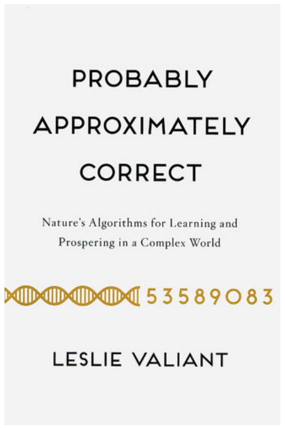
Approximation Algorithms!


PROBABLY APPROXIMATELY CORRECT

Nature's Algorithms for Learning and Prospering in a Complex World

LESLIE VALIANT

Slides adapted from Tandy Warnow, Sandip Das

Admin

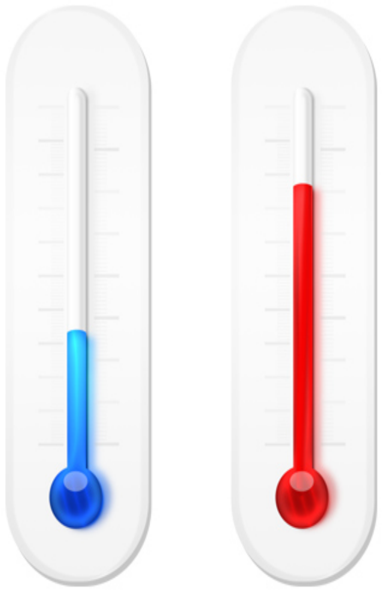
- Two more problem sets
  - 13b due Tuesday after Thanksgiving
  - 14b due following Tuesday
- Ahead
  - Parallel Algorithms
  - Wrap-up
  - Review
- Midterm 2

Coping With NP-Hardness

- Brute-force algorithms.
  - Develop clever enumeration strategies.
  - Guaranteed to find optimal solution.
  - No guarantees on running time.

- Heuristics.
  - Develop intuitive algorithms.
  - Guaranteed to run in polynomial time.
  - No guarantees on quality of solution.

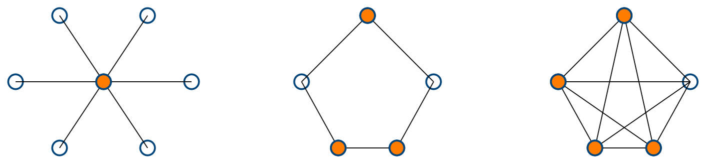
- **TODAY:** Approximation algorithms.
  - Guaranteed to run in polynomial time.
  - Guaranteed to find "high quality" solution, say within 1% of optimum.
  - Obstacle: need to prove a solution's value is close to optimum, without even knowing what the optimum value is!



Vertex Cover

Vertex cover: a subset of vertices which "covers" every edge.
An edge is covered if one of its endpoint is chosen.

The Minimum Vertex Cover Problem:
Find a vertex cover with minimum number of vertices.



Alternatives

- ❖ Special graph classes
  e.g. vertex cover in bipartite graphs, perfect graphs.
- ❖ Fixed parameter algorithms
  find a vertex cover of size k efficiently for small k.
- ❖ Average case analysis
  find an algorithm which works well on average.
- ❖ **Approximation algorithms**
  find an algorithm which return solutions that are
  guaranteed to be close to an optimal solution.

Approximation Algorithms

Key: provably close to optimal.

Let OPT be the value of an optimal solution,
and let SOL be the value of the solution that our algorithm returned.

Additive approximation algorithms: SOL <= OPT + c for some constant c.
Very few examples known: edge coloring, minimum maximum-degree spanning tree.

Constant factor approximation algorithms: SOL <= cOPT for some constant c.
Many more examples known.

Approximation Algorithms

- Suppose we want to find a *minimum* cost solution to some problem (e.g., smallest vertex cover, minimum cost tour of a graph, minimum cost Steiner tree, etc.)
- We define the relative cost of the solution S to be

for Maximization

$$1/\rho \leq \text{Cost}(S)/\text{Cost}(S^{opt}) \leq \rho$$

for Minimization

where S^{opt} is the best solution.

