Lecture 23 (08/11/24) - AA

P1: Travelling Salesperson problem (with Triangle Inequality)

- A salesman must visit 'n' cities. Starting from the hometown (city 1), the salesman wants to create a tour by visiting every city exactly once and finishing in city at which the tour started.
- The salesman incurs a non-negative cost $c(i,j)$ to travel from city i to city j. The salesman wants to create a $\boxed{\text{Tour}}$ of minimum total cost.
  → Hamiltonian cycle (Travel every vertex exactly once)

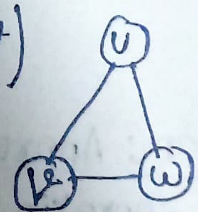
Input: Undirected complete graph G with non-negative cost assigned to every edge

Output: The goal is to find a Hamiltonian cycle (Tour) with minimum cost. Let $A \subseteq E$ be the set of edges in the Tour

$$c(A) = \sum_{e \in A} c_e$$

Important Property: The cost function satisfies triangle inequality

$$c(u,v) \leq c(u,w) + c(w,v)$$



Approx-TSP-Tour (G, c):

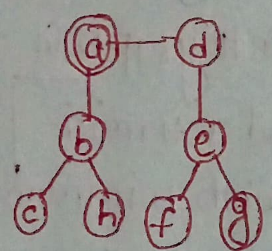
1. Select a vertex $r \in V$ as root
2. Construct a minimum spanning tree T of G starting from r
3. Compute preorder traversal of T and store the vertices in H according to when they have first visited in the preorder traversal
4. Return H as the tour

Ex.

Input: A complete graph on vertices $\{a, b, c, d, e, f, g, h\}$

After Step 2

↳ MST T



Preorder traversal → $\{a, b, c, h, d, e, f, g\}$

H

$\{a \to b \to c \to h \to d \to e \to f \to g\}$ Final tour

Theorem: Approx-TSP-Tour algorithm is a polynomial time 2-approximation algorithm.

Proof:

Let H^* be an optimal tour

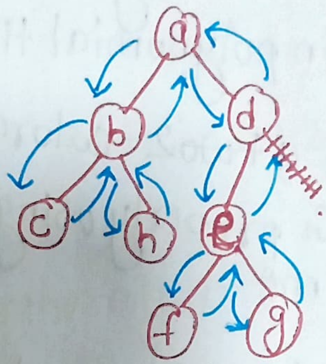
Removing one edge from H^* results in a spanning tree (path). ~~Let T be the MST~~

Let T be the MST computed by the algorithm.

$c(T) \leqslant c(H^*)$ —— ①

Let $\boxed{w}$ be the $\boxed{\text{full walk}}$ of $\boxed{\text{MST } T}$

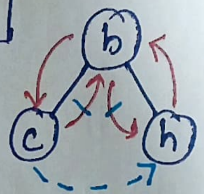
$W = a-b-c-b-h-b-a-d-e-f-e-g-e-d-a$



$c(W) = 2c(T)$ —— ②

From ① and ②

$\boxed{c(W) = 2c(T) \leqslant 2c(H^*)}$



Transformation

$c(c,h) \leqslant c(c,b) + c(b,h)$

After repeatedly apply the transformation (application of triangle inequality) we will get: H

$H' \leftarrow H + c(g,a)$

$c(H') \leqslant c(W) \leqslant 2c(H^*)$

$\Rightarrow \boxed{c(H') \leqslant 2c(H^*)}$ proved.

P2: Coloring 3-colorable graph

Δ = maximum degree of a vertex in G

$(\Delta+1)$ - coloring

Fact 1: $(\Delta+1)$ - coloring for any graph G can be done in polynomial time

Fact 2: Given a ~~two~~ 2-colorable graph G, we can color G properly using 2-colors in polynomial time

Input: A 3-colorable graph $\boxed{G}$

Consider $v \in G$

Neighbors of v : $N(v)$ → 2-colorable (Bipartite)

Approx-3-color(G) → G(V, E) | $|V| = n$

1. $G' \leftarrow G$
2. while there exists a vertex v with $\boxed{\text{degree} \geq \sqrt{n}}$ in G'
3. select $\boxed{\text{3 new colors}}$ of
4. Color $\boxed{v}$ and $\boxed{\text{neighbours' } v}$ using the $\boxed{\text{3 new colors}}$
5. Remove v and neighbors of v from G'
6. EndWhile
7. Color G' using $\sqrt{n}$ colors

$\Delta = \sqrt{n} - 1$

Theorem: Approx-3-color uses at most $4\sqrt{n}$ colors

Proof: Every iteration of the loop selects 3 new colors further in every iteration we are removing atleast $\sqrt{n}$ vertices. Therefore maximum number of iteration of the loop is less than or equal to $\frac{n}{\sqrt{n}}$

Maximum number of colors used in complete loop is $\leq 3\sqrt{n}$

Total colors used by the algorithm $\leq 3\sqrt{n} + \sqrt{n}$
$= 4\sqrt{n}$

Approximation ratio $= \frac{4\sqrt{n}}{3}$

P3: The set cover problem

- An instance (X, F) of the set cover problem consists of a finite set X and a family F of subsets of X such the every element of X belongs to atleast one subset in F

$$X = \bigcup_{S \in F} S$$

- We say that a subset S cover the element in S.

