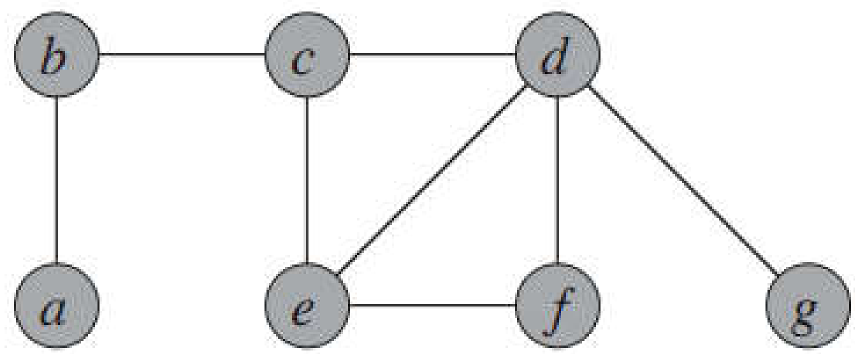
Approximation algorithms 3
Vertex cover, TSP, k-center

CS240 Spring 2020

Rui Fan

Vertex cover

- Input A graph with vertices V and edges E.
- Output A subset V' of the vertices, so that every edge in E touches some vertex in V'.
- Goal Make |V'| as small as possible.



source: *Introduction to Algorithms*, Cormen et al.

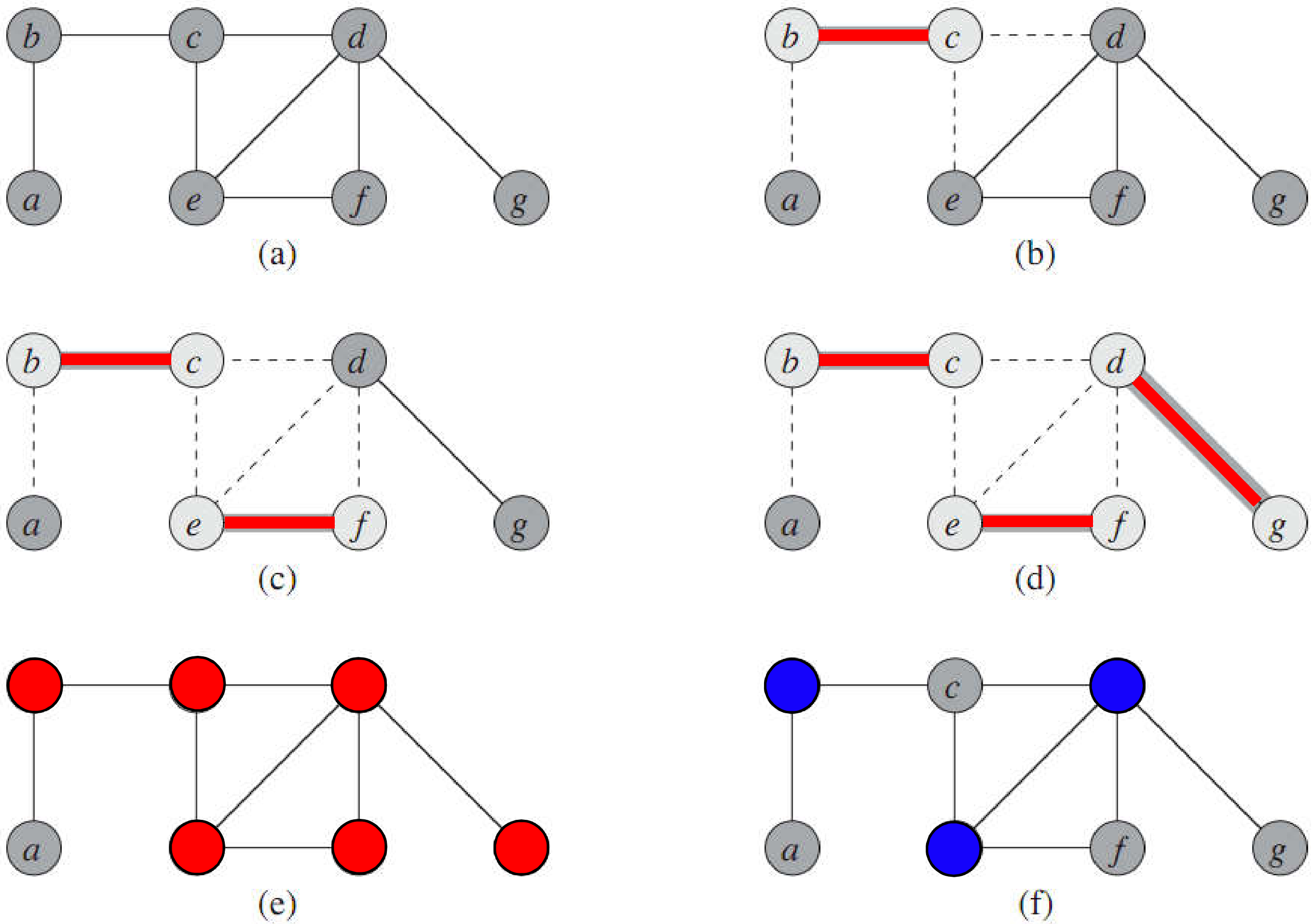
- Finding the minimum vertex cover is NP-complete.
- Vertex cover is a special case of (unweighted) set cover, where each element (edge) can be covered by at most two sets (vertices).
- We'll see a simple 2 approximation for this problem.

