Lecture 14 +15+16

Core Algorithms

1, ... n $(x_1, \ldots x_n) \in \{0,1\}^n$

$p_1 \ldots p_n$

$w_1 \ldots w_n$.

N-V algo for TSP
- $O(n^3 \phi)$
- in expectation.

maximize $p_1 x_1 + \ldots + p_n x_n$

subject to $w_1 x_1 + \ldots + w_n x_n \leq W$

$x_i \in \{0,1\} \quad \forall i$

Fractional knapsack
→ where fractions of items can be stored

Relaxation:

$x_i \in [0,1] \quad \forall i$

ie x_i - allowed to be a real value.

Greedy for frac. knapsack

Sort items in descending order of p_i/w_i ratio

1 2 ... n

$p_i/w_i \geq \frac{p_{i+1}}{w_{i+1}}$

Add items as long as weight does not exceed capacity.

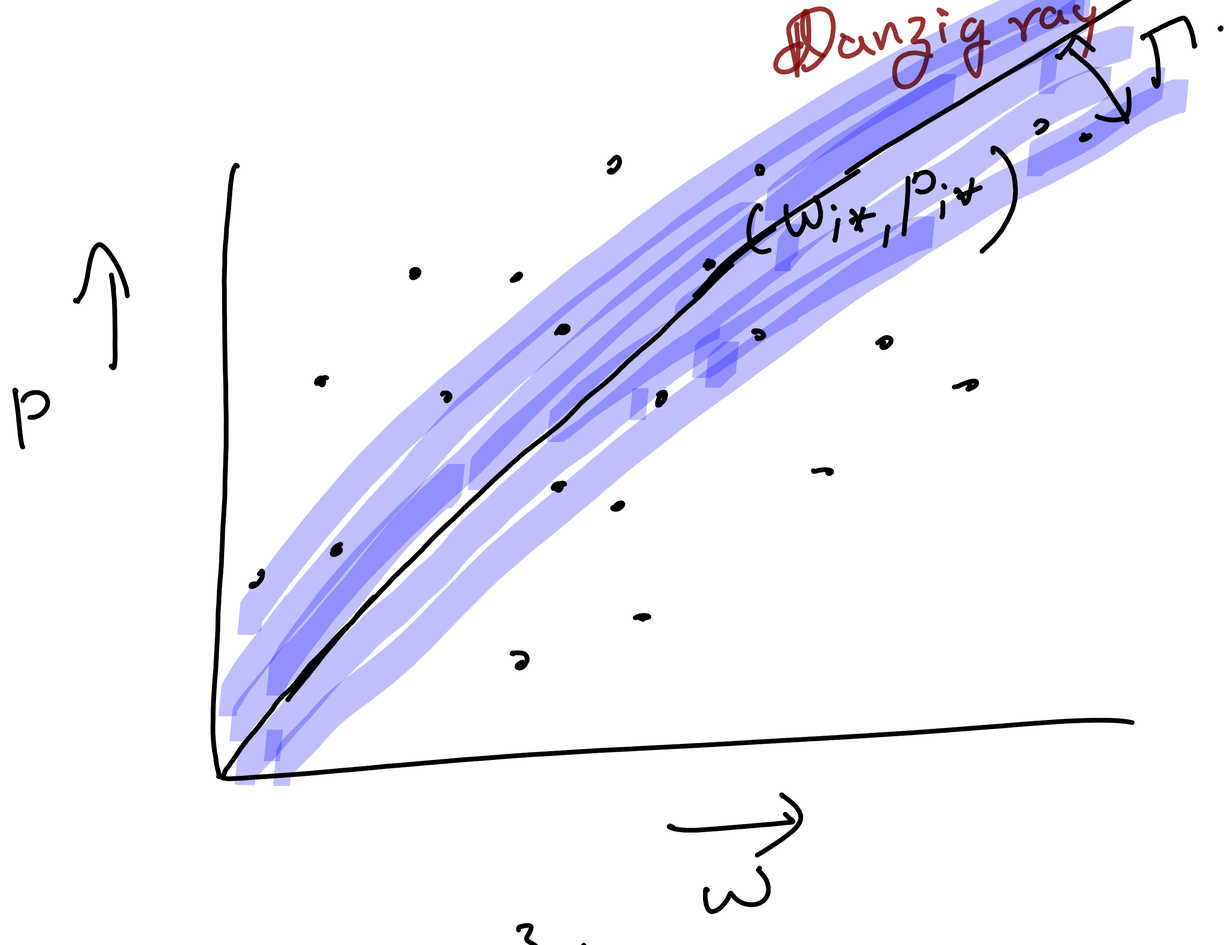
i^*-1

$x_1, x_2 \ldots x_{i^*-1}$

obs:- The solution is optimal & there is at one most one item i^* s.t $x_{i^*} \notin \{0,1\}$

(w_{i^*}, p_{i^*}).

$\hat{x}$ be an optimal solution.