- An algorithm has *ratio* ρ if it always returns a solution with relative cost at most ρ.

Approximation Algorithms and Schemes

- ρ-approximation algorithm.
  - An algorithm A for problem X that runs in polynomial time.
  - For every problem instance, A outputs a feasible solution within ratio ρ of true optimum for that instance.
- Polynomial-time approximation scheme (PTAS).
  - A family of approximation algorithms $\{A_\varepsilon : \varepsilon > 0\}$ for a problem X.
  - A_ε is a $(1 + \varepsilon)$ - approximation algorithm for X.
  - A_ε runs in time polynomial in the input size for a fixed ε.
- Fully polynomial-time approximation scheme (FPTAS).
  - PTAS where A_ε is runs in time polynomial in input size and $1 / \varepsilon$.

Vertex Cover Problem: Ideas?

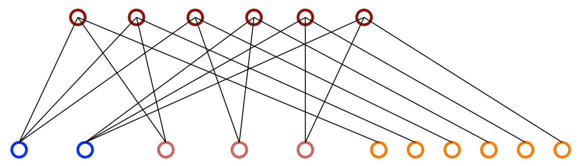
Vertex Cover: Greedy Algorithm 1

Idea: Keep finding a vertex which covers the maximum number of edges.

Greedy Algorithm 1:
1. Find a vertex v with maximum degree.
2. Add v to the solution and remove v and all its incident edges from the graph.
3. Repeat until all the edges are covered.

How good is this algorithm?

Vertex Cover: Greedy Algorithm 1



OPT = 6, all red vertices.

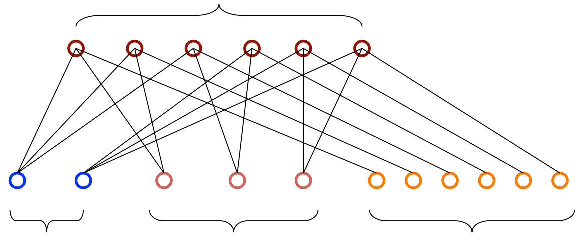
SOL = 11, if we are unlucky in breaking ties.
First we might choose all the blue vertices.
Then we might choose all the pink vertices.
And then we might choose all the orange vertices.

Vertex Cover: Greedy Algorithm 1

k! vertices of degree k

Not a constant factor approximation algorithm!

Generalizing the example!



k!/k vertices of degree k

k!/(k-1) vertices of degree k-1

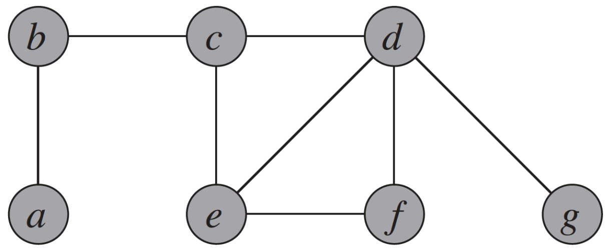
k! vertices of degree 1

OPT = k!, all top vertices.

SOL = k! (1/k + 1/(k-1) + 1/(k-2) + … + 1) ≈ **k! log(k)**, all bottom vertices.

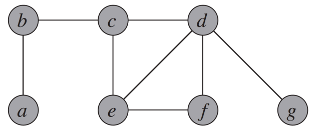
Approximate Vertex Cover

```
APPROX-VERTEX-COVER$(G)$
1  $C = \emptyset$
2  $E' = G.E$
3  while $E' \neq \emptyset$
4      let $(u, v)$ be an arbitrary edge of $E'$
5      $C = C \cup \{u, v\}$
6      remove from $E'$ every edge incident on either $u$ or $v$
7  return $C$
```



Worksheet:

- Find an example graph for which Approx-VC does not return an optimal Vertex Cover



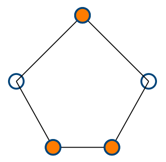
```
APPROX-VERTEX-COVER$(G)$
1  $C = \emptyset$
2  $E' = G.E$
3  while $E' \neq \emptyset$
4      let $(u, v)$ be an arbitrary edge of $E'$
5      $C = C \cup \{u, v\}$
6      remove from $E'$ every edge incident on either $u$ or $v$
7  return $C$
```

Vertex Cover: Greedy Algorithm 2

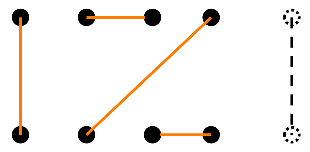
In bipartite graphs, maximum matching = minimum vertex cover.

