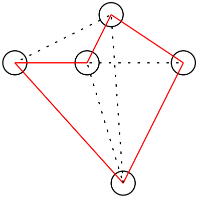
Traveling Salesman



Given $G = (V, V \times V)$, find simple cycle $C = (v_1, v_2, \ldots, v_n, v_1)$ such that $n = |V|$ and $\sum_{(u,v) \in C} d(u, v)$ is minimized.

Repeat

Approximation algorithms	Online algorithms
NP-hard problems	Incomplete information
look for good solution	look for good solution
Approximation ratio	Competitive ratio
Polynomial time	possibly exponential time
TSP, Knapsack,	online TSP, Knapsack,...
Load balancing,...	
	Paging, Ski rental, ...

Traveling Salesman

Given $G = (V, V \times V)$, find simple cycle $C = (v_1, v_2, \ldots, v_n, v_1)$ such that $n = |V|$ and $\sum_{(u,v)\in C} d(u,v)$ is minimized.

Applications:

- drilling printed circuit boards
- the analysis of the structure of crystals (Bland and Shallcross 87)
- the overhauling of gas turbine engines (Panteet al. 87)
- material handling in a warehouse (Ratliff & Rosenthal 81)
- cutting stock problems (Garfinkel 77)
- clustering of data arrays (Lenstra and Rinooy Kan 75)
- sequencing of jobs on a single machine (Gilmore and Gomory 64)
- assignment of routes for planes of a specified fleet (Boland et al. 94)

Theorem 1

It is **NP***-hard to approximate the general TSP within any factor* α.

Proof.

Reduction from Hamilton Cycle ... $\square$

Hamilton Cycle Problem:
Given a graph decide whether it contains a simple cycle visiting all nodes

Proof.

We want to find a Hamilton Cycle in $G = (V, E)$.
Consider $G' = (V, V \times V)$ and the weight function

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ \alpha n & \textit{else} \end{cases}$$

Suppose G has a Hamilton cycle.
Then there is a Hamilton cycle of weight n in G'
$\rightarrow$ an α-approx. algorithm delivers one with weight $\leq \alpha n$
If there is no Hamilton cycle in G, every Hamilton Cycle in G' has weight $\geq \alpha n + n - 1 > \alpha n$.

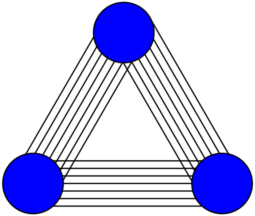
$\square$

Proof (continued)

Assume that there exists an α-approximation algorithm for TSP.
Decision algorithm: Run α-approx TSP on G'
Solution has weight $\leq \alpha n \rightarrow$ Hamilton path exists
Else there is no Hamilton cycle. [e.g. Vazirani Theorem 3.6] $\square$

Metric TSP

G is undirected and obeys the triangle inequality
$\forall u, v, w \in V : d(u,w) \leq d(u,v) + d(v,w)$



Metric completion Consider any connected undirected graph $G = (V, E)$ with weight function $c : E \to \mathbb{R}_+$. Define
$d(u,v) :=$ shortest path distance from u to v
Example: (undirected) street graphs $\to$ distance table

2-Approximation by MST

Lemma 2

The total weight of an MST $\leq$
The total weight of any TSP tour

Algorithm:

T:= MST(G) // weight(T) $\leq$opt
T':= T with every edge doubled // weight(T') $\leq$2opt
T'':= EulerTour(T') // weight(T'') $\leq$ 2opt
output removeDuplicates(T'') // shortcutting

Exercise: Implementation in time $\mathcal{O}(m + n \log n)$ where m is number of edges before metric completion

2-Approximation by MST

Lemma 2

The total weight of an MST $\leq$
The total weight of any TSP tour

Algorithm:

T:= MST(G) // $\text{weight}(T) \leq \text{opt}$
T':= T with every edge doubled // $\text{weight}(T') \leq 2\text{opt}$
T'':= EulerTour(T') // $\text{weight}(T'') \leq 2\text{opt}$
output removeDuplicates(T'') // shortcutting

Exercise: Implementation in time $\mathcal{O}(m + n \log n)$ where m is number of edges before metric completion

2-Approximation by MST

Lemma 2

The total weight of an MST $\leq$
The total weight of any TSP tour

Algorithm:

T:= MST(G) // weight(T) $\leq$opt
T':= T with every edge doubled // weight(T') $\leq$2opt
T'':= EulerTour(T') // weight(T'') $\leq$ 2opt
output removeDuplicates(T'') // shortcutting

