Traveling Salesman Problem (TSP)

Input: undirected graph $G=(V,E)$, $c: E \rightarrow R^{+}$

Goal: find a tour (Hamiltonian cycle) of minimum cost

Traveling Salesman Problem (TSP)

Input: undirected graph $G=(V,E)$, $c: E \rightarrow R^{+}$

Goal: find a tour (Hamiltonian cycle) of minimum cost

Q: Is there a reasonable heuristic for this?

Traveling Salesman Problem (TSP)

Input: undirected graph G=(V,E) , c: E $\rightarrow \mathcal{Z}^+$

Goal: find a tour (Hamiltonian cycle) of minimum cost

Q: Is there a reasonable heuristic for this?

Claim: Unless *P=NP* no “good” heuristic. Why?

TSP and Hamiltonian cycle

Claim: Unless P=NP no “good” heuristic. Why?

Can solve NP-Complete Hamiltonian cycle problem using a good heuristic for TSP

Proof: Given graph $G=(V,E)$ create a new graph $H = (V, E')$ where H is a complete graph

Set $c(e) = 1$ if $e \in E$, otherwise $c(e) = B$

TSP is hard

Proof: Given graph G=(V,E) create a new graph H = (V, E') where H is a complete graph
Set $c(e) = 1$ if $e \in E$, otherwise $c(e) = B$

If G has a Hamilton cycle, OPT = n
otherwise $OPT \geq n-1 + B$

Approx alg with ratio better than $(n-1+B)/n$ would enable us to solve Hamiltonian cycle problem

TSP is hard

If G has a Hamilton cycle, OPT = n
otherwise OPT $\geq$ n-1 + B

Approx alg with ratio better than (n-1+B)/n would solve the Hamiltonian cycle problem

Can choose B to be any poly-time computable value (n^2, n^3, ..., 2^n) .

Conclusion: TSP is “inapproximable”

Metric-TSP

$G=(V, E)$, $c: E \rightarrow \mathcal{Z}^+$

Find a min cost tour that visits all vertices

Allow a vertex to be visited *multiple* times

Equivalent to assuming the following:

G is a complete graph

c satisfies triangle inequality: for vertices u,v,w

$$c(uv) + c(vw) \geq c(uw)$$

Why?

Nearest Neighbour Heuristic

Natural Greedy Heuristic:

1. Start at an arbitrary vertex s
2. From the current vertex u, go to the *nearest unvisited* vertex v
3. When all vertices are visited, return to s

Nearest Neighbour Heuristic

Natural Greedy Heuristic:

1. Start at an arbitrary vertex s
2. From the current vertex u, go to the *nearest unvisited* vertex v
3. When all vertices are visited, return to s

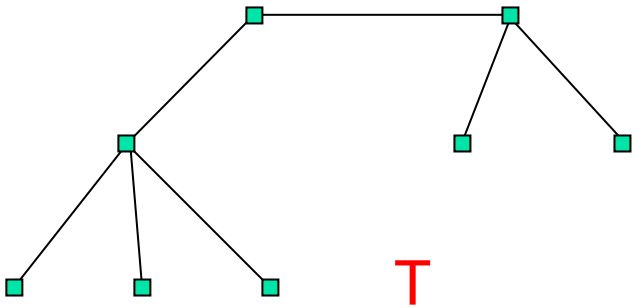
Exercise 1: Not a constant factor approximation for any constant c

Exercise 2*: Is an O(log n) approximation

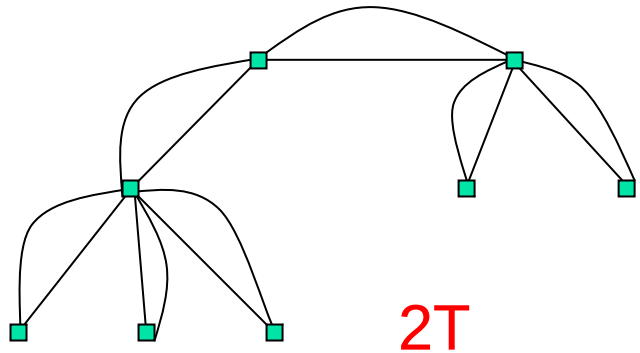
MST based algorithm

1. Compute an MST of G, say T
2. Obtain an Eulerian graph H=2T by *doubling* edges of T
3. An Eulerian tour of 2T gives a tour in G