Stmt: if $w_1 \cdots w_n$ & $p_1 \cdots p_n$ are ϕ-perturbed then $\exists$ Π s.t # of items within a distance of Π from the Danzig ray is $O(n^{\frac{1}{3}})$

$\therefore$ Expected running time = $Sort(n) + (n^{\frac{1}{3}})^3 \cdot \phi$.

$\leq O(n \log n)$.

Details see Röglin's notes.

A quick review of Complexity classes

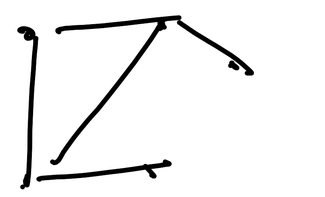
NP: non-deterministic polynomial time.

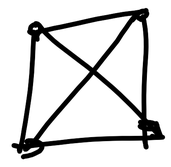
Set of all problems that have efficiently verifiable soln/witnesses.

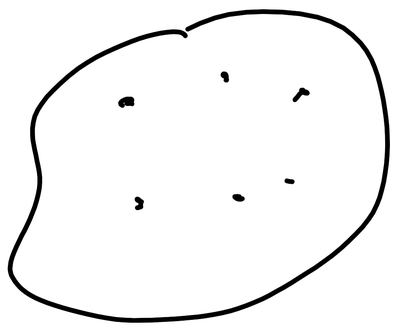
e.g. CLIQUE problem.

i/p = $G = (V, E)$ & $K > 0$

o/p: YES iff G has a clique on K vertices







witness for YES instance

A subset $S \subseteq V$, $|S| = K$
s.t $\forall\, i \neq j \in S$, $(i,j) \in E$.

$\therefore$ CLIQUE $\in$ NP

Vertex Cover
Dominating set
Hamiltonian cycle

Given G & K is there a path of length K

NP-hard

A problem B is NP hard if:
there is a poly time algo for B then for every problem $A \in NP$, we have a polynomial time algorithm.

Note:- CLIQUE, DomSet, VC, HamCycle, k-path, k-cycle
Knapsack, TSP, clustering.
All are NP hard problem. (in fact NP-complete).

For us:- No NP-complete problem is likely to have a polynomial time algorithm. ie $NP \neq P$.
P≠NP hypothesis

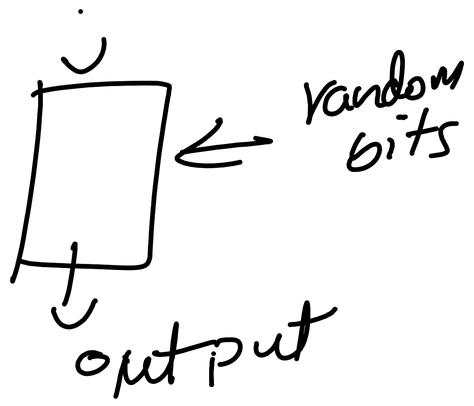
Suppose an algorithm A used random coin tosses-

Monte Carlo & Las Vegas.

runtime expectation.
o/p: guaranteed

runtime guaranteed
o/p: expection (ie possibility of errors).

TSP-implementations
Concorde-



Definition A problem B is said to be in ZPP (Zero error prob. poly time)
if it has a randomized algo which has expected poly time & outputs correct value.

Weakened P vs NP

Lecture - 15

Weakened P vs NP

NP ≠ ZPP.

Knapsack: has smoothed poly time algo

TSP ? - 2OPT - does not give exact soln.

Steiner tree ?

K-means ?

Strongly NP - hard

A problem B (a binary optimization) is said to B strongly NP hard if the restriction when $|c_i| \in [0, n^c]$ for some $c>0$ integers. remains NP hard.

$$\max_{\min} \; c_1x_1 + \cdots + c_nx_n$$

Subj: . . .

$x_1 \ldots x_n \in \{0,1\}$.

Example:- TSP when the edge weights are 1 or 2 remains NP hard.

∴ TSP is strongly NP-hard.

Ex:-

① Show that ~~TSP~~ can be formulated as a binary optimization problem.

② Suppose there is a poly time algorithm that solves TSP where weights are 1 or 2. Show that

this algo. can be used to solve the HamCycle problem.

Thm: Let Π be a binary opt. problem which is strongly NP-hard. N – input size n – # of variables.
Then there does not exist an algo for Π whose expected run. time is bounded by $poly(N, \phi)$ for ϕ – perturbed instances with coefficients from $[-1, 1]$ unless $NP \subseteq ZPP$.

Corollary:- TSP cannot have smoothed poly time algo unless $NP \subseteq ZPP$.

Pf:- A be an algo for ϕ- perturbed instances of Π

$\forall f_1 \cdots f_n$ $\qquad$ $f_1 \cdots f_n$

s.t $\mathbb{E}[\text{time}(A, y)] = poly(N, \phi)$

$y \leftarrow \phi$ perturb

Now, Π is strongly NP hard, Let Q be a poly s.t the special case of Π where $|c_i| \leq Q(N)$ remain NP-hard.