In general graphs, this is not true.

How large can this gap be?



Vertex Cover: Greedy Algorithm 2



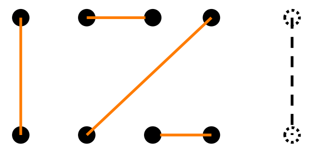
Fix a maximum matching. Call the vertices involved **black**.

Since the matching is maximum, every edge must have a black endpoint.

So, by choosing all the black vertices, we have a vertex cover.

SOL <= 2 * size of a maximum matching

Vertex Cover: Greedy Algorithm 2



What about an optimal solution?

Each edge in the matching has to be covered by a **different** vertex!

OPT >= size of a maximum matching

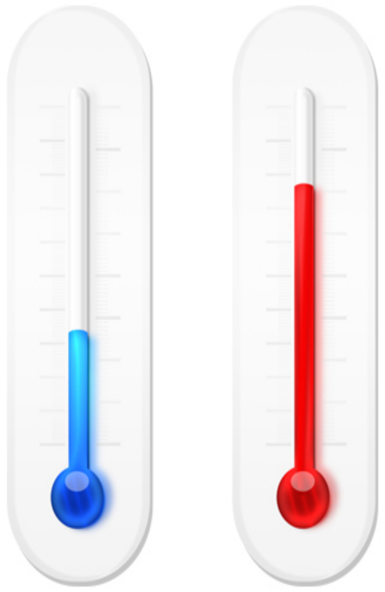
So, OPT <= 2 SOL, and we have a 2-approximation algorithm!

Vertex Cover: Greedy Algorithm 2

Approximate min-max theorem:

Maximum matching <= minimum vertex cover <= 2*maximum matching

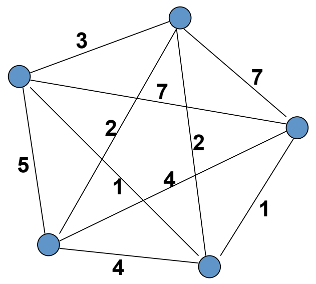
Hardness result: It is NP-complete even to ***approximate*** within a factor of 1.36!!



Vertex Cover: LP Alg 3

Traveling Salesman Problem

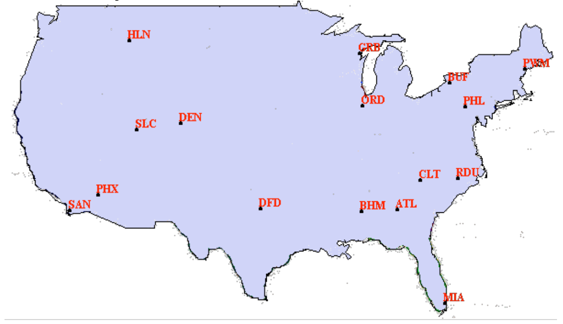
•Given a complete graph with edge weights, determine the shortest tour that includes all of the vertices (visit each vertex exactly once, and get back to the starting point)



Find the minimum cost tour

Traveling Salesperson Problem

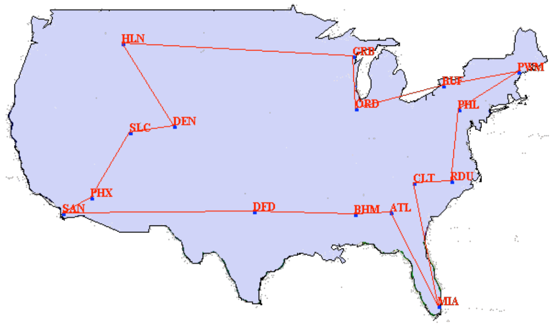
•TSP: Given a graph G = (V, E), nonnegative edge weights c(e), and an integer C, is there a Hamiltonian cycle whose total cost is at most C?



