24 April 2019 — Approximation Algorithms (Chapter 11)

Announcement: Homework 10 will be released Sunday, deadline the following Sunday.

Theme: What do you do if you have to solve an NP-hard problem?

Approx algs: always run in poly time, and their answer, while not always optimal, is guaranteed to be within some distance of optimal.

This definition makes sense for optimization problems: those that require maximizing or minimizing some quantity. (Aside: also makes sense for counting problems.)

Def: Algorithm ALG is an α-approximation for problem Π if it is the case that $\forall$ input instance x of Π with optimal solution $OPT(x)$,

$$val(OPT(x)) \leq val(ALG(x)) \leq \alpha \cdot val(OPT(x)) \quad \text{[minimization problem]}$$

$$val(ALG(x)) \leq val(OPT(x)) \leq \alpha \cdot val(ALG(x)) \quad \text{[maximization problem]}$$

automatic by definition of $OPT(x)$

$\frac{1}{2} val(OPT(x)) \leq$ ← This inequality is the right-hand inequality, rescaled.

"approximation guarantee" $\alpha \geq 1$.

Closer to $1 \Rightarrow$ closer to optimal.

EXAMPLE: Metric Traveling Salesman Problem.

Recall, in this problem we are given a table of distances $d(i,j)$ between cities i, j for $1 \leq i, j \leq n$.

Assume d is

- symmetric: $d(i,j) = d(j,i)$
- metric (Δ ineq): $d(i,j) + d(j,k) \geq d(i,k) \quad \forall i,j,k.$

$\Rightarrow$ any path $i_1, i_2, \ldots, i_p$

$$d(i_1,i_2) + d(i_2,i_3) + \ldots + d(i_{p-1},i_p) \geq d(i_1,i_p)$$

Objective: visit each of cities $1, \ldots, n$ and return to starting point, minimizing total distance traveled.

Recall: NP-hard because UNDIR HAM CYCLE $\leq_p$ METRIC TSP

If G is an instance of undir Ham cycle with vertex set $\{v_1, \ldots, v_n\}$ then create metric TSP with

$$d(i,j) = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 2 & \text{if not} \end{cases}$$

A traveling salesman tour of total length n exists if and only if G has a Ham. cycle.

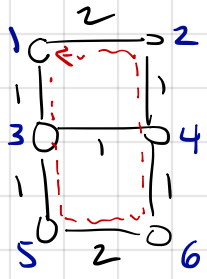
How to approximate the optimum?

Beautiful idea:

[1] if we take any n-cycle through the cities and remove one of its edges, we get a spanning tree. (which happens to be a path.)

[2] we know how to compute the min spanning tree efficiently.

[3] maybe the min spanning tree is close to the minimum traveling salesman tour?



(All other distances are shortest path distances.)

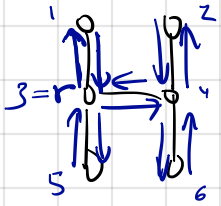
Cost of cheapest traveling salesman tour?

(A) 6

(B) 8

(C) 10

If you solve MST on this graph:



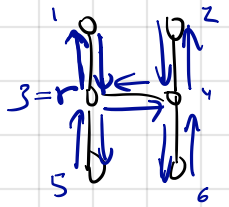
(cont 5)

What order to visit the cities, using this tree as a hint?

Answer: DFS!

DFS order starting from root node 3 is 3, 1, 3, 5, 3, 4, 6, 4, 2, 4, 3

If you solve MST on this graph:



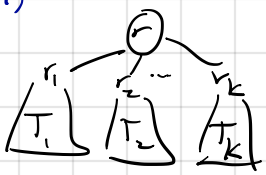
(cost 5)

What order to visit the cities, using this tree as a hint?

Answer: DFS!

