Fall 2006

CS598CC: Approximation Algorithms

Chandra Chekuri

Administrivia

http://www.cs.uiuc.edu/homes/chekuri/teaching/fall2006/approx.htm

Grading: 4 home works (60-70%), 1 take home final (30-40%)

Mailing list: cs598cc@lists.cs.uiuc.edu

Office hours: Tue-Thu, 1pm-2pm and by appointment

Pre-reqs: grad algorithms *or* undergrad algs + good math background

Recommended text book: Approximation Algorithms by Vijay Vazirani, Springer, 2004.

Other sources: Skeleton lecture notes, course notes from various places: links on web site

Course objectives

Appreciate that not all NP Optimization problems are the same from approximation point of view

Techniques for design and analysis of approx algorithms via some fundamental problems

Ability to relate new problems to known problems

Ability to design/analyze algorithms for new problems

Concepts and example of hardness of approximation

Syllabus

Motivation, intro etc

Techniques

Problems

See webpage for more details

P vs NP and consequences

P: class of polynomial time solvable problems

theoretical notion of efficiency/solvability

$\neq$?

NP: class of polynomial time *verifiable* problems

many problems that arise in practice

P vs NP and consequences

P: class of polynomial time solvable problems

theoretical notion of efficiency/solvability

$\neq$?

NP: class of polynomial time *verifiable* problems

many problems that arise in practice

Q: What to do while we await the answer?

P vs NP and consequences

P: class of polynomial time solvable problems

 theoretical notion of efficiency/solvability

$\neq$?

NP: class of polynomial time *verifiable* problems

 many problems that arise in practice

Q: What to do while we await the answer?

Practice: Heuristics that settle for sub-optimality

Theory: principled study of heuristics

Heuristics

Many approaches:

- mathematical programming
- constraint/logic programming
- local search
- genetic algorithms, simulated annealing, tabu search, …
- *approximation algorithms*

Operations Research, Engineering, CS, …

Approximation Algorithms

Computer Science Theory view point

- in the broader context of computational complexity: what can computers solve given some resources (time, space, randomness)
- explain why problems behave differently
- provably good guarantees on resource usage and quality
- worst case approach (mostly)
- emphasis on analysis/tools/structure/math
- less emphasis on practical engineering aspects/non-quantifiable approaches

Approximation Algorithm

Crudely (formal definitions later)

An efficient algorithm (polynomial time etc)

For every instance of the problem, the *value* of the solution output by algorithm is within some factor of the optimum value (worst case)

Approximation Ratio: factor compared to optimum

Pros

Worst case ratio *robust* in various ways

Explains why problems vary in their difficulty

Analysis often reveals difficult vs easy cases

Tools/algorithmic ideas permeate heuristic development

Can often get good practical heuristics

Sophisticated and beautiful ideas

Cons

- (Usually) Worst case oriented. Can ignore potentially good algorithms/heuristics that work well in practice or on average

- (Usually) no incremental tradeoff of running time for quality of solution (ex: integer programming)

- Limited to cleanly stated problems

- Framework does not directly apply to decision problems/inapproximable problems

Approximation Algorithms broadly

Not limited to NP-hard problems:

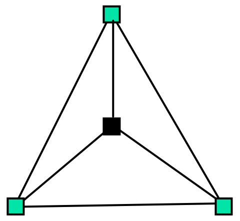
- faster polynomial time algorithms (near linear, sub-linear etc)
- limited space (external memory algorithms)
- limited access to data (streaming algorithms)

Many applications in variety of areas

Steiner tree

Graph $G = (V, E)$, edge costs $c: E \to \mathcal{R}^+$

Terminals: $S \subseteq V$



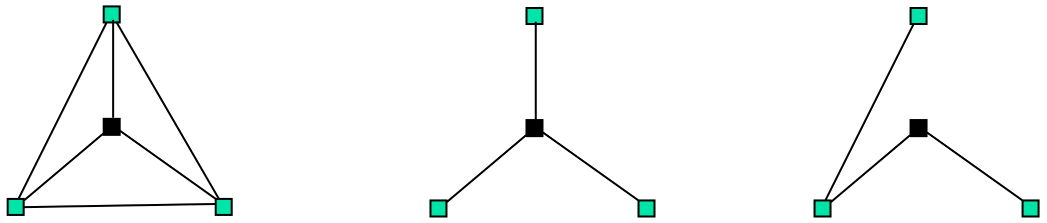
Goal: find a minimum cost tree T that contains all terminals (and hence connects them)

