`Lecture 1

Friday, February 24, 2017 10:38 AM

- For a minimization problem, an α-appx algo $\mathcal{A}$ takes any instance $\mathcal{I}$ of the problem and returns a solution S s.t

$$\mathrm{cost}(S) \leq \alpha \cdot \mathrm{opt}(\mathcal{I})$$

opt-cost ie, cost of opt soln for instance $\mathcal{I}$

<u>Note</u>: $\alpha \geq 1$, and closer it's to 1, the better is quality of the algorithm

- For a maximization problem, we have a similar defn except

$$\mathrm{cost}(S) \geq \frac{\mathrm{opt}(\mathcal{I})}{\alpha}$$

Again $\alpha \geq 1$.

Sometimes one says a ρ-appx with $\rho < 1$ for max. problems — in that case one means

$$\mathrm{cost}(S) \geq \rho \cdot \mathrm{opt}(\mathcal{I})$$

<u>Examples</u>

① <u>Travelling Problem (TSP)</u>

<u>Input</u>: n points on a metric space (X, d)

$(\forall u, v, w \in X,\ d(u,w) \leq d(u,v) + d(v,w))$

<u>Output</u>: A tour / ordering of vertices in X

$(\sigma_1, \sigma_2, \ldots, \sigma_n) \leftarrow$ permutation.

<u>Objective</u>: Minimize

$$\sum_{i=1}^{n-1} d(\sigma_i, \sigma_{i+1}) + d(\sigma_n, \sigma_1)$$

$$\sum_{i=1} d(v_i, v_{i+1}) + d(v_n, v_1)$$

$\boxed{\text{NP-hard}}$

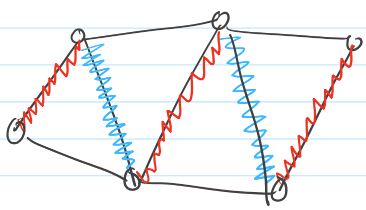
② Matching

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

Output : $M \subseteq E$ st $\deg_M(v) \leq 1$

degree in $G(V, M)$

Example :



G

Both the red edges and the blue edges are valid matchings.

Objective : Maximize $|M|$. $\boxed{\text{in P}}$

Algorithm for Matching Problem

ⓐ Initially $M = \emptyset$, empty set.

ⓑ Considere edges of G in any order.

ⓒ While considering edge (u, v)

if $\deg_M(u) = \deg_M(v) = 0$,

then $M = M + (u, v)$

Simple algorithm, fast, how good is it?

Claim: The above algorithm is a 2-appx algo.

Proof:- Fix a graph G and let M^* be the maximum matching. Let M be the matching returned by the above algorithm.

We wish to show $|M| \geq \frac{1}{2} \cdot |M^*|$

In order to do so, we define a many-to-1 map $\phi : M^* \longrightarrow M$ s.t.

$\forall\, e \in M$, there are at most two edges $e_1, e_2 \in M^*$ with

$$\phi(e_1) = e \ \& \ \phi(e_2) = e$$

This will prove $|M| \geq \frac{1}{2} \cdot |M^*|$

For all $(u,v) \in M^* \cap M$, $\phi(u,v) = (u,v)$

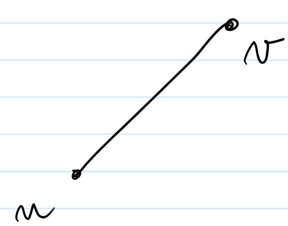
For all $(u,v) \in M^* \setminus M$,

since we haven't picked it M

$\Rightarrow \exists\, (v,w)$ or (u,x) or both in M.

Arbitarily map $\phi(u,v)$ to one of them.

Pick an edge $(u,v) \in M$,



If $e \in M^*$ has $\phi(e) = (u,v)$

then $e \sim (u,v)$, ie, e and (u,v) must share a common endpoint then $e \sim (u,w)$, ie, e and (u,w) must share a common end point.

No two $e, f \in M^*$ can share the same ept. $\because$ M^* is a matching.

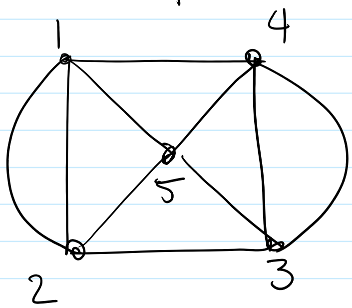
Since (u,v) has only two epts, atmost 2 edges in M^* map to (u,v)

$\blacksquare$

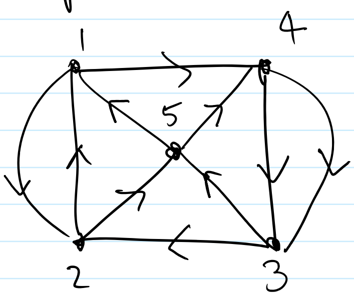
Algorithms for TSP

Preliminaries

Eulerian Tour: A walk in G is an Eulerian walk if every edge is visited exactly once. It's called an Eulerian tour if start and end points are same



has an Eul. tour



$5 \to 1 \to 2 \to 5 \to 4 \to 3 \to 2 \to 1 \to 4 \to 3 \to 5$

Thm: G has an Eulerian tour iff G is connected & $\deg(v)$ is even for all v.

* Such graphs are called Eulerian.

∴ Checking if there is a tour visiting every edge of a G (exactly once) is easy.

Eulerian tours vs metric TSP

Given a metric (X, d), let G be the complete graph with $wt(u,v) = d(u,v)$.
Let F be any Eulerian subgraph of G.