Is there a tour of length at most 1570?

Traveling Salesperson Problem

•TSP: Given a graph G = (V, E), nonnegative edge weights c(e), and an integer C, is there a Hamiltonian cycle whose total cost is at most C?



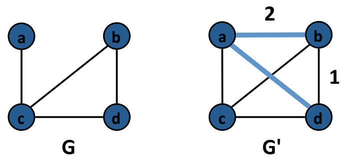
Is there a tour of length at most 1570? Yes, red tour = 1565.

Hamiltonian Cycle Reduces to TSP

•HAM-CYCLE: given an undirected graph G = (V, E), does there exists a simple cycle C that contains every vertex in V.

•TSP: Given a complete (undirected) graph G, integer edge weights c(e) ≥ 0, and an integer C, is there a Hamiltonian cycle whose total cost is at most C?

•Claim. HAM-CYCLE is NP-complete.



•Proof. (HAM-CYCLE transforms to TSP)

- Given G = (V, E), we want to decide if it is Hamiltonian.
- Create instance of TSP with G' = complete graph.
- Set c(e) = 1 if e ∈ E, and c(e) = 2 if e ∉ E, and choose C = |V|.
- Γ Hamiltonian cycle in G ⇔ Γ has cost exactly |V| in G'.
  Γ not Hamiltonian in G ⇔ Γ has cost at least |V|+1 in G'.

TSP

•TSP-OPT: Given a complete (undirected) graph G = (V, E) with integer edge weights c(e) ≥ 0, find a Hamiltonian cycle of minimum cost?

•Claim. If P ≠ NP, there is no ρ-approximation for TSP for any ρ ≥ 1 .

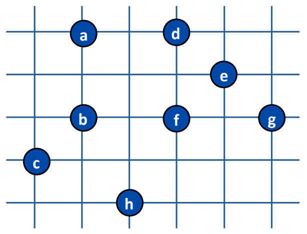
•Proof (by contradiction).

- Suppose A is ρ-approximation algorithm for TSP.
- We show how to solve instance G of HAM-CYCLE.
- Create instance of TSP with G' = complete graph.
- Let C = |V|, c(e) = 1 if e ∈ E, and c(e) = ρ|V|+1 if e ∉ E.
- Γ Hamiltonian cycle in G ⇔ Γ has cost exactly |V| in G'
  Γ not Hamiltonian in G ⇔ Γ has cost more than ρ |V| in G'
- Gap ⇒ If G has Hamiltonian cycle, then A must return it.

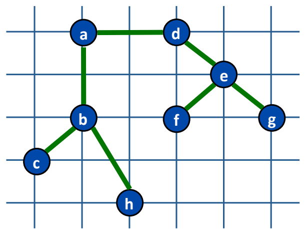
TSP Heuristic

•APPROX-TSP(G, c)

– Find a minimum spanning tree T for (G, c).



Input
(assume Euclidean distances)



MST

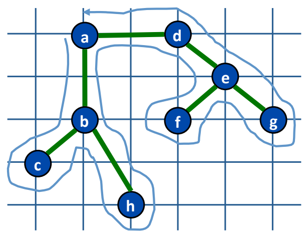
TSP Heuristic

•APPROX-TSP(G, c)

– Find a minimum spanning tree T for (G, c).

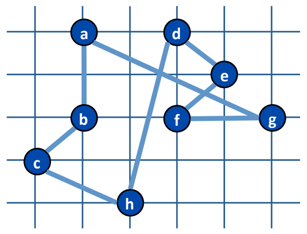
– W ← ordered list of vertices in preorder walk of T.

– H ← cycle that visits the vertices in the order L.