Applications in networking: multicast, VPN

Steiner tree

Graph $G = (V, E)$, edge costs $c: E \to \mathcal{R}^+$

Terminals: $S \subseteq V$



Goal: find a minimum cost tree T that contains all terminals (and hence connects them)

Steiner vertex: a vertex in $V \setminus S$ that belongs to T

Metric completion

Given $G = (V, E)$ with edge weights $c: E \to \mathcal{R}^+$

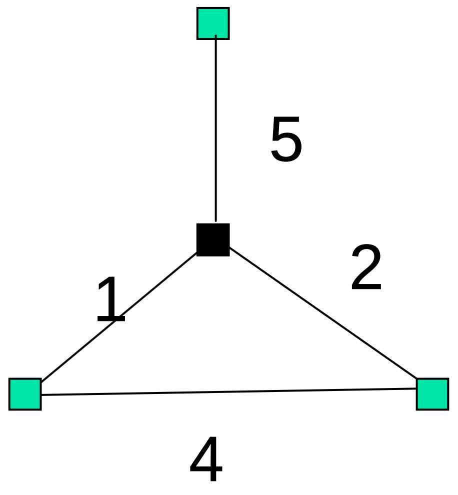
metric-completion of G is a complete graph
$H = (V, E')$ with $c': E \to \mathcal{R}^+$

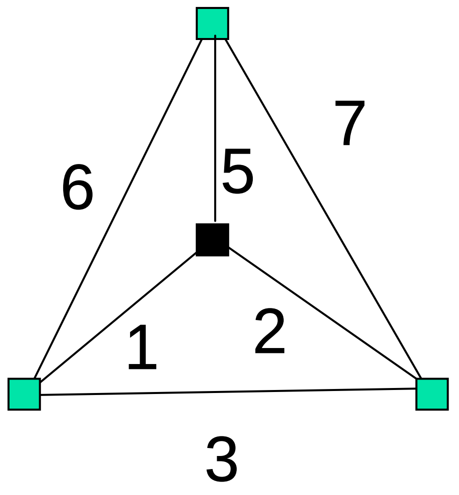
$c'(uv)$ = shortest path distance in G between u and v using edge costs given by c

Note: c' satisfies triangle inequality (forms a metric on V)

$c'(uv) + c'(vw) \geq c'(uw)$ for all u,v,w

Metric completion





Observation: can restrict attention to H for Steiner trees (and several other problems)

Why?

What’s an easy case?

$|S| = 2$ (shortest path)

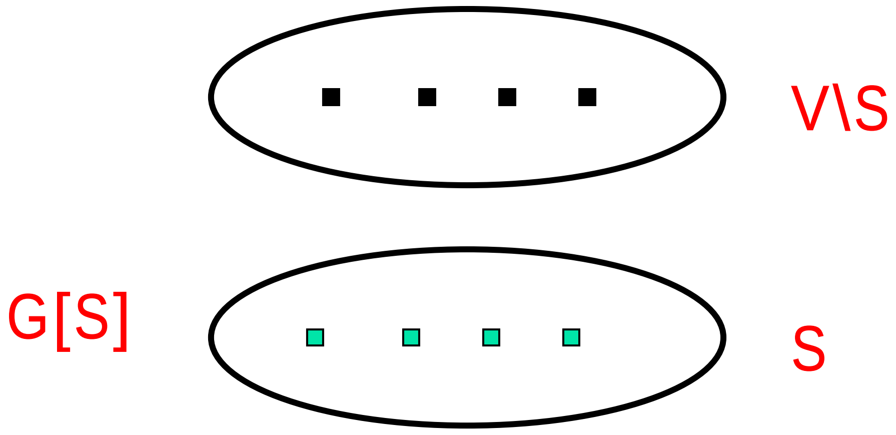
$S = V$

Solution? MST - minimum spanning tree

MST based heuristic

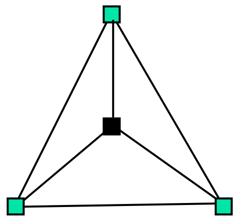
Find MST T in G[S] the induced graph on S

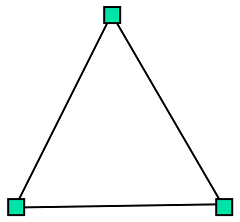
Output T

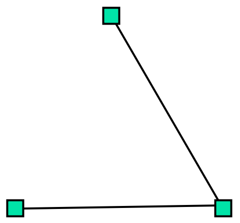


Ignore Steiner vertices!