A vertex cover algorithm

- Initially, let D be all the edges in the graph, and C be the empty set.
  - C is our eventual vertex cover.
- Repeat as long as there are edge left in D.
  - Take any edge (u,v) in D.
  - Add {u,v} to C.
  - Remove all the edges adjacent to u or v from D.
- Output C as the vertex cover.

Example

source: CLRS



Algorithm's vertex cover

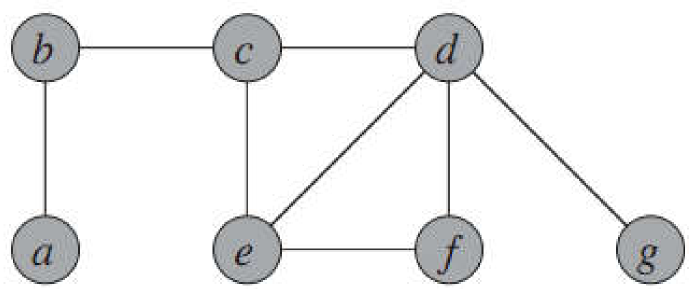
Optimal vertex cover

Proof of correctness

- The output is certainly a vertex cover.
  - In each iteration, we only take out edges that get covered.
  - We keep adding vertices till all edges are covered.
- Now, we show it's a 2 approximation.
- Let C* be an optimal vertex cover.
- Let A be the set of edges the algorithm picked.

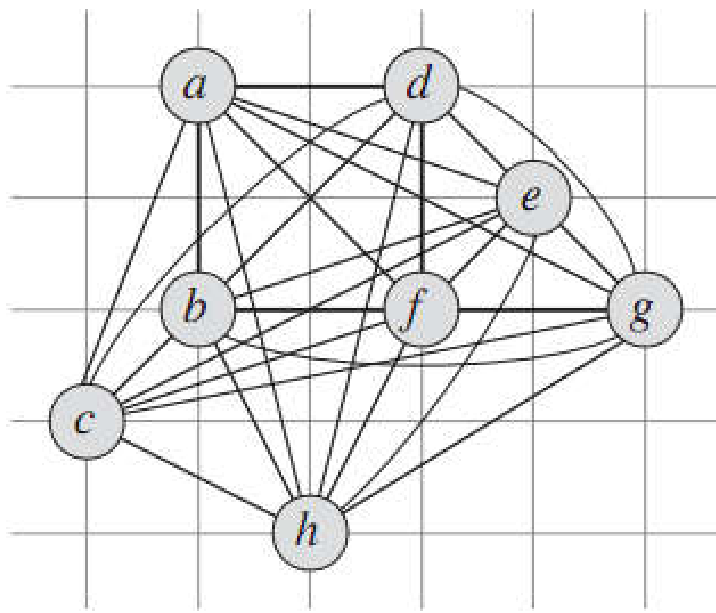
Proof of Correctness

- None of the edges in A touch each other.
  - Each time we pick an edge, we remove all adjacent edges.
- So each vertex in C* covers at most one edge in A.
  - The edges covered by a vertex all touch each other.
- Every edge in A is covered by a vertex in C*.
  - Because C* is a vertex cover.
- So $|C^*| \geq |A|$.
- The number of vertices the algorithm uses is $2|A|$.
  - If alg picks edge (u,v), it uses {u,v} in the cover.
- So (# vertices alg uses) / (# vertices in opt cover) = $2|A| / |C^*| \leq 2|A| / |A| = 2$.