Preorder Traversal Full Walk W

a b c b h b a d e f e g e d a

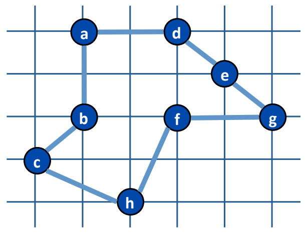


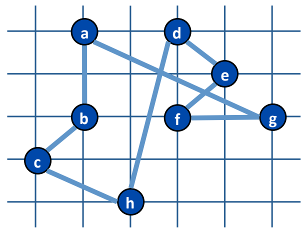
Hamiltonian Cycle H

a b c h d e f g a

TSP Heuristic

- APPROX-TSP(G, c)
  - Find a minimum spanning tree T for (G, c).
  - W ← ordered list of vertices in preorder walk of T.
  - H ← cycle that visits the vertices in the order L.





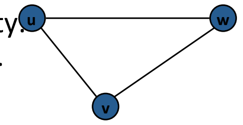
An Optimal Tour: 14.715

Hamiltonian Cycle H: 19.074

(assuming Euclidean distances)

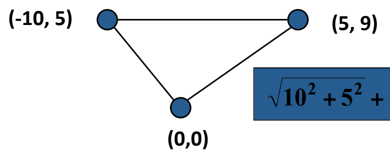
TSP With Triangle Inequality

- Δ-TSP: TSP where costs satisfy Δ-inequality.
  - For all u, v, and w: c(u,w) ≤ c(u,v) + c(v,w).



- Claim. Δ-TSP is NP-complete.
- Proof. Transformation from HAM-CYCLE satisfies Δ-inequality.

- Ex. Euclidean points in the plane.

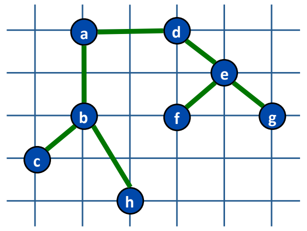


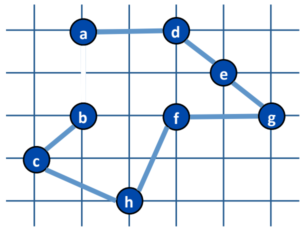
$$\sqrt{10^2+5^2}+\sqrt{5^2+9^2}+\sqrt{15^2+4^2}=37.000\ldots$$

- PTAS for Euclidean TSP (Arora 1996, Mitchell 1996)

TSP With Triangle Inequality

- Theorem. APPROX-TSP is a 2-approximation algorithm for Δ-TSP.
- Proof. Let H* denote an optimal tour. Need to show c(H) ≤ 2c(H*).
  - c(T) ≤ c(H*) since we obtain spanning tree by deleting any edge from optimal tour.



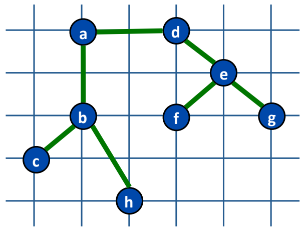


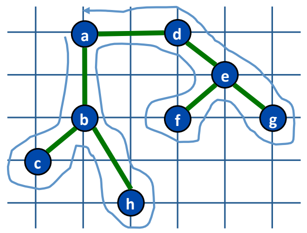
MST T

An Optimal Tour, H*

TSP With Triangle Inequality

- Theorem. APPROX-TSP is a 2-approximation algorithm for Δ-TSP.
- Proof. Let H* denote an optimal tour. Need to show c(H) ≤ 2c(H*).
  - c(T) ≤ c(H*) since we obtain spanning tree by deleting any edge from optimal tour.
  - c(W) = 2c(T) since every edge visited exactly twice.





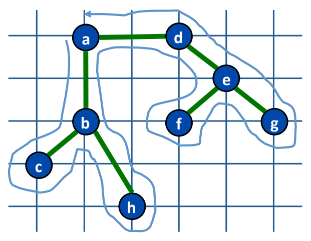
MST T

Walk W

a b c b h b a d e f e g e d a

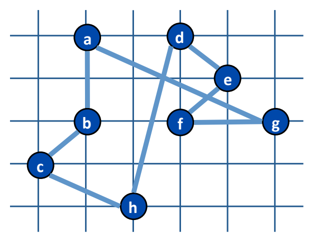
TSP With Triangle Inequality

- Theorem. APPROX-TSP is a 2-approximation algorithm for Δ-TSP.
- Proof. Let H^* denote an optimal tour. Need to show $c(H) \leq 2c(H^*)$.
  - $c(T) \leq c(H^*)$ since we obtain spanning tree by deleting any edge from optimal tour.
  - $c(W) = 2c(T)$ since every edge visited exactly twice.
  - $c(H) \leq c(W)$ because of Δ-inequality.