Example


G


G[S]


T

MST heuristic: how good is it?

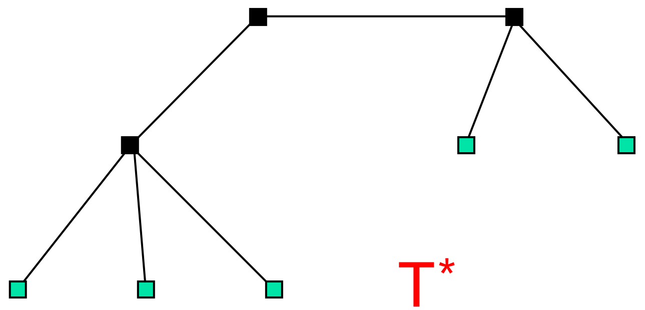
T^* : optimum solution in G, might include Steiner vertices

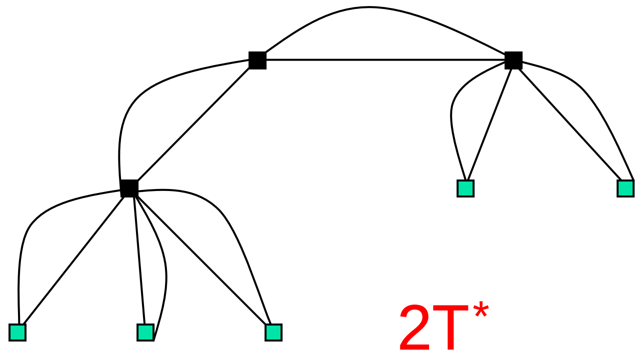
Claim: $c(T) \leq 2\, c(T^*)$

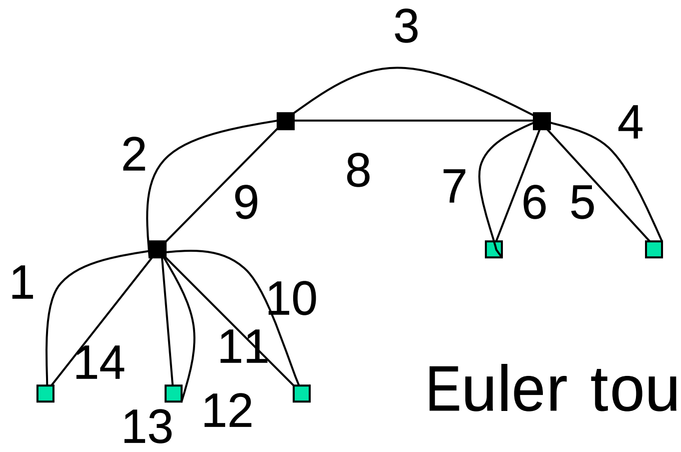
Proof: exhibit a tree T' in $G[S]$ s.t $c(T') \leq 2\, c(T^*)$

