TSP = traveling salesman problem

Given $G=(V,E)$ which is a complete undirected graph with a weight w_{ij} for edge (i,j)

Find the tour of minimum weight

tour = <u>cycle</u> that visits every vertex exactly once.

for a tour T, $w(T) = \sum_{e \in T} w(e)$

assume $w(e) \geq 0$ for all $e \in E$.

Search version:

input: G with weights $w()$ & bound k

output: tour T of weight $\leq k$ if one exists
NO otherwise.

TSP-search version is NP-complete.

Moreover, NP-hard to approximate within any constant factor:
for all $c > 0$, no poly-time algorithm whose output T satisfies:

$$w(T) \leq c\,w(T^*)$$

where T^* is a tour of min weight,

unless NP=P.

Proof:

Consider Hamiltonian cycle problem:

input: $G=(V,E)$ (not necessarily complete)

output: tour T if one exists in G

NO otherwise

(G is unweighted, so just trying to find any tour)

Hamiltonian cycle is NP-complete.

Suppose we had a poly-time algorithm A to approximate TSP within a constant factor c.

Let's use that to solve Hamiltonian cycle.

Take input $G=(V,E)$ for Hamiltonian cycle.

Create complete graph G' on V where edge (i,j) has weight:

- if $(i,j) \in E$ then $w(i,j) = 1$
- if $(i,j) \notin E$ then $w(i,j) = c(n+1)$ where $n = |V|$.

Run A on this new weighted graph G'.

If there is a tour in the original graph G then in this new graph G' there is a tour of weight $= n$.

Hence A will output a tour of weight $\leq cn$.

If original G doesn't have a tour, then in G' we need to use at least one non-edge of G.

So the tour in G' will have weight $\geq c(n+1)$.

Hence, if output of A has weight $\leq cn$ then G has a Hamiltonian cycle.

& if output of A has weight $\geq c(n+1)$ then G does not have a Hamiltonian cycle.

$\blacksquare$

Note: same proof shows its NP-hard within any poly(n) factor.

Look at restricted class of inputs:

Assume triangle inequality is satisfied:

for all vertices i, j, k:

$$w(i,j) \leq w(i,k) + w(k,j).$$

This is Metric TSP.

2-approximation algorithm for Metric TSP:

1) Compute a MST T of input G.

2) Make a walk W on T so that visit every edge of T exactly twice & every vertex at least once.

Example: just follow DFS traversal.

3) Shortcut W: for any vertices visited more than once, cut off visits after 1st.

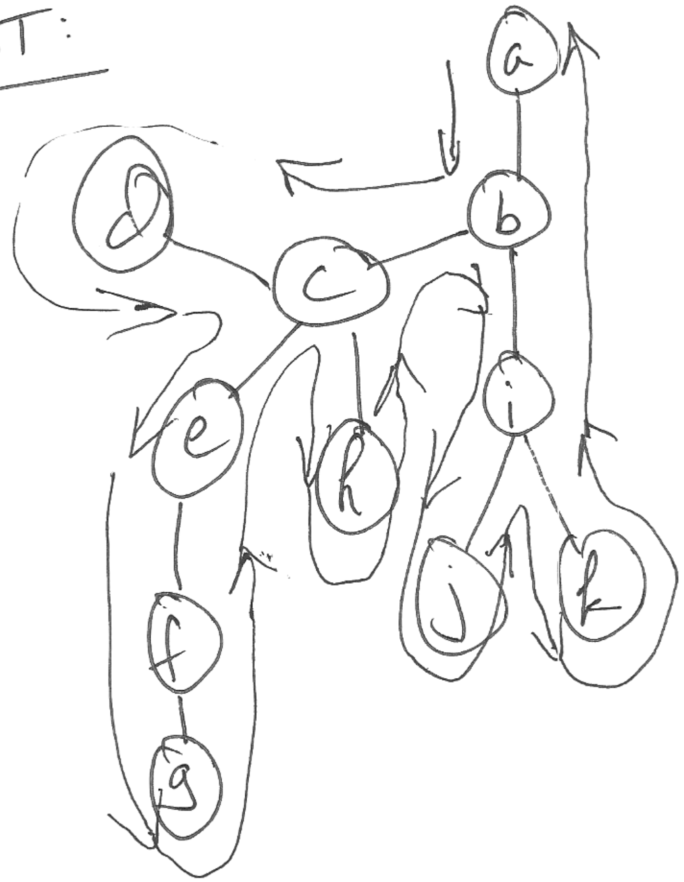
Example: if vertex 3 is visited & later on we have $\ldots \to 5 \to 3 \to 7 \to \ldots$

then replace by: $\ldots \to 5 \longrightarrow 7 \to \ldots$

Let T' be the final tour.

Example:

MST:



Walk W:

$a \to b \to c \to d \to c \to e \to f \to g \to f \to e \to c \to h \to c \to b \to i \to j \to i \to k \to i \to b \to a$

Shortcut version T:

$a \to b \to c \to d \to e \to f \to g \to h \to i \to j \to k \to a$

By triangle inequality,

$$w(T') \leq w(W)$$

Since W traverses every edge twice,

$$w(W) \leq 2w(T)$$

Take optimal tour T^*.