DFS order starting from root node 3 is 3,1,3,5,3,4,6,4,2,4,3

DFS order of tree T: if r = root of T, and $r_1, r_2, \ldots, r_k$ are children of r, and $T_1, T_2, \ldots, T_k$ are the subtrees rooted at these children



DFS-order(T) is defined recursively by

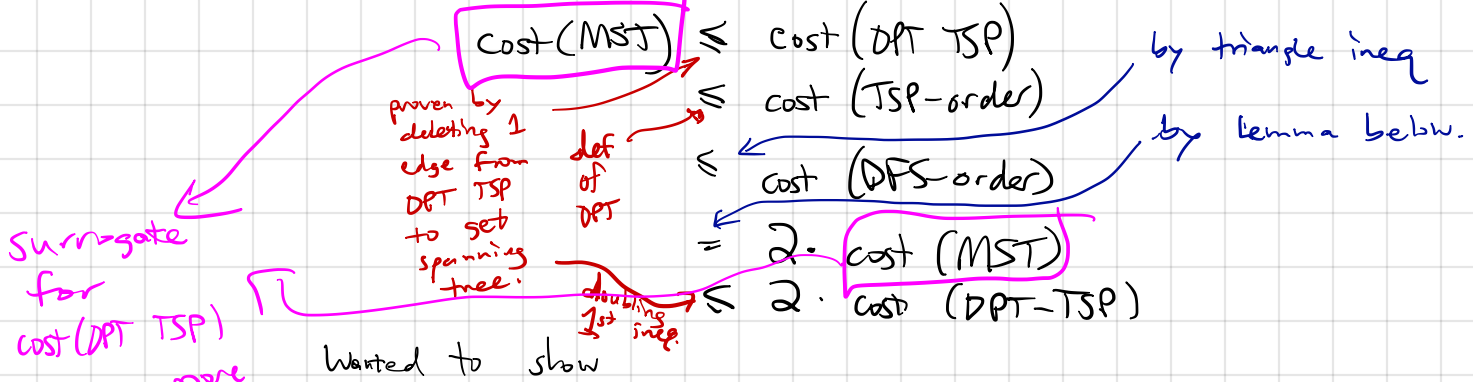
- $[r]$ if $k = 0$ (T has one node)
- $[r,$ DFS-order(T_1), $r,$ DFS-order(T_2), $r, \ldots,$ DFS-order(T_k), $r]$

TSP-order(T) is defined by computing DFS-order(T) and removing all copies of each node except the first one in the sequence. Finally, repeat starting pt.

E.g. 3,1,3,5,3,4,6,4,2,4,3 becomes 3,1,5,4,6,2,3

Claim: If you solve any Metric TSP problem by computing min spanning tree, T, and returning TSP-order(T), this is a 2-approximation.

Proof. Sketch is as follows:



Wanted to show

$$\text{cost}(\text{OPT-TSP}) \leq \text{cost}(\text{TSP order}) \leq 2 \cdot \text{cost}(\text{OPT-TSP})$$

cost (TSP-order)
$\leq$ cost (DFS-order) — by triangle ineq
$=$ 2 · cost (MST) — by lemma below.

For any sequence $i_1, i_2, \ldots, i_p$ define

$$\text{cost}(i_1, i_2, \ldots, i_p) = \sum_{s=1}^{p-1} d(i_s, i_{s+1}).$$

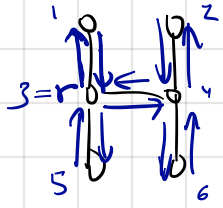
DFS-order: 3, 1, 3, 5, 3, 4, 6, 4, 2, 4, 3

TSP-order: 3, 1, 5, 4, 6, 2, 3

cost(top seq) $\geq$ cost(bottom seq) by Δ ineq.

Lemma: Each edge of the MST T appears exactly twice when we look at the list of pairs of nodes that occur consecutively in DFS-order(T).

Proof. (by picture)



(by induction) True when T has one node and no edges,

DFS-order(T) $=$ $[r]$

Else DFS-order$(T) = [r,$ DFS-order(T_1), r, DFS(order(T_2), $r, \ldots,$ DFS-order(T_k), $r]$

(r, r_2) occurs twice

edge (r, r_1) occurs twice

Every edge of r occurs exactly twice.

Ind hyp: every edge of $T_1 \cup T_2 \cup \ldots \cup T_k$ occurs exactly twice.