$c(T) \leq c(T') \leq 2\, c(T^*)$

Euler tour of 2T*

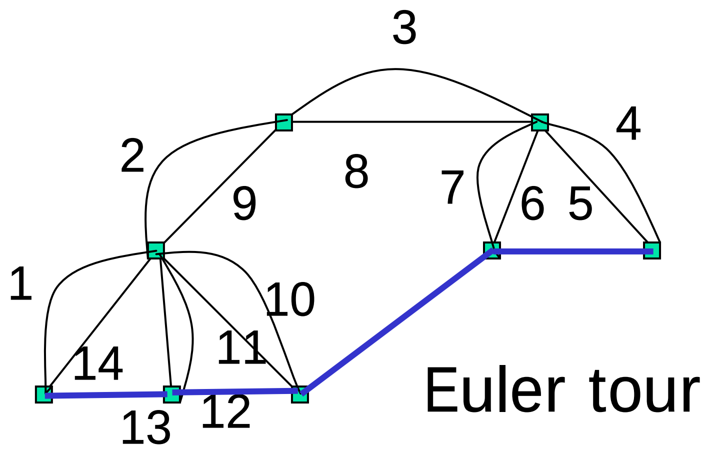


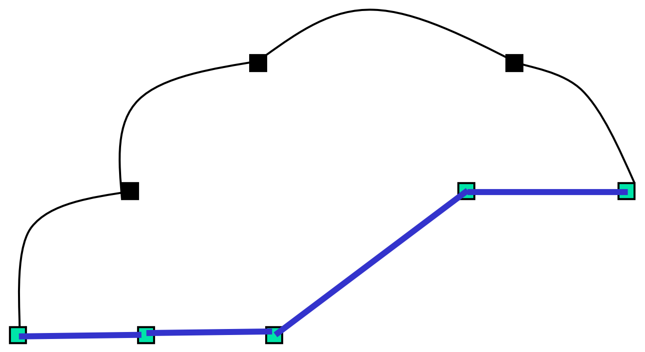




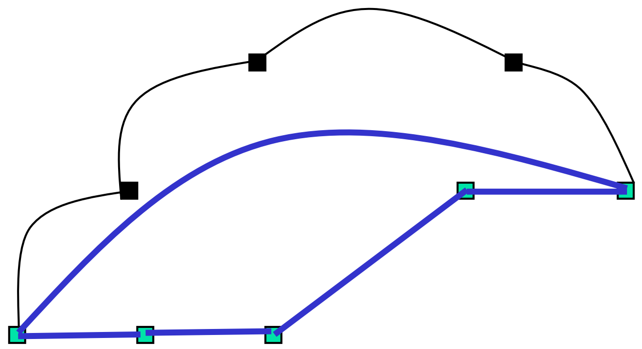
Euler tour

Shortcutting Euler tour





Eliminate Steiner vertices



Eulerian Graph/Tour

Multi-graph G = (V, E) (no loops, parallel edges)

Eulerian tour: a walk in G that visits each *edge* in E exactly once

Fact: G is has an Eulerian tour iff all vertices have *even* degree

If G is directed then, iff each vertex has in-degree equal to out-degree

Eulerian Graph/Tour

Fact: G is has an Eulerian tour iff all vertices have *even* degree

Corollary: Given any undirected G, duplicating edges creates an Eulerian graph

Fact: In a connected graph and Eulerian tour visits a vertex v, d(v)/2 times

Exercise: Check and find an Eulerian tour in O(|E|) time

Analysis

Double T^* and take an Euler tour of $2T^*$

Shortcut Euler tour of $2T^*$ and eliminate Steiner vertices to obtain a tour of S in G[S]

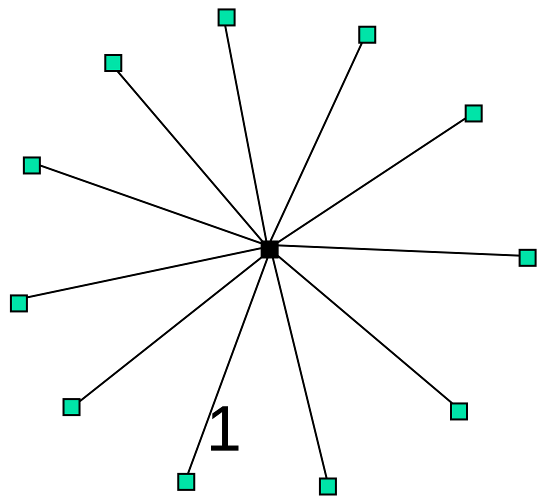
Tour of S in G[S] is at least as expensive as MST

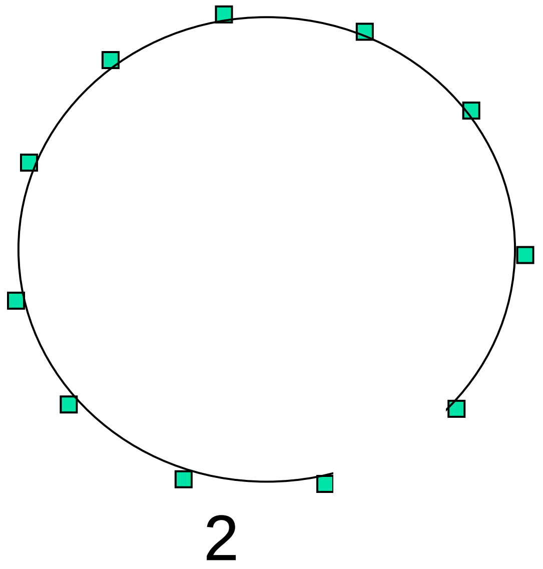
so

$c(T) \leq c(\text{tour}) \leq c(2T^*) \leq 2\ \text{OPT}$

Exercise: show bound is actually $2\text{OPT}(1-1/|S|)$

Tight example





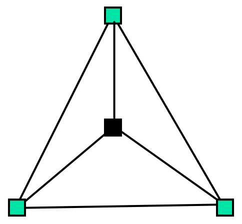
OPT = k

MST(G[S]) = 2(k-1)