Traveling Salesman Problem

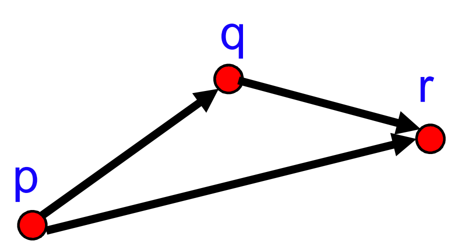
- Input A complete graph with weights on the edges.
- Output A cycle that visits each city once.
- Goal Find a cycle with minimum total weight.



source: CLRS

Metric TSP

- TSP is NP-hard. In fact, it's even NP-hard to approximate when weights can be arbitrary.
- However, TSP is approximable for special types of weights.
- A weighted graph satisfies the triangle inequality if for any 3 vertices p, q, r, we have $d_{pq}+d_{qr} \geq d_{pr}$.
  - I.e., direct path is always no worse than a roundabout path.
  - This is called a metric TSP.
- There is a 1.5-approx algorithm for TSP in graphs with the triangle inequality.
  - Let's look at a simpler 2-approx first.

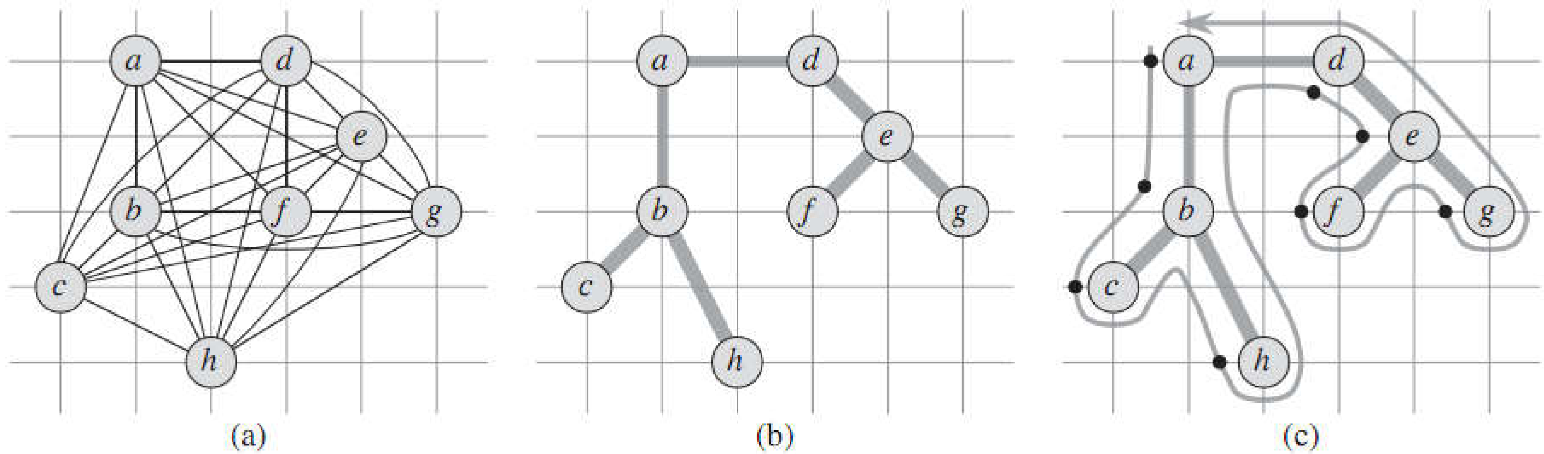


A 2-approximation for TSP

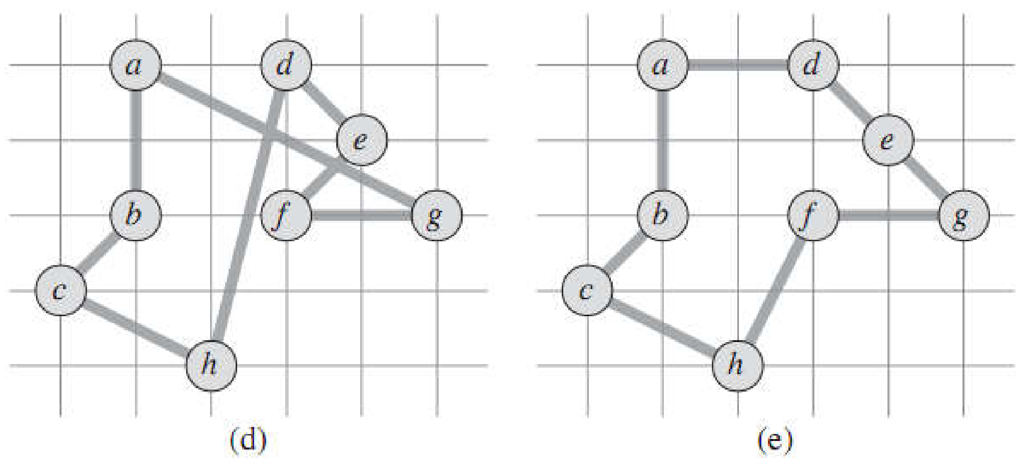
- Construct a minimum spanning tree T on G.