Exercise: Implementation in time $\mathcal{O}(m + n \log n)$ where m is number of edges before metric completion

2-Approximation by MST

Lemma 2

The total weight of an MST $\leq$
The total weight of any TSP tour

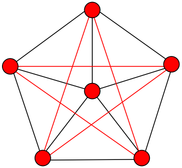
Algorithm:

T:= MST(G) // weight(T) $\leq$opt
T':= T with every edge doubled // weight(T') $\leq$2opt
T'':= EulerTour(T') // weight(T'') $\leq$ 2opt
output removeDuplicates(T'') // shortcutting

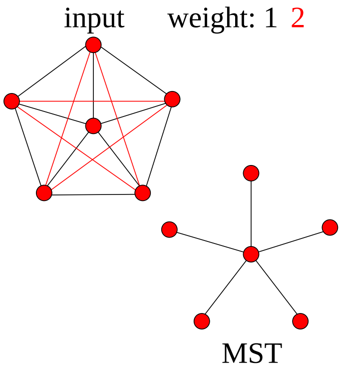
Exercise: Implementation in time $\mathcal{O}(m + n \log n)$ where m is number of edges before metric completion

Example

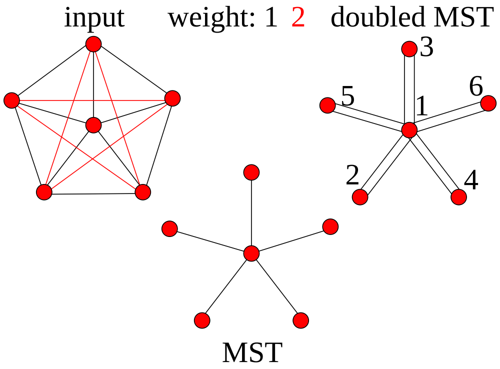
input weight: 1 2



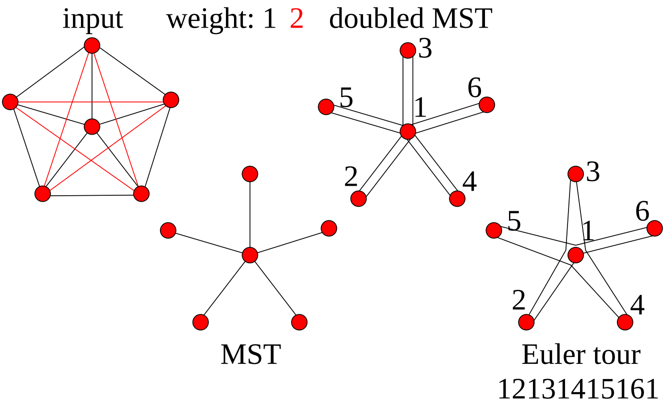
Example



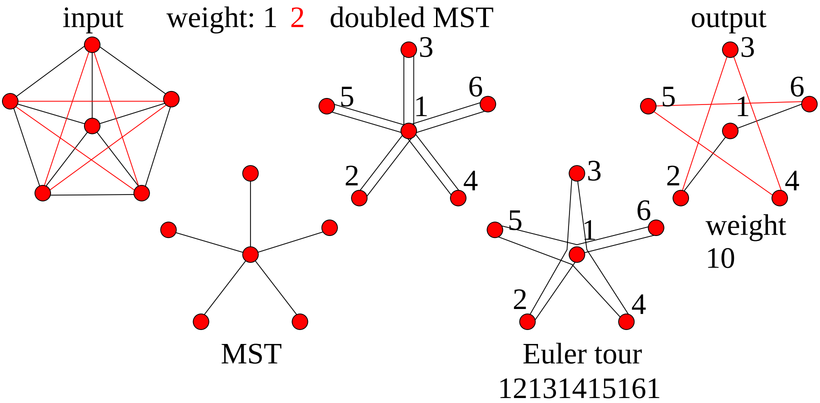
Example



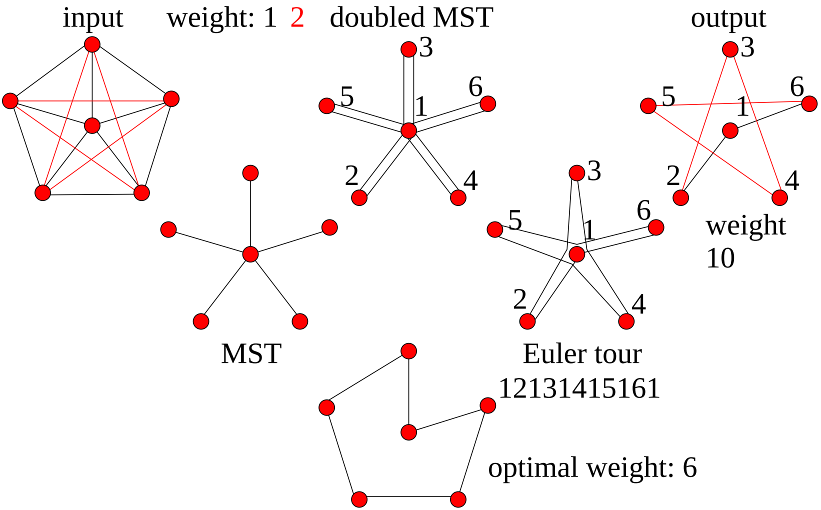
Example


input
weight: 1 2
doubled MST
3
5
6
1
2
4
MST
3
5
6
1
2
4
Euler tour
12131415161

Example


input
weight: 1 2
doubled MST
output
3
5
6
1
2
4
MST
Euler tour
12131415161
weight
10

Example



input
weight: 1 2
doubled MST
output
3
5
6
1
2
4
MST
Euler tour
12131415161
weight
10
optimal weight: 6

Proof of Lemma 2

Lemma 2

The total weight of an MST $\leq$
The total weight of any TSP tour

Proof.

Let T denote the optimal TSP tour
remove one edge from T makes T lighter
now T is a spanning tree
which is no lighter than the MST $\square$

General Technique: Relaxation
here: a TSP path is a special case of a spanning tree

More on TSP

- Practically better 2-approximations, e.g. lightest edge first
- Relatively simple yet unpractical 3/2-approximation
- PTAS for Euclidean TSP
- Guinea pig for just about any optimization heuristics
- Optimal solutions for practical instances. Rule of thumb:
  If it fits in memory it can be solved.
  [`http://www.tsp.gatech.edu/concorde.html`]
  lines of code is six digit number
- TSP-like applications are usually more complicated

Online TSP

- Metric space
- Algorithms move with speed at most 1
- Requests appear over time
- Future requests are unknown
