6.046
5/13/03
BT

L23.1

Lecture 23: Approximation Algorithms

General Flow of Algorithm Presentation

1. Problem
2. Algorithm
3. Analysis
   - Running Time - polynomial or better
   - Correctness

Violated this last week (NP-hard problems)

Will violate this today

How to solve problems that are NP-hard

1. For sufficiently small input → apply exponential algorithm
   - gives correct solution
   - may require harnessing large computing resources
2. Accept potentially incorrect solution in return for polynomial running time
   ⟹ Approximation algorithm

Focus here today; often this is sufficient (and problem must be solved quickly — military). Most effective if have guarantee on soln quality

example: primality testing for RSA crytosystem
"Sort of" → very small probability that non-prime would be ~~de~~ called prime (decision)

☆ → valid solution that is not optimal — Traveling salesperson problem with tour not ~~shortest~~ possible. (optimality)

Formalism

Given optimization problem on input of size n

Let $C^* =$ "cost" of optimal soln (not running time, but tour length)

Let $C =$ cost of approx alg soln

Then

Ratio Bound $\rho(n)$

$$\max\left(\underbrace{\frac{C}{C^*}}_{\text{minimization}}, \underbrace{\frac{C^*}{C}}_{\text{maximization}}\right) \le \rho(n) \qquad \text{for any } n$$

Relative Error Bound $\varepsilon(n)$

$$\frac{|C - C^*|}{C^*} \le \varepsilon(n) \qquad \text{for any } n$$

Examples

1. TSP: 2-approximation algorithm (can be improved to 1.5)
2. Set cover: $O(\lg m)$-approximation algorithm
3. Vertex cover: 2-approximation algorithm

TSP (optimization) NP-hard

Input: Undirected, complete graph with edge lengths $c(u,v)$

Output: A cycle tour of minimum length that visits each vertex exactly once.

(Note: NP-hard)

APPROX-TSP-TOUR

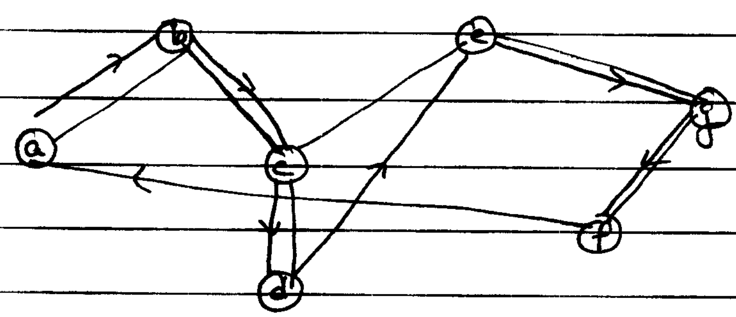
a) minimum
b) exactly once

1. Build minimum spanning tree (MST) T for G
2. Let L be the list of vertices visited in preorder tree walk of T
3. RETURN a tour H that visits vertices in order L.

parent visited before children

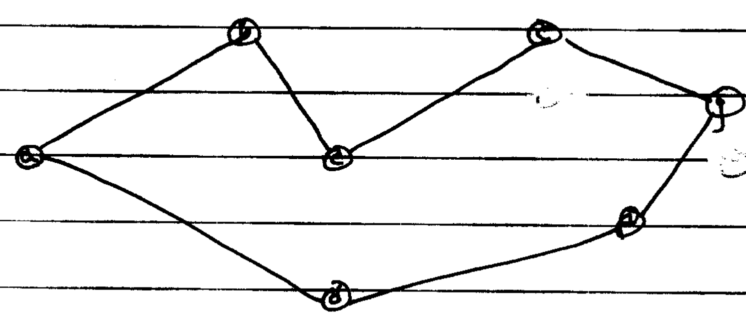
Example

① Problem
② MST
③ Tour



total distance ≈ 24.00

OPTIMAL (visit d later)



total distance ≈ 20.44

<u>Theorem</u>: Approx-TSP-Tour (with triangle inequality) has ratio bound of 2.

<u>Proof:</u>

- Let H^* be an optimal tour for G
- Since T is a MST for G, $c(T) \leq c(H^*)$
- Let W be a "full walk" of T (each ~~MST~~ vertex visited twice)
  $\Rightarrow \quad c(W) = 2c(T)$

- H tour obtained by shortcutting T
- Δ-inequality $\Rightarrow$ $c(H) \leq c(W) = 2c(T) \leq 2c(H^*)$

Note ① even with simple MST-PRIM, running time is $\Theta(V^2)$
$\Rightarrow$ polynomial

② A ratio bound was proven without access to the optimal solution

Set Cover (another NP-complete problem)

Input: Collection of sets $S_1, \ldots, S_n \subset U$, $\bigcup_i S_i = U$, $|U| = m$

Goal: Find $I \subset \{1 \ldots n\}$ of smallest size with $\bigcup_{i \in I} S_i = U$

Why?