- Use depth-first traversal to visit all the vertices in T, starting from an arbitrary vertex.
- Convert this depth-first traversal T’ to a cycle H that doesn’t revisit any vertex.
- Return H as the TSP tour.

Example

source: CLRS



(a) (b) (c)

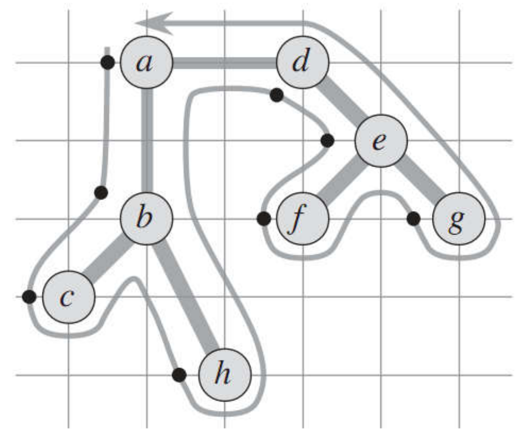


(d) (e)

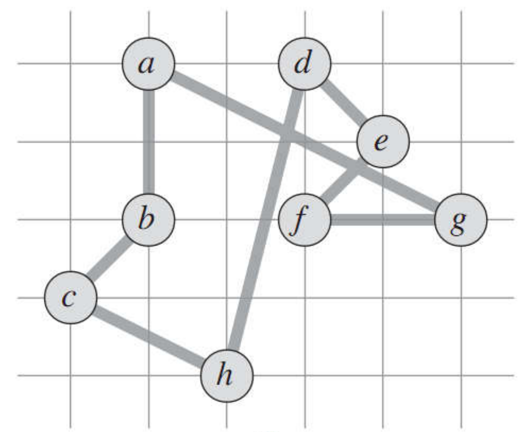
- (b) The MST T.
- (c) visit T in order abcbhbadefegeda.
- (d) converts the tour from (c) to a Hamiltonian cycle, that doesn't revisit any vertices.
- (e) is the optimal TSP.

Making the tour Hamiltonian

- To go from (c) to (d), we need to make a tour T’ that revisits vertices into a cycle H that doesn’t revisit vertices.
- We use shortcutting.
  - If we revisit a vertex in T’, we directly jump to the next vertex in T’ we haven’t visited.
    - We allow revisiting the first vertex.
  - The sequence of vertices we now visit is H.
  - Ex abcbhbadefegeda → abchdefga.