Can tight example hold in $\mathcal{R}^2$

MST heuristic is a 2/sqrt{3} $\simeq$ 1.154 approximation in the Euclidean plane

Tight example!



Running time?

$n = |V|$

$m = |E|$

$k = |S|$

Bottleneck? computing edge costs in $G[S]$

$$O(k(n \log n + m))$$

Greedy algorithm for Steiner tree

Order terminals in S: $v_1, v_2, \dots, v_k$

$S_i = \{v_1, v_2, \dots, v_i\}$

```
Start with $T = \{v_1\}$
for $i = 2$ to $k$ do
    find shortest path $P_i$ from $v_i$ to $V(T)$ in $G$
    add $P_i$ to $T$
```

Analysis of Greedy

Theorem: Greedy provides an $2 \ln k$ approx and there are examples where it produces an $\Omega(\log k)$ approx

Advantage of Greedy: online algorithm

Greedy vs MST heuristic

Think of Prim’s algorithm for MST

Prim’s algorithm as MST heuristic

```
Start with $T = v_1$
for $i = 2$ to $|S|$ do
  let $u$ be the terminal in $S \setminus V(T)$ closest to $V(T)$
  add $u$ to $T$ via edge to closest neighbour in $V(T)$
```

Ordering of vertices is *adaptive*

Analysis of Greedy

Theorem: Greedy provides an $O(\log k)$ approx.

$c(i)$ = $\mathrm{cost}(P_i)$ cost paid by v_i

$\mathrm{cost}(T) = c(1)\ (=0) + c(2) + c(3) + \ldots. + c(k)$

$i_1, i_2, \ldots, i_k$ permutation of $\{1,2,\ldots,k\}$ s.t

$$c(i_1) \geq c(i_2) \geq \ldots c(i_k) = 0$$

Claim: $c(i_j) \leq 2\mathrm{OPT}/j$

Analysis of Greedy

Theorem: Greedy provides an $O(\log k)$ approx.

Claim: $c(i_j) \leq 2\,OPT/j$

Assuming the claim:

$$\begin{aligned} c(T) &= c(i_1) + c(i_2) + \dots + c(i_k) \\ &\leq 2OPT/1 + 2OPT/2 + \dots + 2OPT/k \\ &\leq 2\,(1 + 1/2 + \dots + 1/k)\,OPT \\ &\leq 2\,H_k\,OPT \leq 2 \ln k\ OPT \end{aligned}$$

Proof of Claim

Claim: $c(i_j) \leq 2OPT/j$

Consider $A_j = \{v_1, v_{i_1}, v_{i_2}, \ldots, v_{i_j}\}$

Suppose $c(i_j) > 2OPT/j$ then any two vertices in A_j are at distance $> 2OPT/j$ implies

$MST(A_j) > 2OPT$

However $MST(A_j) \leq 2OPT$ (take optimum tree T^* and use Eulerian trick to show this as before)

Other results on Steiner trees

For points in Euclidean plane or small dimensions a polynomial time approximation scheme (PTAS) $(1+\varepsilon)$ approx for any $\varepsilon > 0$

Best known approximation ration: 1.55 using MST heuristic + local search

An LP relaxation that is conjectured to be very good, worst gap known is 8/7

Approximation Algorithm/Ratio

Minimization problem Π: $\mathcal{A}$ is an approximation algorithm with *(relative)* approximation ratio α iff

- $\mathcal{A}$ is polynomial time algorithm
- for all instance I of Π, $\mathcal{A}$ produces a feasible solution $\mathcal{A}$(I) such that
  value($\mathcal{A}$(I)) $\leq \alpha$ value(OPT(I))

Remark: α can depend in size of I, hence technically it is α(|I|).

Example: α(|I|) = log n