U = set of tasks

S_i = set of tasks that person i can perform

Finds "skeleton crew"

Greedy soln

REPEAT until all elements covered

- Choose a new set S_i containing max # uncovered elmts
- Add i to I
- Mark all elements from S_i as covered

Ex:

$$\underset{S_1}{\{1,2\}} \quad \underset{S_2}{\{3,4\}} \quad \underset{S_3}{\{5,6\}} \quad \underset{S_4}{\{1,3,5\}}$$

optimal: $I = \{1,2,3\}$

greedy: $I = \{1,2,3,4\}$

However, this is $(\ln m)$-approximation algorithm

<u>Proof:</u> Let k be size of minimal cover

Fact: At any point, there is at least one set S_i that covers $\geq 1/k$ fraction of uncovered elements

Because: ~~If each set covered $< 1/k$ fraction of uncovered elements,~~ Otherwise there would be no way to cover U in only k sets.

<u>ln m-approximation</u>

- Let u_i be # uncovered elements after i^{th} iter.
- $u_{i+1} \leq (1 - \frac{1}{k}) u_i$
- Initially $u_0 = m$, so $u_i \leq (1 - \frac{1}{k})^i m$
- At $i = k \ln m \Rightarrow u_i \leq (1-\frac{1}{k})^{k \ln m} m \leq e^{-\ln m} m = 1$

  so at $\underline{k \ln m + 1}$ steps algorithm covers all elements and stops
- Thus, algorithm is $(\ln m + \frac{1}{k})$-approximate

  { we skip this for simplicity

<u>Again</u>

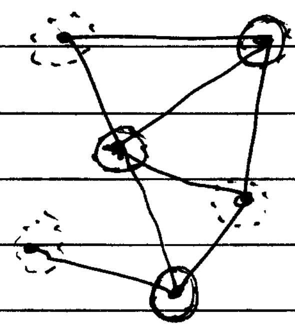
- Prove bound without access to optimal soln
- Note that greedy algorithms can be exact or approx.

Vertex Cover Problem (NP-complete)

Input: Undirected graph $G(V,E)$

Output: Minimum size set $C \subseteq V$ such that each edge in E has at least one endpoint in C.

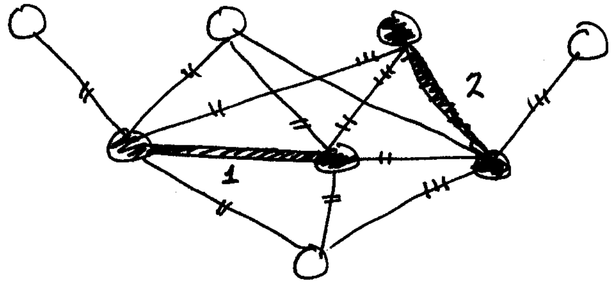
(Special case of set cover, in which every element [endpoint edge] is contained in only 2 sets [endpoints])



APPROX-VERTEX-COVER (G)

```
C ← ∅
E' ← E[G]
while E' ≠ ∅
    do let (u,v) be an arbitrary edge of E'
        C ← C ∪ {u,v}
        remove from E' every edge incident on either u or v
return C
```

Example



→ Done: Cover = 4

(optimal = 3 -- remove top ~~node~~ vertex)

Analysis

Correct

- only remove covered edges from E'
- Iterate until E' is empty

Running Time: $O(V+E)$ ← Each edge is added to E' on initialization and removed once. Each vertex is added to C at most once.

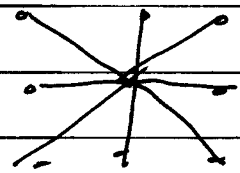
Theorem: APPROX-VERTEX-COVER has ratio bound of 2

Proof:

- A = {edges picked at do}
- no 2 edges in A share an endpoint
- $|C| = 2|A|$
- Optimal cover must include at least one endpoint for each edge in A.
- $|A| \le |C^*| \Rightarrow |C| \le 2|C^*|$

Why aren't the following improvements?

① Only pick one endpoint:



→ could have $n-1$ cover, rather than optimal, of 1 (~~greedy~~ approx gives 2)

② Being greedy

- Pick vertex with largest # of incident edges at each iteration
- Turns out worst case is $\Theta(\lg n)$ approximate

③ Could be greedy in edge selection (always pick edge whose endpoints have highest degree)

- does not improve worst case bound
- complicates analysis

BUT ④ Post processing can help ~~practically~~

- remove extra vertices at end
- can't weaken approximation bound
- can be quite useful in practical situations