- Minimize finishing time (makespan)

Online TSP

- What is the worst that can happen to an online algorithm?
  - Algorithm is at location X
  - Request occurs somewhere very far away from it, at Y
  - Optimal solution is to serve it immediately
  - No further requests arrive
  - Algorithm still needs to move to Y: high competitive ratio

Online TSP

- However...
- The optimal solution must have had enough time to travel to Y before the request arrives
- It started at the origin, like the online algorithm
- **Idea**: do not move “too far” from the origin
- Close enough = within a factor of time elapsed

Algorithm Return Home (RH)

(Lipmann, 2003)

- Whenever a new request arrives, return to O at full speed
- In O, calculate optimal tour for **all** requests that appeared so far
- Follow this tour at maximum speed such that distance to O is at most $(\sqrt{2}-1)t$ at time t, for all t

Theorem 3

Return Home has a competitive ratio of $\sqrt{2}+1$.

Proof.

Let t be the time at which the last request arrives.
Clearly $OPT \geq t$.
If RH does not slow down after time t, it needs time at most

$$t + (\sqrt{2}-1)t + OPT \leq (\sqrt{2}+1)OPT)$$

Else, let the last request for which RH slows down be a distance x from the origin. RH serves it at time $x/(\sqrt{2}-1) = (\sqrt{2}+1)x$. RH serves remainder of tour (T_x) at full speed. We have $OPT = x + T_x$ and RH is ready at time

$$(\sqrt{2}+1)x + T_x \leq (\sqrt{2}+1)OPT$$

□

Theorem 3

Return Home has a competitive ratio of $\sqrt{2}+1$.

Proof.

Let t be the time at which the last request arrives.
Clearly $OPT \geq t$.
If RH does not slow down after time t, it needs time at most

$$t + (\sqrt{2}-1)t + OPT \leq (\sqrt{2}+1)OPT)$$

Else, let the last request for which RH slows down be a distance x from the origin. RH serves it at time $x/(\sqrt{2}-1) = (\sqrt{2}+1)x$.
RH serves remainder of tour (T_x) at full speed. We have $OPT = x + T_x$ and RH is ready at time

$$(\sqrt{2}+1)x + T_x \leq (\sqrt{2}+1)OPT$$

□

Theorem 3

Return Home has a competitive ratio of $\sqrt{2}+1$.

Proof.