(c)



(d)

Making the tour Hamiltonian

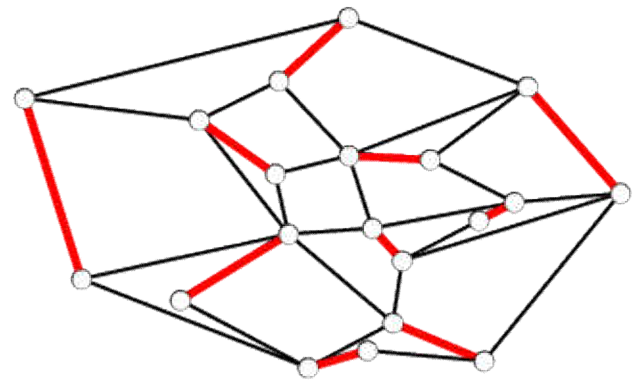
- Lemma If H is the shortcut of T', then c(H)≤c(T').
- Proof We formed H from T' by skipping over some vertices. E.g. we directly went from c to h, skipping over b.
  - But by the triangle inequality, $d_{cb}+d_{bh} \geq d_{ch}$.
    - So shortcutting from c to h didn't increase the distance.
  - The same thing applies to all our shortcuts.
  - So H is no longer than T'.

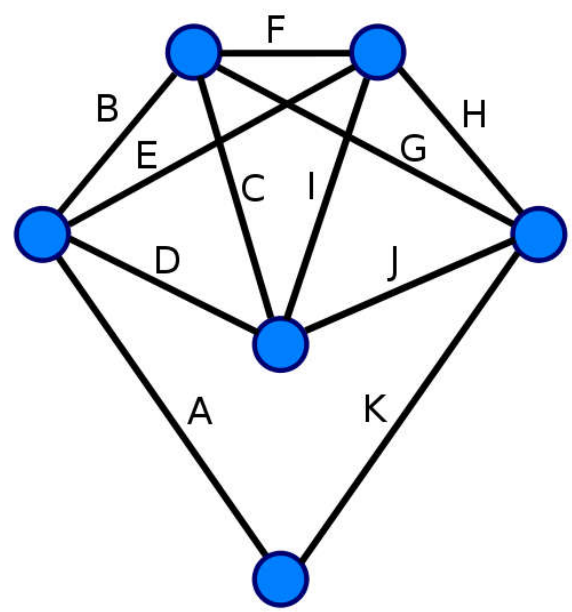
Proof of 2-approximation

- Let H* be an optimum TSP.
- If we delete an edge from H*, we get a spanning tree.
- Since T is an MST, $c(T) \leq c(H^*)$.
- Call the path from the depth-first traversal T'.
  - T' crosses each edge in T twice.
  - So $c(T') = 2\, c(T)$.
- Let H be the outcome of shortcutting T'.
  - H is a Hamiltonian cycle. It visits all the vertices, and ends where it started.
  - $c(H) \leq c(T')$, by the lemma.
  - $c(H) \leq c(T') = 2\, c(T) \leq 2\, c(H^*)$.
- So H is a 2-approximation.

Matchings and Euler cycles

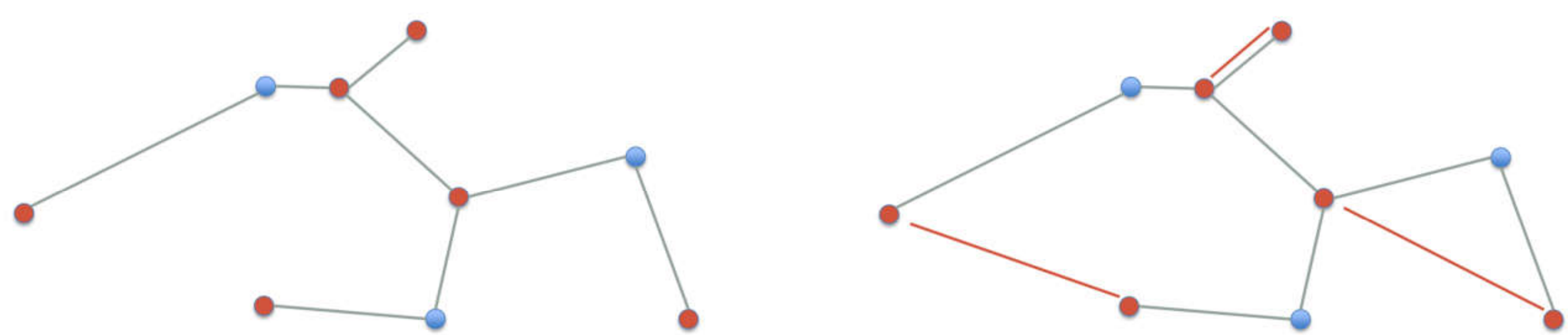
- A matching in a graph is a set of nonintersecting edges.
  - A perfect matching is a matching that includes every vertex.
- An Euler tour of a graph is a path that starts and ends at the same vertex, and visits every edge once.
  - Hamiltonian tour visits every vertex once.
- Thm (Euler) A graph has an Euler tour if and only if all vertices have even degree.
- Note how deciding if graph has Euler tour is trivial, but deciding if it has Hamiltonian tour is NPC!