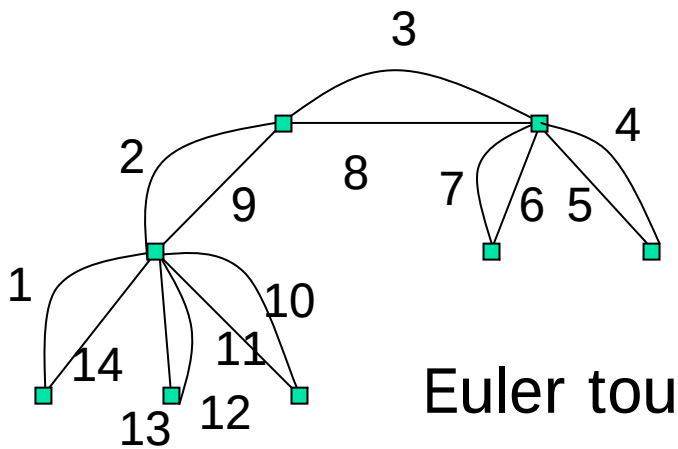
MST based algorithm



T



2T



Euler tour

MST based algorithm

Claim: MST heuristic is a 2-approximation algorithm

$c(T) = \sum_{e \in E(T)} c(e) \leq OPT$

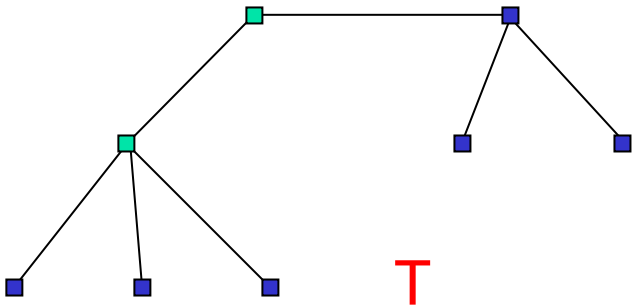
$c(2T) = 2\, c(T) \leq 2\, OPT$

Christofides heuristic

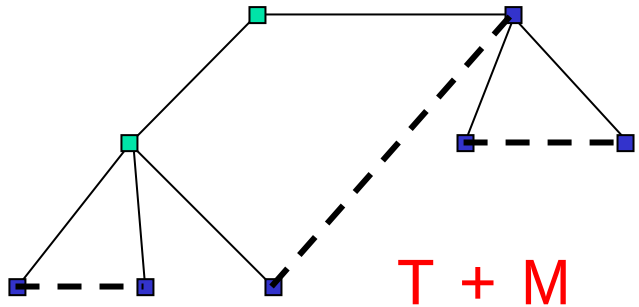
Can we convert T into an Eulerian graph?

1. Compute an MST of G, say T. Let S be the vertices of *odd* degree in T (Note: |S| is even)
2. Find a minimum cost matching M on S in G
3. Add M to T to obtain Eulerian graph H
4. Compute an Eulerian tour of H