Claim :- There is a tour of cost $\leq wt(F)$

$$wt(F) = \sum_{e \in F} wt(e)$$

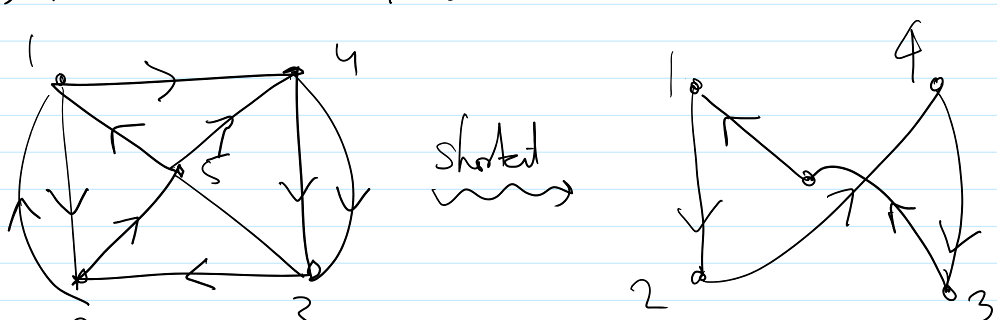
Pf :- Let W be the Eulerian tour of F.

$\sigma \equiv$ Shortcut(W)

whenever a vertex is repeated we just skip it.

eg: in the example above W is

$5 \to 1 \to 2 \to 5 \to 4 \to 3 \to 2 \to 1 \to 4 \to 3 \to 5$



shortcut$(W) \equiv$ $5 \to 1 \to 2 \to 4 \to 3 \to 5$

$cost(\sigma) \leq cost(W)$ ∵ of Δ-ineq.

eg: $\to 2 \to 5 \to 4 \to$ is shortcut to

$\to 2 \to 4 \to$

but $d(2,4) \leq d(2,5) + d(5,4)$

This is where metric prop is crucially used

$\therefore$ Finding "small" tours in (X,d) boils down to finding "small cost" Eulerian subgraphs of G.

$\boxed{\text{Algo 1}}$

1. T be the MST of G
2. $2T$ be the graph on X obtained by taking two parallel copies of each edge of T.
3. W be the Eulerian tour of $2T$
4. return Shortcut(W)

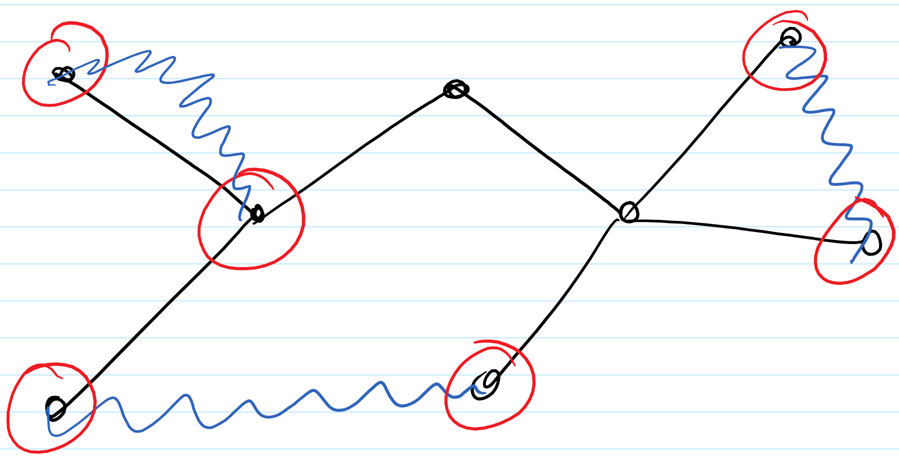
Thm :- Algo 1 is a 2-appx algo for TSP.

Pf :

* $2T$ is Eulerian by defn.
* $\text{cost}(2T) \leq 2\,\text{cost}(T) \leq 2\,OPT$

since the opt tour contains a sp. tree

In the previous algorithm, we ensured that every degree is even by taking two copies of every edge in T. But we can do something better.



Suppose T is the mst. The "problematic" vertices are the odd-degree vertices.

<u>Obs</u>: # of odd-degree nodes in <u>any</u> tree is even.

<u>Idea</u>: "Pair these nodes up."

How? In the cheapest possible way.

By adding a minimum wt perfect matching.

Algorithm 2

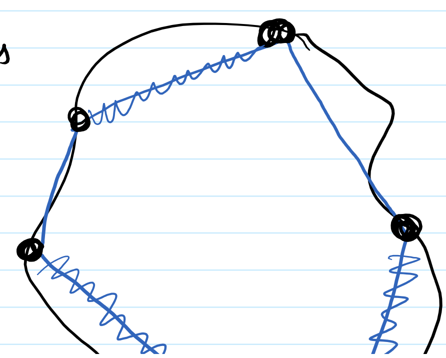
1. Find T : mst of G
2. O be the set of odd-degree vertices in T
3. M be the min wt perfect matching connecting O in G
4. $T \cup M$ is an Eulerian graph.
   W : Eulerian tour in $T \cup M$
   Return : Shortcut (W)

<u>Thm</u>: The above algo is $\frac{3}{2}$-approximate.

<u>Pf</u>: Suffices to show $wt(M) \leq \frac{1}{2} \cdot OPT$

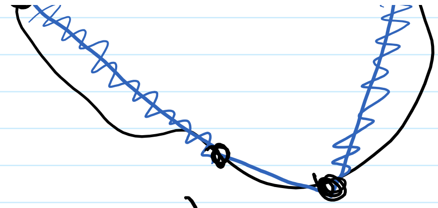
since $w(T) \leq OPT$.

Again look at the opt. tour and consider the O-vertices in this tour.



Note :

Note :



OPT $\geq$ "total blue length"

$\geq 2 \cdot$ (min wt matching)

since the blue lines partition into two matchings.

$\blacksquare$