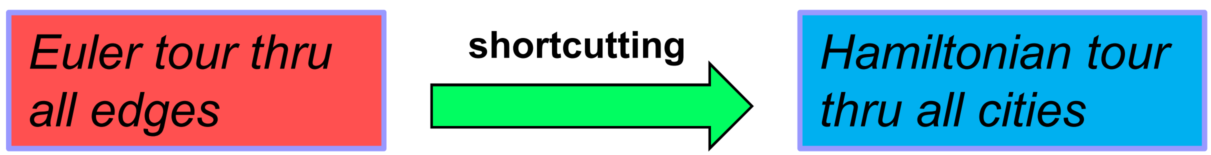
Christofides 3/2-approx algorithm

- A 3/2-approximation for TSP with triangle inequality.
- Construct a minimum spanning tree T on G.
- Find the set V' of odd degree vertices in T.
- Construct a minimum cost perfect matching M on V'.
- Add M to T to obtain T'.
- Find an Euler tour T'' in T'.
- Shortcut T'' to obtain a Hamiltonian cycle H. Output as the TSP.



Why Christofides works well

- In the 2-approx, we found a TSP by “doubling” the MST to an Euler tour, then shortcutting.
  - We need to start with Euler tour before shortcutting to ensure we visit all cities.



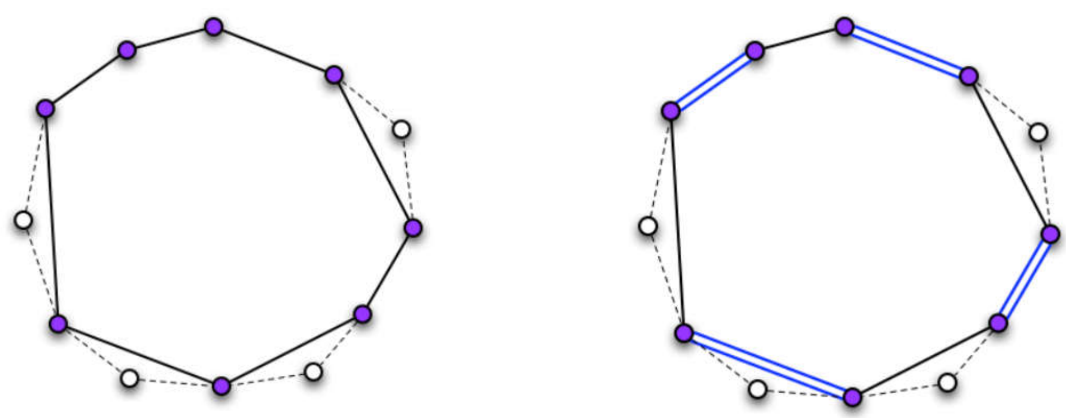
- Key to Christofides is to find a shorter Euler tour, without doubling the MST.
  - A graph with only even degree vertices always has Euler tour.
  - So we want to modify the MST to have all even degrees, by adding a matching.

Proof of correctness

- Lemma T' has an Euler tour.
- Proof There are an even number of vertices in V', because the total degree of T is even.
  - Since G is a complete graph and |V'| is even, there's a perfect matching on V'.
    - The min cost perfect matching can be found in $O(n^2)$ time using the blossom algorithm.
  - The degree of every node in M is odd. Since V' are the odd degree nodes in T, adding M to T makes all nodes in T' have even degree.
  - T' has Euler tour by Euler's theorem.

