Approximation Algorithms, I: Traveling Salesman

Slides by Carl Kingsford

Jan. 20, 2014

Approximation Algorithms

- How do we deal with problems where we don't have an efficient algorithm?
- One option: heuristics
- But we'd like some guarantee: the answer we get should never be *too* far from the optimal.
- $\longrightarrow$ Approximation Algorithms

The desired statement

Definitions:

- Let $A_P(I)$ be the value of the solution using algorithm A to instance I of some minimization problem P
- Let $OPT_P(I)$ be the optimal (smallest) solution for instance I.

Goal: To say we have an approximation algorithm for a minimization problem P, we want to prove something like:

For any instance I, $A_P(I) \leq \alpha(|I|)OPT_P(I)$.

for some function $\alpha(|I|)$.

$\alpha(n)$ might be a constant like "2" or maybe $O(\log n)$, etc.

Approximation Guarantee

Approximation Guarantee:

For any instance I, $A(I) \leq \alpha(|I|)OPT(I)$.

Clearly, $\alpha \geq 1$ (for minimization problems) because we can't have a solution smaller than the optimal.

Want $\alpha(\cdot)$ to be as small as possible.

For example, if $\alpha(n) = 2$, we have the statement that the solution returned by our greedy algorithm is never more than twice as large as the optimal.

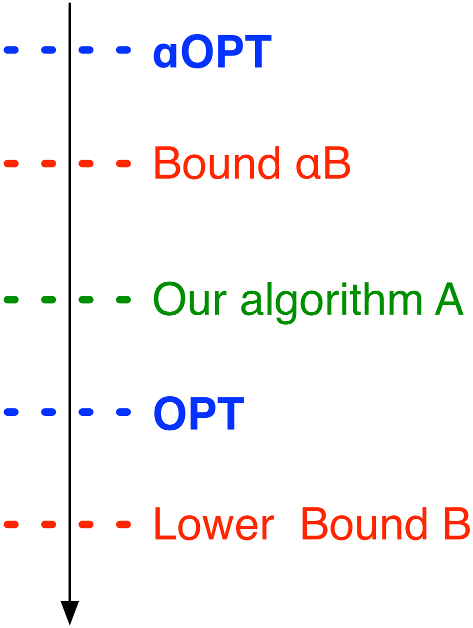
Lower Bounds

Analysis Problem: We don't know the optimal, so how do we compare against it?

Insight: A lower bound on the optimal works almost as well:

- Suppose we know that $B(I) \leq OPT(I)$ for some function B.
- If we can prove $A(I) \leq \alpha B(I)$, then that immediately implies that $A(I) \leq \alpha OPT(I)$.

Lower Bounds, Picture



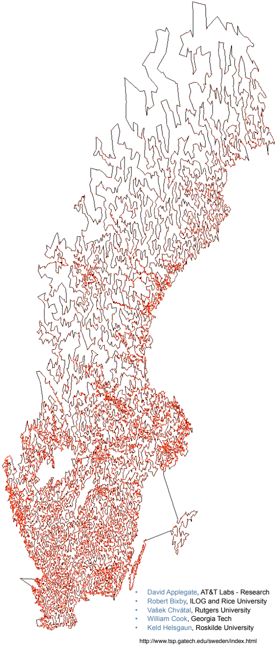
Euclidean Traveling Salesman

Euclidean TSP

Given n cities, with distances $d(u, v)$ between them (that satisfy the triangle inequality), find the order to visit them that minimizes the length of the route.

Can think of input as a complete graph G with $\binom{n}{2}$ edges.

TSP Large Instance



- TSP visiting 24,978 (all) cities in Sweden.
- Solved by David Applegate, Robert Bixby, Vašek Chvátal, William Cook, and Keld Helsgaun
- `http://www.tsp.gatech.edu/sweden/index.html`
- Lots more cool TSP at `http://www.tsp.gatech.edu/`

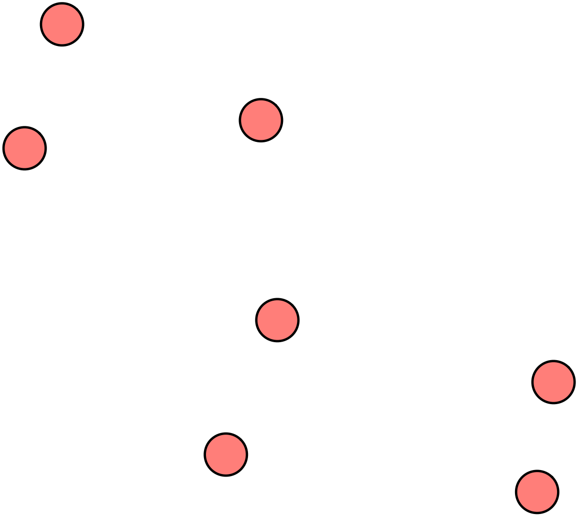
Approximation Algorithm

Euclidean TSP Approximation Algorithm:

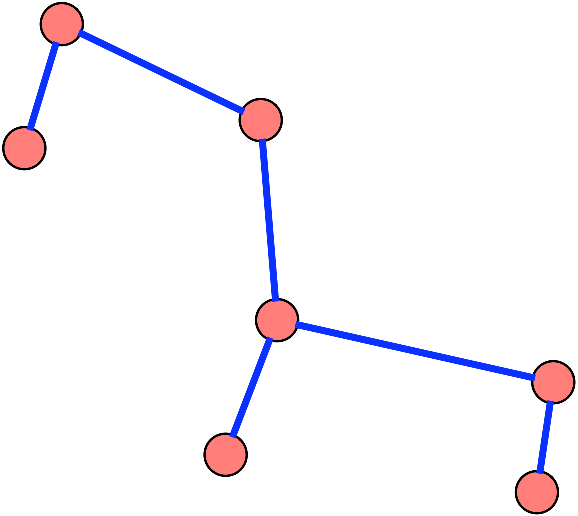
1. Compute a minimum spanning tree T connecting the cities.
2. Visit the cities in order of a preorder traversal of T.

"Preorder traversal" = visit a node, then the entire subtree of its first child, then the entire subtree of the second child, etc.

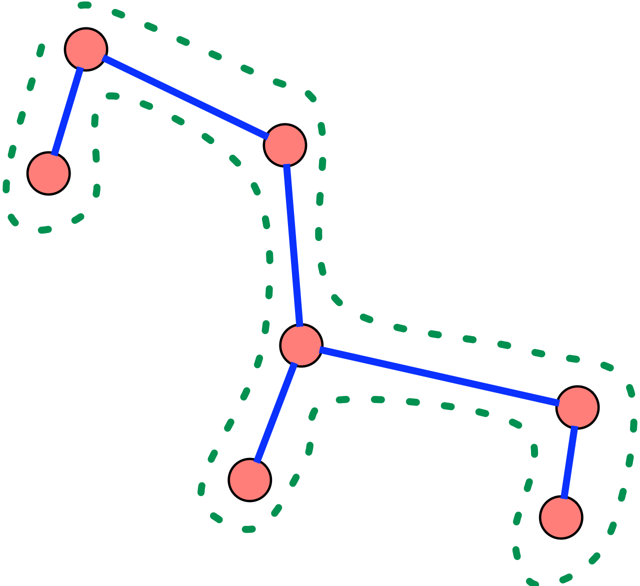
Example



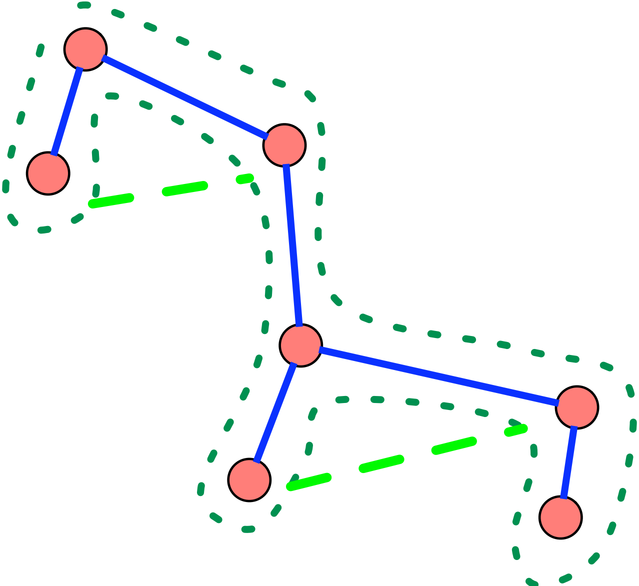
Example



Example



Example



TSP Approximation Algorithm

Notation:

- Let $\mathrm{cost}(A)$ be the total length of the edges in some set A.
- Let A^* be the edges visited on the optimal tour.
- Let A be the edges visited on the tour found by our algorithm.

Theorem. $cost(A) \leq 2cost(A^*)$.

(The algorithm gives a 2-approximation to the optimal TSP.)

Proof

Proof. The cost of a minimum spanning tree T is less than the cost of the optimal tour: $\mathrm{cost}(T) \leq \mathrm{cost}(A^*)$. Why?

A full walk W that "traces" the MST is of length $2\mathrm{cost}(T)$ because every edge is crossed twice.

So: $\mathrm{cost}(W) = 2\mathrm{cost}(T) \leq 2\mathrm{cost}(A^*)$.

W isn't a tour because it visits cities more than once. We can shortcut all but the *first* visit to a city. By the triangle inequality, this only reduces the cost of the tour.

So: $\mathrm{cost}(A) \leq 2\mathrm{cost}(T) \leq 2\mathrm{cost}(A^*)$. $\square$

Approximation Algorithms Summary

- A way to deal with hard problems.
- Analysis main idea: good lower bounds to "approximate" optimal.
- A constant-factor approximation algorithm for Metric Traveling Salesman uses MST.

We will see additional approximation algorithms toward the end of the course.