- The goal is to find a minimum size subset $C \subseteq F$ whose members cover all of X.

$$X = \bigcup_{S \in C} S$$

Size of C = Number of sets in C.

$$\left[H(n) = 1 + \frac{1}{2} + \frac{1}{3} \dots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k} = \underbrace{\ln(n) + O(1)}_{\text{Approx } \log n} \right]$$

Greedy-SET-Cover (X, F) : $|X|, |F|$

1. $U \leftarrow X$ — Uncover
2. $C \leftarrow \phi$ — Cover
3. while $U \neq \phi$
4. Select a set $S \in F$ that maximize $|S \cap U|$ — $|X| \cdot |F|$
5. $U = U - S$
6. $C = C \cup \{S\}$
7. End while
8. Return C

min $\{|X|, |F|\}$

$$T(n) = O(|X| \cdot |F| \cdot \min\{|X|, |F|\})$$

let,

$n = |X|$

Analysis:

Proof Sketch

- → ~~For~~ In every iteration, set selected by greedy algorithm increase the solution size C by 1.
- → Therefore we assign cost of 1 to every set picked by the algorithm.
- → Further, we distribute the 1 unit cost uniformly among the elements in the set ~~th~~ among those covered for the first time

$S = \{\cancel{1}, \cancel{2}, 3, 4, 5\}$ — $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$

$\boxed{\text{Covered by previous choice}}$

- Let $|C|$ be the ~~cost~~ size of greedy algorithm and let $|C^*|$ be the size of any optimal solution

~~c(x)~~ c_x denotes the cost assigned to any element $x \in X$

$$\sum_{x \in X} c_x = |C| \quad \text{—— (1)}$$

- Let S_i be the set selected by the greedy algorithm in the 'i' th iteration

$$i = 1, 2, 3 \ldots |C|$$

- If an element x is covered for the first time by S_i

$$c_x = \frac{1}{|S_i - \underbrace{(S_1 \cup S_2 \ldots \cup S_{i-1})}_{\text{Already covered}}|}$$

- In optimal solution C^*, each element $x \in X$ must be covered by atleast one set

$$\underbrace{\sum_{S \in C^*} \sum_{x \in S} c_x}_{\substack{\text{Taking set 1 by 1 from } C^* \text{ and calculating} \\ \text{the costs of elements}}} \geq \sum_{x \in X} c_x \quad \text{—— (2)}$$

From (1) and (2)

$$\boxed{\sum_{S \in C^*} \sum_{x \in S} c_x \geq |C|} \quad \text{—— (4)}$$

$$\sum_{x \in S} c_x$$

Some set S belongs to F

- Consider any set $S \in F$. Consider the i^{th} iteration in the greedy algorithm.

Let, $u_i = \left| S - \underbrace{(S_1 \cup S_2 \cup \ldots \cup S_i)}_{\text{Covered already}} \right|$ = Number of elements in S left uncovered after the i^{th} iteration

$u_0 = |S|$

- Let $\boxed{k}$ be the iteration in which S is Covered

$\therefore u_k = 0$

- The sets $S_1, S_2 \cdots S_{k-1}$ ~~S_k covers~~ covers all elements in S and some elements are left uncovered in $S_1, S_2 \cdots S_{k-1}$.

$u_{i-1} = |S - (S_1 \cup S_2 \ldots \cup S_{i-1})|$

$u_i = |S - (S_1 \cup S_2 \ldots \cup S_i)|$

- $\boxed{u_{i-1} \geq u_i}$ and $u_{i-1} - u_i$ are the elements in the set S covered by the set S_i

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cdots S_{i-1})|}$$

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots S_{i-1})|} \times (u_{i-1} - u_i) \quad \text{--- (3)}$$

S_i ← Set selected by greedy in the ith Iteration

$$|S_i - (S_1 \cup S_2 \cup \dots \cup S_{i-1})| \geq |S - (S_1 \cup S_2 \dots \cup S_{i-1})|$$

$$\doteq \boxed{U_{i-1}}$$

Substitute in 8

$$\sum_{x \in S} C_x \leq \sum_{i=1}^{K} \frac{1}{U_{i-1}} \times (U_{i-1} - U_i)$$

$\downarrow 20$ $\to 15$

$\to 20$

$$= \sum_{i=1}^{K} \sum_{j=U_i+1}^{U_i-1} \frac{1}{U_{i-1}}$$

$\to 16$

$$\leq \sum_{i=1}^{K} \sum_{j=U_i+1}^{U_{i-1}} \left(\frac{1}{j}\right)$$

Think of numbers

$$= \sum_{i=1}^{K} \left(\sum_{j=1}^{U_{i-1}} \frac{1}{j} - \sum_{j=1}^{U_i} \frac{1}{j} \right)$$

$$= \sum_{i=1}^{K} \left(H(U_{i-1}) - H(U_i) \right)$$

$$= H(U_0) - H(U_K)$$

$$= H(|S|) - H(0)$$

$H(0) = 0$

$$= H(|S|)$$

From 4

$$|C| \leq \sum_{S \in C^*} H\left(\max\{|S| : S \in F\}\right)$$

we have $|X| = n$ $\therefore$ $\max\{|S| : S \in F\} \leq n$

$$|C| \leq \sum_{S \in C^*} H(n)$$

$$|C| \leq |C^*| \left(\ln(n) + O(1)\right)$$ Proved.