Proof of correctness

- Lemma Let H^* be an optimal TSP on G, and let m be the cost of M. Then $m \leq c(H^*)/2$.
- Proof Let H' be the optimal TSP on V'.
  - $c(H') \leq c(H^*)$ because H' is an optimal TSP on fewer vertices.
  - H' is a cycle on V', so it consists of two matchings on V'. The cheaper one has cost $m' \leq c(H')/2 \leq c(H^*)/2$.
  - $m \leq m'$ because M has min cost.



http://www.mpi-inf.mpg.de/departments/d1/teaching/ss12/AdvancedGraphAlgorithms/Slides06.pdf

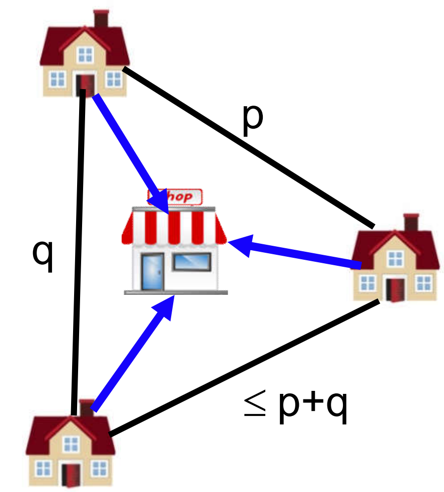
Proof of 3/2-approximation

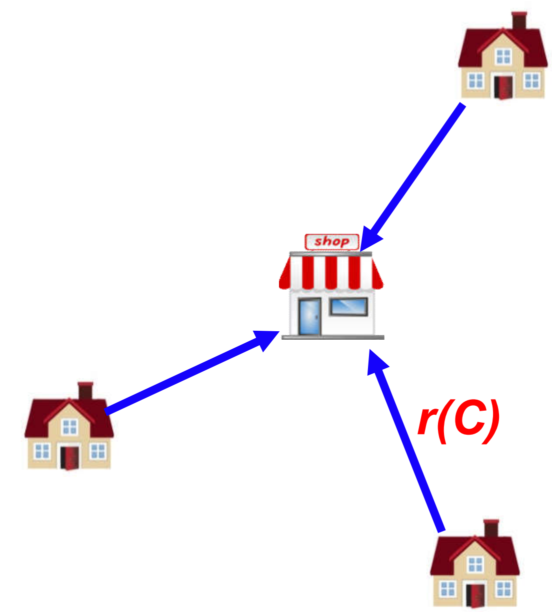
- Thm Let H be the TSP output by Christofides and let H* be an optimal TSP. Then $c(H) \leq 3/2 * c(H^*)$.
- Proof
  - $c(T) \leq c(H^*)$ because T is an MST.
  - $c(T') = c(M) + c(T) \leq c(H^*)/2 + c(H^*) = 3/2 * c(H^*)$.
  - $c(H) \leq c(T')$ because H is the shortcut of T'.

❑*Construct a minimum spanning tree T on G.*
❑*Find set V' of odd-degree vertices in T.*
❑*Construct a minimum cost perfect matching M on V'.*
❑*Add M to T to obtain T'.*
❑*Shortcut T' to obtain a Hamiltonian cycle. Output as the TSP.*

k-Center problem

- Given a city with n sites, we want to build k centers to serve them.
  - Let S be set of sites, C be set of centers.
- Each site uses the center closest to it.
  - Distance of site s from the nearest center is $d(s,C) = \min_{c \in C} d(s,c)$.
- Goal is to make sure no site is too far from its center.
  - We want to minimize the max distance that any site is from its closest center.
    - Minimize $r(C)=\max_{s \in S} \min_{c \in C} d(s,c)$.
  - C is called a cover of S, and r is called C's radius.
  - Where should we put centers to minimize the radius?
- Assume distances satisfy triangle inequality.





Gonzalez's algorithm

- k-Center is NP-complete.
- We'll give a simple 2-approximation for it.
- Idea Say there's one site that's farthest away from all centers. Then it makes the radius large. We'll put a center at that site, to reduce the radius.
  - Note we allow putting center at same location as site.

Gonzalez's algorithm

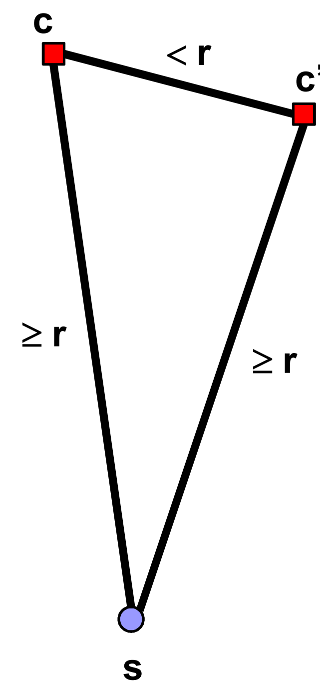
- C is set of centers, initially empty.