Let t be the time at which the last request arrives.
Clearly $OPT \geq t$.
If RH does not slow down after time t, it needs time at most

$$t + (\sqrt{2}-1)t + OPT \leq (\sqrt{2}+1)OPT)$$

Else, let the last request for which RH slows down be a distance x from the origin. RH serves it at time $x/(\sqrt{2}-1) = (\sqrt{2}+1)x$. RH serves remainder of tour (T_x) at full speed. We have $OPT = x + T_x$ and RH is ready at time

$$(\sqrt{2}+1)x + T_x \leq (\sqrt{2}+1)OPT$$

□

Remarks about RH

Karlsruhe Institute of Technology

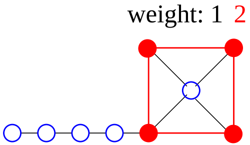
- Uses exponential time to calculate optimal tour
- Nevertheless, leaves O immediately after arriving there
- Theoretical result
- More reasonable: use some approximation algorithm in O
- Competitive ratio increases
- Time for serving requests should be much longer than time needed to calculate approximate tour

Steiner Trees

[C. F. Gauss 18??]
Given $G = (V, E)$, with positive edge weights $cost : E \to \mathbb{R}_+$
$V = R \cup F$, i.e., Required vertices and Steiner vertices
find a minimum cost tree $T \subseteq E$ that connects all required vertices

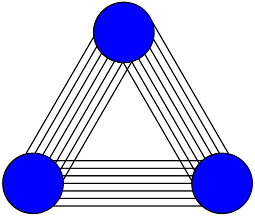
$$\forall u, v \in R : T \text{ contains a } u\text{-}v \text{ path}$$

THE network design problem



Metric Steiner Trees

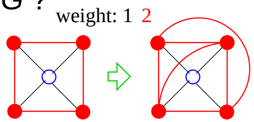
Find Steiner tree in complete graph with triangle inequality
$\forall u, v, w \in V : d(u,w) \leq d(u,v) + d(v,w)$



Easier?
No!

Approximation Factor Preserving Reduction

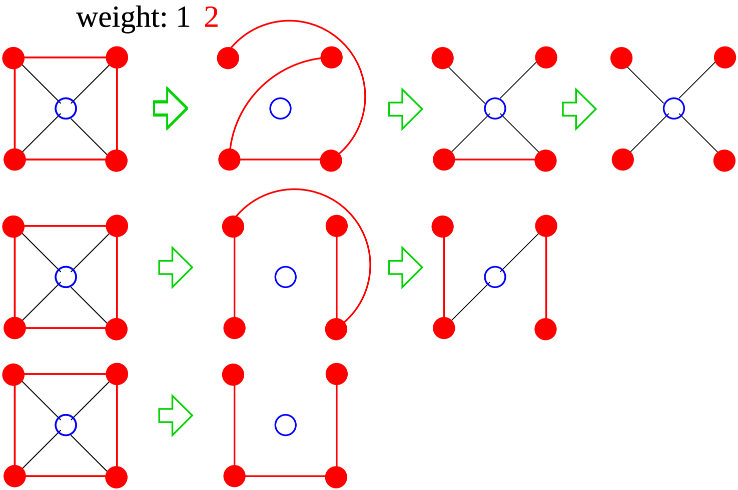
Steiner Tree of G? $\rightsquigarrow$ Metric Steiner Tree of G'?



Complete the graph G.
$\forall u, v \in V : cost(u, v) :=$
shortest path distance between u and v
we only add edges. Hence, $OPT(G') \leq OPT(G)$.
Now consider any Steiner tree $I' \subseteq G'$.
We construct a Steiner tree $I \subseteq G$ with $cost(I) \leq cost(I')$:
replace edges $\rightarrow$ paths
remove edges from cycles

Examples

From Metric Steiner Tree to Steiner Tree



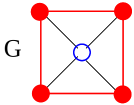
2-Approximation by MST

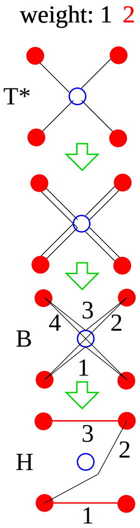
Given metric graph $G = (R \cup F, E)$
Find MST T of subgraph G_R induced by R