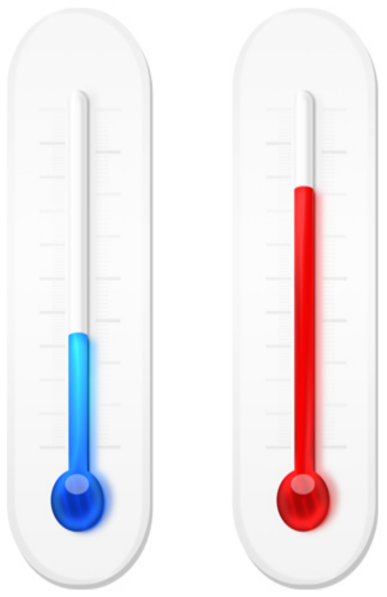
Walk W

a b c b h b a d e f e g e d a



Hamiltonian Cycle H

a b c h d e f g a



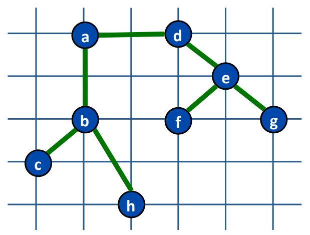
Euclidean TSP

- WORKSHEET: Suppose that the vertices for an instance of the traveling-salesman problem are points in the plane and that the cost c(u,v) is the Euclidean distance between points u and v.

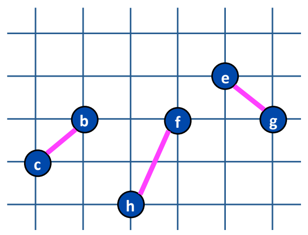
- Can an optimal tour ever cross itself? Why or why not?

TSP: Christofides Algorithm

- Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.
- CHRISTOFIDES(G, c)
  - Find a minimum spanning tree T for (G, c).



MST T

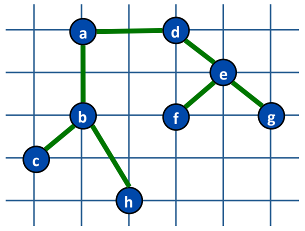


Matching M

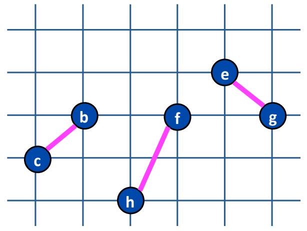
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•CHRISTOFIDES(G, c)

- Find a minimum spanning tree T for (G, c).
- M ← min cost perfect matching of odd degree nodes in T.



MST T

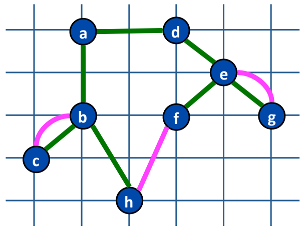


Matching M

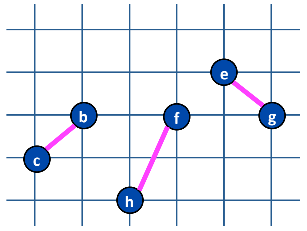
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•CHRISTOFIDES(G, c)

- Find a minimum spanning tree T for (G, c).
- M ← min cost perfect matching of odd degree nodes in T.
- G' ← union of spanning tree and matching edges.



G' = MST + Matching

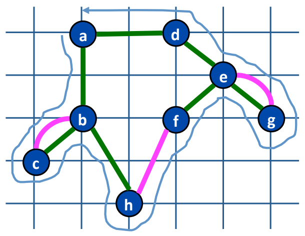


Matching M

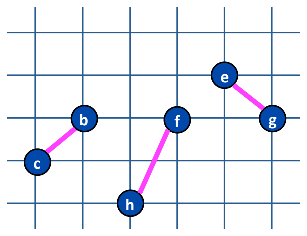
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•CHRISTOFIDES(G, c)

- Find a minimum spanning tree T for (G, c).
- M ← min cost perfect matching of odd degree nodes in T.
- G' ← union of spanning tree and matching edges.
- E ← Eulerian tour in G'.