- repeat k times
  - choose site s with maximum d(s,C)
  - add s to C
- return C

- Note The centers are located at the sites.

Proof of correctness

- Let C be the algorithm's output, and r be C's radius.
  - $r = \max_{s \in S} \min_{c \in C} d(s,c)$
- Lemma 1 For any $c, c' \in C$, $d(c,c') \geq r$.
- Proof Since r is the radius, there exists a point $s \in S$ at distance $\geq r$ from all the centers.
  - If there's no such s, then C's radius < r.
  - So s is distance $\geq r$ from c and c'.
  - Suppose WLOG c' is added to C after c.
  - If $d(c,c') < r$, then algorithm would add s to C instead of c', since s is farther.

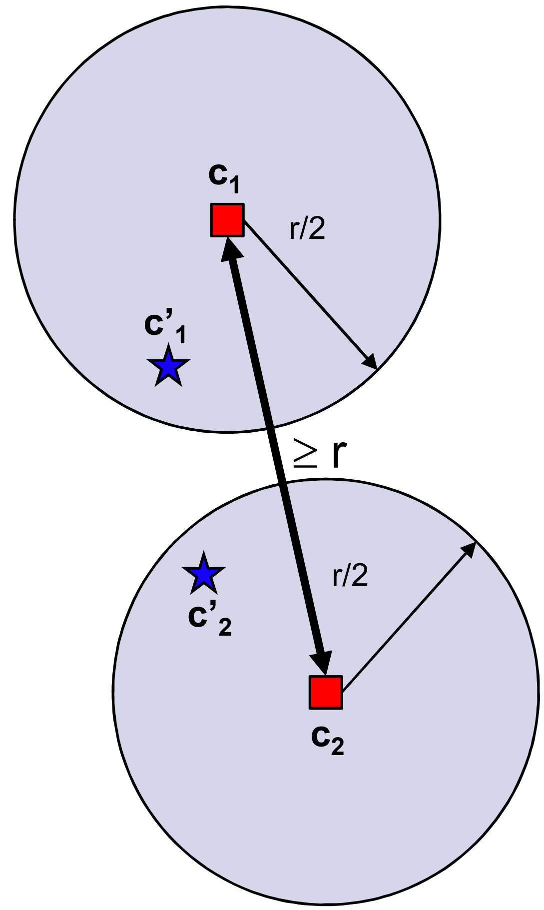


Proof of correctness

- Cor There exist k+1 points mutually at distance $\geq$ r from each other.
  - By the lemma, the k centers are mutually $\geq$ r distance apart.
  - Also, there's an $s \in S$ at distance $\geq$ r from all the centers.
    - Otherwise C's covering radius is $< r$.
  - So the k centers plus s are the k+1 points.
- Call these k+1 points D.

Proof of correctness

- Let C* be an optimal cover with radius r*.
- Lemma 2 Suppose r > 2r*. Then for every $c \in D$, there exists a corresponding $c' \in C^*$. Furthermore, all these c' are unique.
- Proof Draw a circle of radius r/2 around each $c \in D$.
  - There must be a $c' \in C^*$ inside the circle, because
    - c is at most distance r* away from its nearest center, since r* is C*'s radius.
    - $r/2 > r^*$.
  - Given $c_1, c_2 \in D$, let $c'_1, c'_2 \in C^*$ be inside c_1 and c_2's circle, resp.
  - c_1 and c_2's circles don't touch, because $d(c_1, c_2) \geq r$.
  - So $c'_1 \neq c'_2$.



Proof of correctness

- Thm Let C be the output of Gonzalez's algorithm, and let C* be an optimal k-center. Then $r(C) \leq 2r(C^*)$.
- Proof By Lemma 2, if $r(C)>2r(C^*)$, then for every $c \in D$, there is a unique $c' \in C^*$.
  - But there are k+1 points in D, by the corollary.
  - So there are k+1 points in C*. This is a contradiction because C* is a k-center.