ECS 122A Lecture 15 Jasper Lee

Complexity classes

Lecture *slightly* imprecise.

Decision problems : Boolean function
$f(x) \mapsto T/F$

Complexity class P - Set of decision problems computable in poly$(|x|)$ time.

Examples
- Does there exist MST of weight ≤ 5?
- Is there a winning permutation in midterm problem?

Definition (Class NP) nondeterministic polynomial

A problem f is in NP if there exists a deterministic alg $V(x,w)$ and polynomials p, q s.t.

1. $V(x,w)$ runs in $p(|x|+|w|)$ time.
2. For all x s.t. $f(x)=1$, $\exists\, w$ of length $q(|x|)$ s.t. $V(x,w)=1$
3. For all x s.t. $f(x)=0$, $\forall w$, $V(x,w)=0$.

x is called "instance"
w "witness"

Examples

SAT - Given Boolean formula, is there a satisfying assignment?

$$V(\underbrace{x}_{\text{Formula}}, \underbrace{w}_{\text{assignment}})$$ ← Check if assignment makes formula true.

3-colouring : Given undirected, unweighted graph G, can we colour vertices by $\{0, 1, 2\}$

Colouring

$c: V \to \{0,1,2\}$ s.t. for all $(u,v) \in E$, u, v have different colours?

Graph

$$V(\underbrace{x}_{\text{Graph}}, \underbrace{w}_{\text{Colouring}})$$ ← Check if colouring violates any edge

Every problem in P

$V(x, w)$ — Ignore w and just compute $f(x)$!

Informally, NP is the set of problems where it is possible to check solns efficiently

vs P : can find solns efficiently.

<u>Definition</u> (NP-hard) Slightly informal

A problem f is NP-hard if for every problem $g \in NP$, g can be reduced to f.

Includes non-decision problems like opt prob

(f is harder than every $g \in NP$).

Q: Very stringent defn, do NP-hard problems even exist?

<u>Theorem</u> (Cook-Levin)

SAT is NP-hard.

ECS 120 probably proves it.

The OG NP-hard problem, everything else derived from SAT.

<u>Definition</u> (NP-complete)

A decision problem f is NP-complete if $f \in NP \cap NP\text{-hard}$.

(NP-complete problems are the hardest problems in NP)

Corollary $P = NP$ iff $\exists$ NP-complete problem w/ poly-time alg.

Approx Alg

Many opt problems are NP-hard!

Examples:

- Max Independent Set
- Max Clique
- 0-1 knapsack ← Surprising, since we had $O(Cn)$ DP alg

"Pseudopolynomial" algorithm

But C is capacity, takes $\log C$ bits to specify in input.

Life lesson: if you can't beat the game, change the game.

Idea: Compute soln not too far from opt.

Alg (2-approx for 0-1 knapsack)

Hacky greedy alg

1. Sort items in non-increasing order of $\frac{r_i}{w_i}$
2. Compute smallest i s.t. $\sum_{j=1}^{i} w_i > C$

3. Pick better of $\sum_{j=1}^{i-1} r_j$, r_i can change to r_{max}

Thm If OPT is opt reward, then alg gets obj $\geq OPT/2$

Proof

Idea: Compare w/ fractional knapsack opt

Let OPT_{frac} be opt for fractional knapsack for same instance,

achieved by $\rho_1 = \cdots = \rho_{i-1} = 1$

$$\rho_i = \frac{C - \sum_{j=1}^{i-1} w_j}{w_i}$$

So $\sum_{j=1}^{i-1} r_j + \rho_i r_i = OPT_{frac} \geq OPT$

By averaging argument, $\max\left(\sum_{j=1}^{i-1} r_j, \rho_i r_i\right) \geq \frac{1}{2} OPT$ $\square$

≤ 1

Q: Is approx factor of 2 the best we can do in poly time?

No :)

Notion: Fully Polynomial Time Approx Scheme

Given any $\epsilon > 0$, find $(1-\epsilon)$-approx soln in $\mathrm{poly}(\frac{1}{\epsilon}, n)$ time.

integer reward, real weights

Idea: Use different 0-1 knapsack DP, plus <u>rounding</u>.

that takes $O(r_{max}\,\mathrm{poly}(n))$ time.

DP: $T[i, r]$ = min weight subset of items $1..i$ w/ reward at least r

$$T[i,r] = \min(T[i-1, r],\ w_i + T[i-1, r-r_i])$$

Size $n \times \sum r_i \le n \times n \times r_{max}$

Rounding: Need to make sure r_{max} is $\mathrm{poly}(\frac{1}{\epsilon}, n)$

Let $k = \epsilon\, r_{max} / n$

Then consider $r'_i = \left\lceil \frac{r_i}{k} \right\rceil k$

The number of reward values $\le r_{max}/k = \frac{n}{\epsilon}$

So runtime $= O(n^3/\epsilon)$ for rounded DP.

Approx ratio can be calculated to be $(1-\epsilon)$.

Inapproximability

Some NP-hard optimisation problems have no good approx alg!

Travelling Salesman Problem (TSP):

Given an undirected weighted graph G, find a min weight "Hamiltonian" cycle.

Hamiltonian Cycle: Cycle on graph visiting every vertex exactly once.

NP-hard just to find a Ham cycle, so TSP is also NP-hard.

Thm (Inapproximability of TSP)

For any $\alpha \in (1, 2^n)$, there is no poly time α-approx alg for TSP unless $P = NP$.

Proof

Suppose there is a poly time α-approx alg for TSP.

Given unweighted graph G, construct new graph $G' = (V', E')$

where $V' = V$

$E' = E$ w/ weight 1

plus $(V \times V) \setminus E$ w/ weight $\alpha n + 2$

If G has Hamiltonian cycle

then G' TSP must be a Ham Cycle on G

(any non-G edge adds $\alpha n + 1$ cost to TSP, $> \alpha \cdot$ length of cycle)

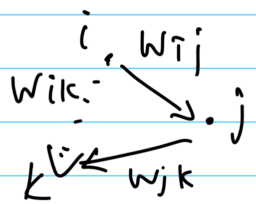
If G has no Ham cycle,

G' TSP must contain non-G edge $\square$

Problem solving lesson:

If you can't beat the game, look for rules / assumptions you missed.

Metric TSP: Graph weights satisfy triangle inequality.



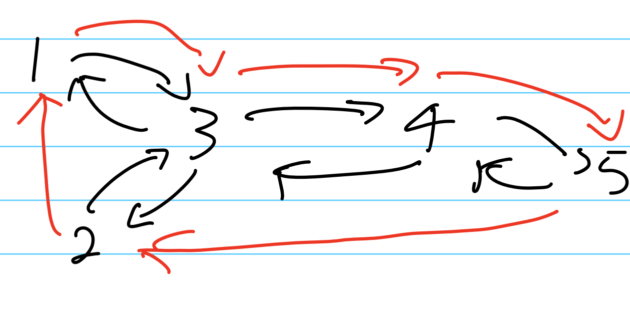
$w_{ik} \le w_{ij} + w_{jk}$ for all i, j, k

<u>Alg</u> (2-Approx for metric TSP)

1. Compute MST T of graph G
2. Compute DFS tour of T.
3. Turn DFS tour into Hamiltonian cycle by skipping visited vertices

<u>Example</u> MST + Tour

Ham cycle



Thm Alg returns Ham cycle w/ cost at most 2 times min TSP tour.

Proof

Consider min TSP tour. Delete any edge, this path is a spanning tree.

So $w(\text{min TSP tour}) \geq w(MST)$

Now $w(\text{alg output}) \leq 2 \cdot w(MST)$ $\square$