E = Eulerian tour in G'

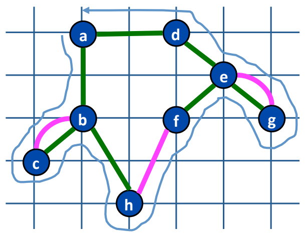


Matching M

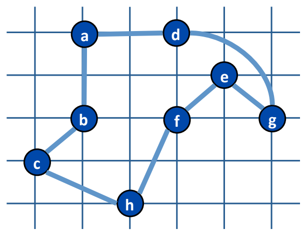
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•CHRISTOFIDES(G, c)

- Find a minimum spanning tree T for (G, c).
- M ← min cost perfect matching of odd degree nodes in T.
- G' ← union of spanning tree and matching edges.
- E ← Eulerian tour in G'.
- H ← short-cut version of Eulerian tour in E.



E = Eulerian tour in G'

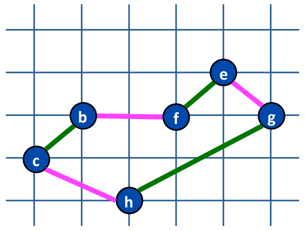


Hamiltonian Cycle H

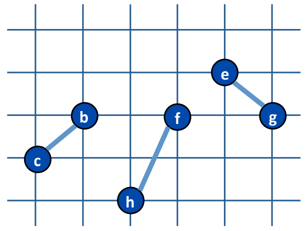
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•Proof. Let H* denote an optimal tour. Need to show c(H) ≤ 1.5 c(H*).

– c(T) ≤ c(H*) as before.
– c(M) ≤ ½ c(Γ*) ≤ ½ c(H*).
  • second inequality follows from Δ-inequality
  • even number of odd degree nodes
  • Hamiltonian cycle on even # nodes comprised of two matchings



Optimal Tour Γ* on Odd Nodes

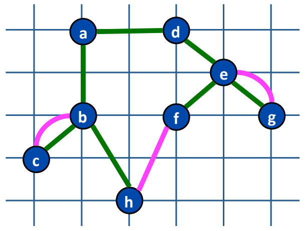


Matching M

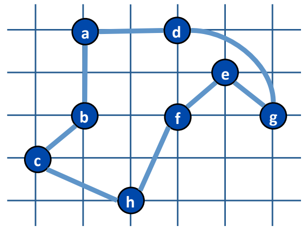
•Theorem. There exists a 1.5-approximation algorithm for Δ-TSP.

•Proof. Let H* denote an optimal tour. Need to show c(H) ≤ 1.5 c(H*).

– c(T) ≤ c(H*) as before.
– c(M) ≤ ½ c(Γ*) ≤ ½ c(H*).
– Union of MST and and matching edges is Eulerian.
  • every node has even degree
– Can shortcut to produce H and c(H) ≤ c(M) + c(T).



MST + Matching



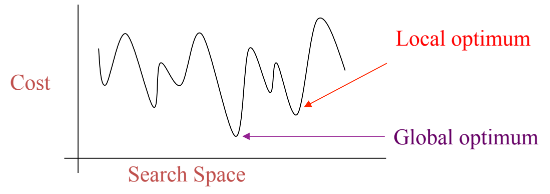
Hamiltonian Cycle H

Using Approximation Algorithms

- Approximation algorithms can give reasonable solutions to NP-hard problems, but sometimes heuristics (without any performance guarantees) can be even better
- Approximation algorithms with performance guarantees can be used to get bounds (both lower and upper) on the best achievable score

Approaches for Local Search

1. Hill-climbing heuristics (which can get stuck in local optima)
2. Randomized algorithms for getting out of local optima
3. Approximation algorithms for MP (based upon Steiner Tree approximation algorithms).


Cost
Local optimum
Global optimum
Search Space