Theorem 4: $cost(T) \leq 2OPT$

Proof:
consider optimal solution T^* $(cost(T^*) = OPT)$
double edges of T^*
find Euler tour B $(cost(B) = 2OPT)$
use shortcuts to obtain Hamilton cycle H
$(cost(H) \leq cost(B) = 2OPT)$.
drop heaviest edge. Now H is a spanning tree of G_R

$$cost(\mathrm{MST}) \leq cost(H) \leq cost(B) = 2OPT$$





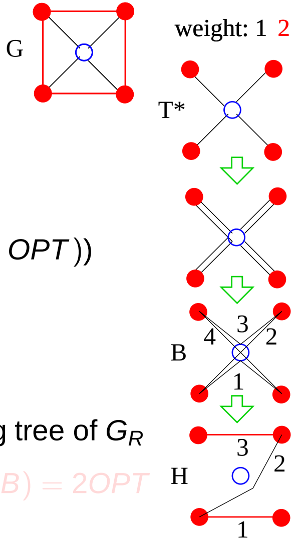
2-Approximation by MST

Given metric graph $G = (R \cup F, E)$
Find MST T of subgraph G_R induced by R

Theorem 4: $cost(T) \leq 2OPT$

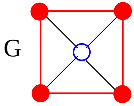
Proof:
consider optimal solution T^* ($cost(T^*) = OPT$))
double edges of T^*
find Euler tour B ($cost(B) = 2OPT$)
use shortcuts to obtain Hamilton cycle H
($cost(H) \leq cost(B) = 2OPT$).
drop heaviest edge. Now H is a spanning tree of G_R

$cost(\mathrm{MST}) \leq cost(H) \leq cost(B) = 2OPT$



2-Approximation by MST

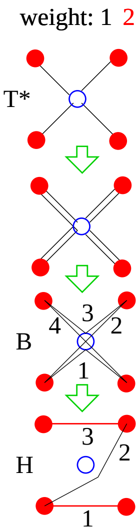
Given metric graph $G = (R \cup F, E)$
Find MST T of subgraph G_R induced by R



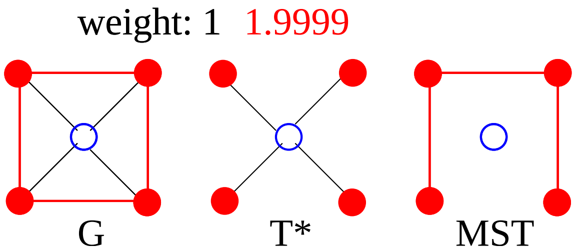
Theorem 4: $cost(T) \leq 2OPT$

Proof:
consider optimal solution T^* ($cost(T^*) = OPT$)
double edges of T^*
find Euler tour B ($cost(B) = 2OPT$)
use shortcuts to obtain Hamilton cycle H
($cost(H) \leq cost(B) = 2OPT$).
drop heaviest edge. Now H is a spanning tree of G_R

$$cost(\mathrm{MST}) \leq cost(H) \leq cost(B) = 2OPT$$



Tight Example



$$cost(T^*) = 4, cost(\mathrm{MST}) = 6.$$

More general:

$$|R| = n \to cost(T^*) = n, cost(\mathrm{MST}) = (2-\epsilon)(n-1)$$

More on Steiner trees

- Complicated Approximation down to 1.39 [Jaroslaw et al. 2010]
- Optimal solutions for large practical instances. [PhD Polzin,Daneshmand, 2003, Dortmund, Mannheim, MPII-SB]
- Many applications: multicasting in networks, VLSI design(?), phylogeny reconstruction

Directed Steiner Trees

Karlsruhe Institute of Technology

Theorem 4

It is hard to approximate the directed Steiner tree problem within a factor $\ln|R|$.

Proof by approximation preserving reduction from the set covering problem

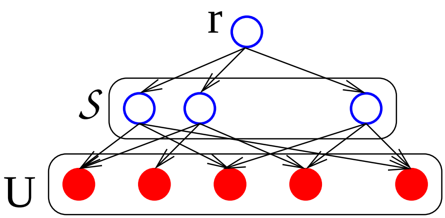
The Set Covering Problem

Given universe U, subsets $\mathcal{S} = \{S_1, \ldots, S_k\}$, cost function $c : \mathcal{S} \rightarrow \mathbb{N}$.
Find minimum cost $\mathcal{S}' \subseteq S$ such that $\bigcup_{S \in \mathcal{S}'} S = U$

Theorem 5

It is hard to approximate the set covering problem within a factor $\ln |U|$. [Feige 98]

Approximation Preserving Reduction: Directed Steiner Tree from Set Covering



$$
\begin{aligned}
V &= \{r\} \cup \mathcal{S} \cup U \\
E &= \{(r, S) : S \in \mathcal{S}\} && \text{cost } c(S) \\
&\quad \cup \{(S, u) : S \in \mathcal{S}, u \in S\} && \text{cost } 0.
\end{aligned}
$$