Delete any one edge from T^* & we have a tree S. T is a MST so:

$$w(T) \leq w(S) \leq w(T^*)$$

Therefore:

$$w(T') \leq 2w(T) \leq 2w(T^*)$$

So our output T' is at most 2 times the weight of the optimal tour T^*.

Now: 1.5-approximation algorithm

This is Christofide's algorithm [76]

First some basic graph theory facts.

For undirected $G=(V,E)$,

<u>Lemma 1</u>: if every vertex has degree ≥ 2 then G contains a cycle.

<u>Proof</u>: Consider the <u>longest</u> path $v_1, \ldots, v_\ell$ in G.

$\deg(v_1) \geq 2$ so v_1 has another neighbor, call it x. other than v_2

x must appear on the path, so for some $i > 2$, $v_i = x$.

If x doesn't appear on the path we have a longer path: $x, v_1, \ldots, v_\ell$.

The cycle is $v_1, v_2, \ldots, v_i, v_1$

$\blacksquare$

Lemma 2:

if every vertex has even degree then every connected component of G has an Eulerian trail.

Proof:

Induct on # of edges.

Consider a component K of G.

If $|K|=1$ then this vertex by itself is an Eulerian trail of it.

So assume $|K| \geq 2$, and then every vertex in it has degree ≥ 2. Hence there is a cycle C in K.

Delete C from G, call the new graph G'.

By induction G' has an Eulerian trail T in every component.

For any visit to a vertex in C by T, can add C & make an Eulerian trail T' for G.

∎

Remaining fact:

Perfect matching = Subset M of edges containing each vertex exactly once.

Can find a minimum weight perfect matching in polynomial-time.

Edmonds ['61] algorithm for general graphs — next class.

Christofide's algorithm:

1) Find MST T of G.
2) Let S be those vertices of odd degree in T.
   Claim: $|S|$ is even
3) Find a min-cost perfect matching M in S.
4) Add M to T.
   (Now every vertex has even degree)
5) Find Eulerian walk W on $T \cup M$.
6) Shortcut W to get a tour T'.

First, proof of claim that $|S|$ is even:

$$\sum_{v} \deg_T(v) = 2|T| = \text{even } \#$$

$\uparrow$ degree of v in T

$$\sum_{v} \deg_T(v) = \sum_{v \in S} \deg_T(v) + \sum_{v \notin S} \underbrace{\deg_T(v)}_{\text{even } \#\text{'s}}$$

So $\sum_{v \notin S} \deg_T(v) = \text{even } \#$

Hence, $\sum_{v \in S} \deg_T(v) = \text{even } \#$

Since $\deg_T(v)$ is odd, there must be even # of terms (if odd #, get odd sum)

So $|S| = \text{even}$. $\boxtimes$

Let T^* be an optimal tour.

Drop an edge from T^* & we get a tree so:

$$w(T) \leq w(T^*)$$

Claim: $w(M) \leq \frac{1}{2} w(T^*)$

Therefore,

$$w(T') \leq w(W) = w(T) + w(M) \leq \frac{3}{2} w(T^*)$$

which shows we have a $\frac{3}{2}$-approx. algorithm.

Proof of claim:

Recall S = vertices with odd degree in T.

Take an optimal tour on S, call it R.

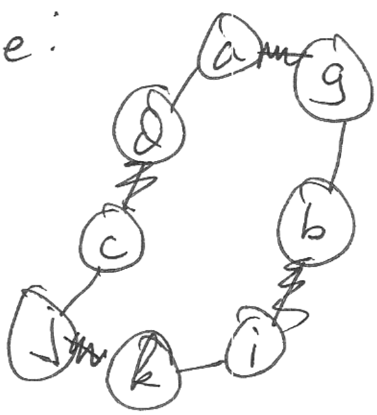
From T^*, shortcut vertices not in S to get a tour of S,

hence $w(T^*) \geq w(R)$.

$|S|$ = even, so R is an even-length cycle.

Hence R gives 2 perfect matchings on S

Example:



Let M_1, M_2 be these 2 perfect matchings.

$$w(R) = w(M_1) + w(M_2)$$

Thus:

$$\min\{w(M_1), w(M_2)\} \leq \frac{1}{2}\left(w(M_1) + w(M_2)\right) = \frac{1}{2} w(R).$$

Say $w(M_1) \leq w(M_2)$.

Then, $w(M_1) \leq \frac{1}{2} w(R)$.

M is a min weight perfect matching on S,

So $w(M) \leq w(M_1)$

Therefore, $w(M) \leq w(M_1) \leq \frac{1}{2} w(R) \leq \frac{1}{2} w(T^*)$. $\boxtimes$

This is still the best for metric TSP.

Recently, better results for graphical TSP:

$w(i,j)$ = length of shortest path between i & j.