedy :-

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

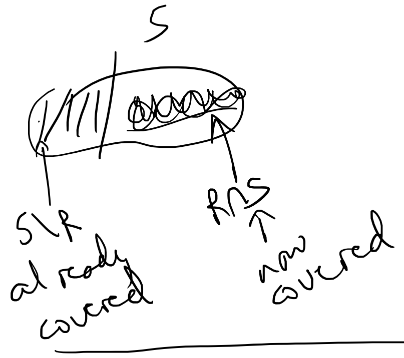
ITS cost $= w_{S_1}$

IT covers $|S_1|$ elts.

$\Rightarrow$ we can try to set $y_e = \frac{w_{S_1}}{\lambda |S_1|}$ 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

$\boxed{e_1 \quad e_2 \quad e_3 \cdots \quad e_\ell}$ 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. p...)

Obs (2)

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

Similarly,

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

& in general

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

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

So,

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

$$= \frac{w_S}{\lambda}\left[\frac{1}{l} + \frac{1}{l-1} + \dots + 1\right]$$

$$= \frac{w_S}{\lambda} \cdot H_l$$

Harmonic#(l) $\approx \ln(l)$.

So we can set

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

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

y_e ... choice of λ

$$\text{Cost(Greedy)} = \lambda \cdot \sum \tilde{y}_e$$
$$\leq \lambda \cdot D^* \quad (\tilde{y}_e \text{ is dual feasible})$$
$$= \lambda \, p^*$$
$$\leq \lambda \cdot OPT$$

$\boxtimes$

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

$\boxtimes$

Vector Discrepancy: The Power of Linear Programming

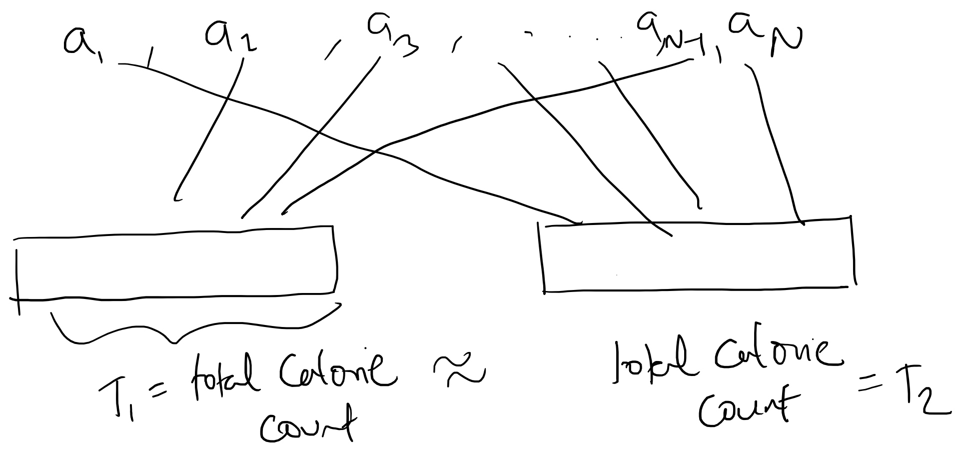
16 February 2021 09:04

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.



In other words,

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

Possible Algorithms :-

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

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

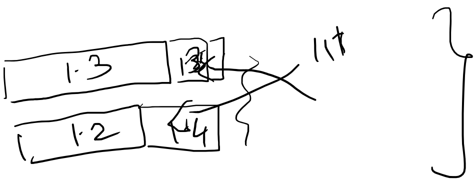
"or some variant of this"

/

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

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: $|T_1 - T_2| \le \text{Max wt} \le 1.$

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?

Optimistic Goal:

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

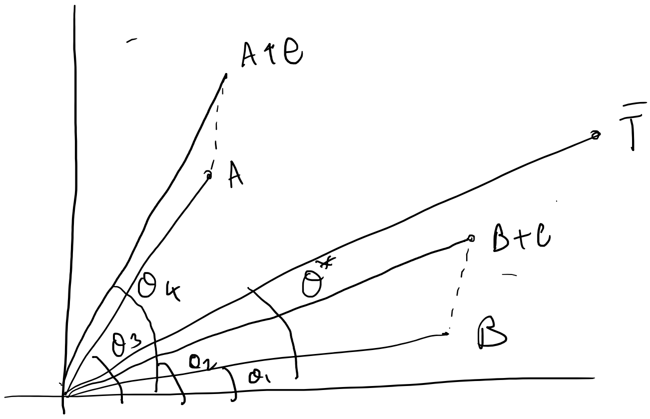
$$\left\| \begin{pmatrix} \frac{\sum A_i}{2} \\ \frac{\sum B_i}{2} \end{pmatrix} - \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \right\|_2$$

Q1 Why ℓ_2 distance?

Q2 what item is next? any item?

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

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



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

I don't know what analysis we get 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

$$\min \lambda$$
$$\left|\sum a_i x_i\right| \le \lambda$$
$$\left|\sum b_i x_i\right| \le \lambda$$
$$x_i \in \{-1, 1\}$$

Can't hope to solve this efficiently b/c Integer Programming is NP-hard.

Relax to

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

Linear Program! Can solve in poly-time

Abs. Value constraints

$\sum a_i x_i \le \lambda$

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

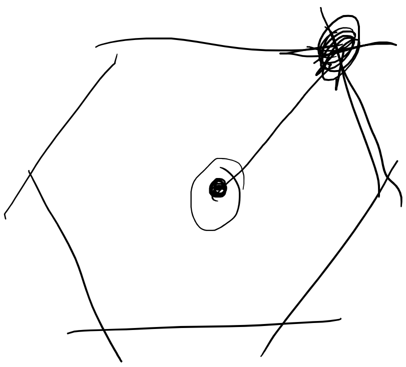
.. can be written as

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.

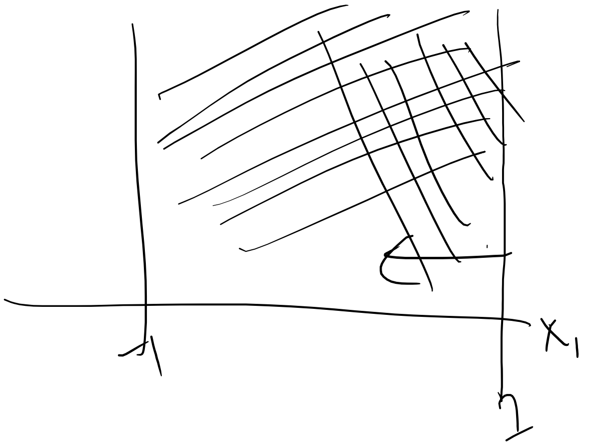


All $(0,0,\ldots 0)$ is a feasible point, but its in the interior of the polytope.

<u>But</u>, 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.

$\boxed{\begin{array}{l}\text{Just "round" the 2 fractional variables to } \pm 1 \\ \text{Total Harm done to constraints is} \le 2 \text{ in total!!}\end{array}}$

Example of a situation where non LP based algos are hard to think about,

but LPs make it super easy!