[eg: for TSP $Q(n) = 2$ suffices]

Π: minimize
$c_1 x_1 + \cdots + c_n x_n$
subj. to: . . .

I be an input instance of Π of length N and integer

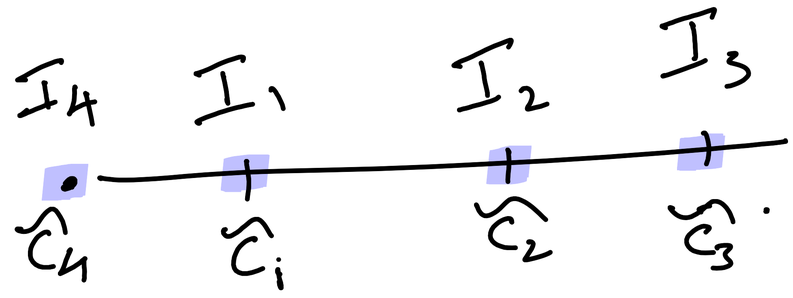
Coeff $c_1, \ldots c_n$, $|c| \le Q(N)$

integers

Step 1: Scale the coeff $\hat{c_1} \ldots \hat{c_n}$ s.t $\hat{c_i} \in [-1, 1]$

$C = \max_i \{|c_i|\}$, $\hat{c_i} = c_i / C$.

Obs: Solutions remain the same



Lecture - 10

Set: $\phi = 2nC$

For each i, choose an interval I_i of length $\frac{1}{\phi}$ in $[-1, 1]$

s.t $\hat{c_i} \in I_i$

ϕ perturbed instance.

choose $c_1', c_2', \ldots, c_n'$ where c_i' is sampled uniformly at random from I_i

Density for c_i': $f_i(x) = \phi$.

Note: c_i' is ϕ-perturbed

I' be the new instance.

Algorithm

input I

① Obtain I'

② Run A on I', Let x^* be the soln.

③ Output x^*.

Time

$\rightarrow O(n)$

$\rightarrow$ Expected $p(n)$ time.

Expected poly

Correctness:- Show that x^* is also an optimal soln for I

pf:- Suppose not. Say x is an optimal soln for I

but x^* is not.

$c = (c_1, \cdots, c_n)$

$$cx^* - cx \geq 1 \quad (\text{as } c_i \text{ - integers})$$

$\therefore \quad \tilde{c}x^* - \tilde{c}x \geq \frac{1}{C}$ (by scaling). $\tilde{c} = (\tilde{c_1}, \cdots, \tilde{c_n})$ $\rightarrow$ (A)

we have $|\tilde{c_i} - c_i'| \leq \frac{1}{\phi} \quad \forall i.$

$c'x^* = \sum c_i' x_i^*$

$c'x = \sum c_i' x_i$

$$\therefore \quad c'x^* - c'x \geq \tilde{c}x^* - \tilde{c}x - \frac{n}{\phi}$$

$$= \tilde{c}x^* - \tilde{c}x - \frac{n}{2nC} = \tilde{c}x^* - \tilde{c}x - \frac{1}{2C} \geq \frac{1}{2C}$$ by (A)

ie x^* is not an optimal soln for I' — contradiction

$\therefore$ x^* must also be opt for I.

$\therefore$ opt

Knapsack $\Longleftarrow$ psly (N, C) - algorithm, where $C = \max\{|c_i|\}$.

$W \rightarrow$ max wt

$O(W \cdot n^2)$

$\hookrightarrow$ pseudo linear time

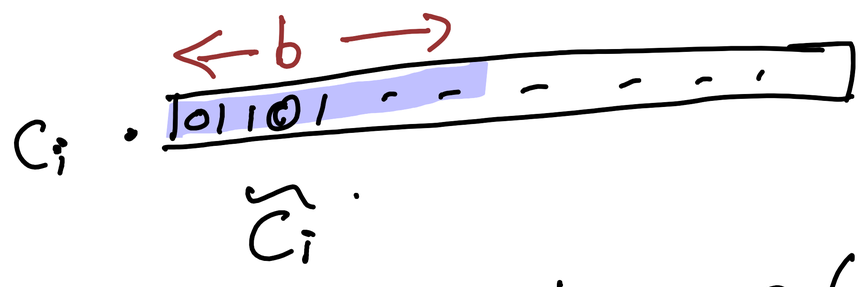
Thm Let Π be binary optimization problem such that it has a pseudo linear time algorithm. Then Π has an expected poly time algo on ϕ-perturbed instances.

pf:- A_p is a pseudo linear time algo for Π

time$(A_p) = O(N^k \cdot C)$ → cont
↳ weight

Assumption: coeff are from $[-1, 1]$.

a/b - $\max\{|a|, |b|\}$



$c_1 \dots c_n$

$\tilde{c}_1 \dots \tilde{c}_n$ } - can be solved in time $O(N^k \cdot 2^b)$

$b = O(\log N)$