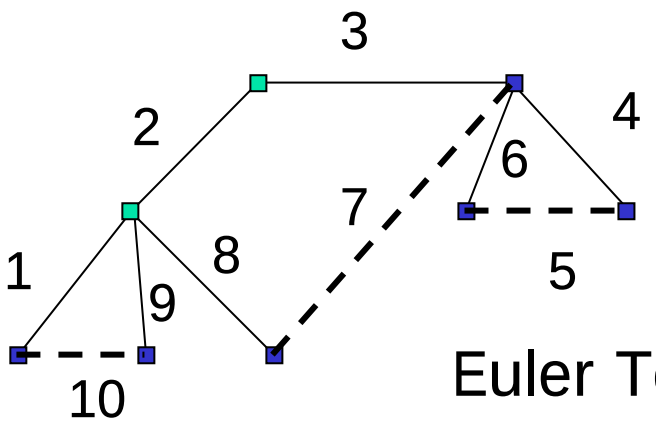
Christofides heuristic



T

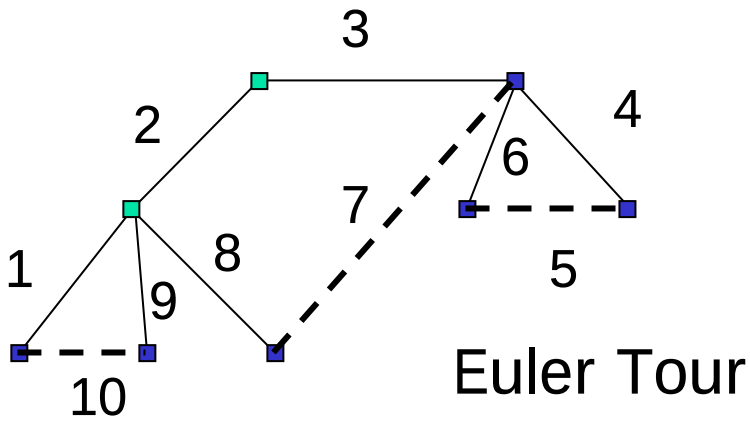


T + M

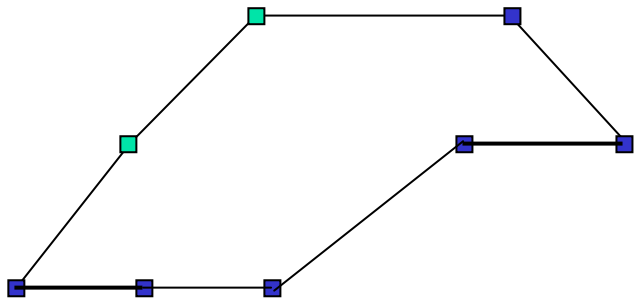


Euler Tour

Christofides heuristic



Euler Tour



Shortcut

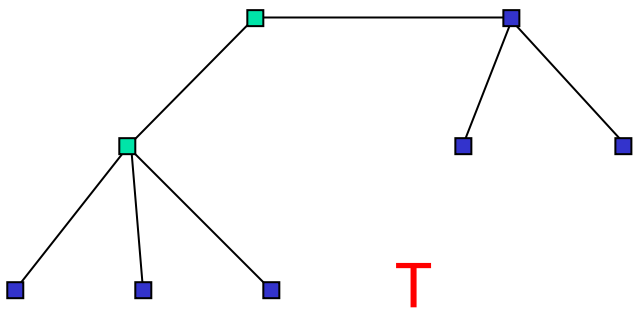
3/2 approx for Metric-TSP

Lemma: $c(M) \leq OPT/2$

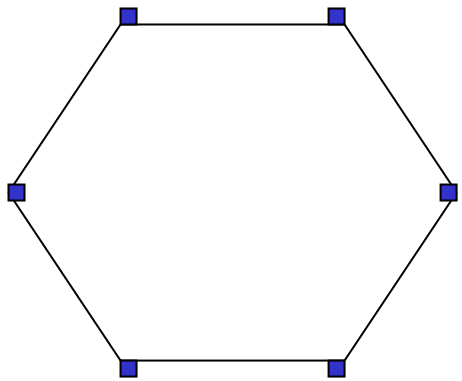
$c(H) = c(T) + c(M) \leq OPT + OPT/2 = 3OPT/2$

Proof of Lemma

A tour in G of cost OPT implies a tour on S of cost at most OPT (why?)



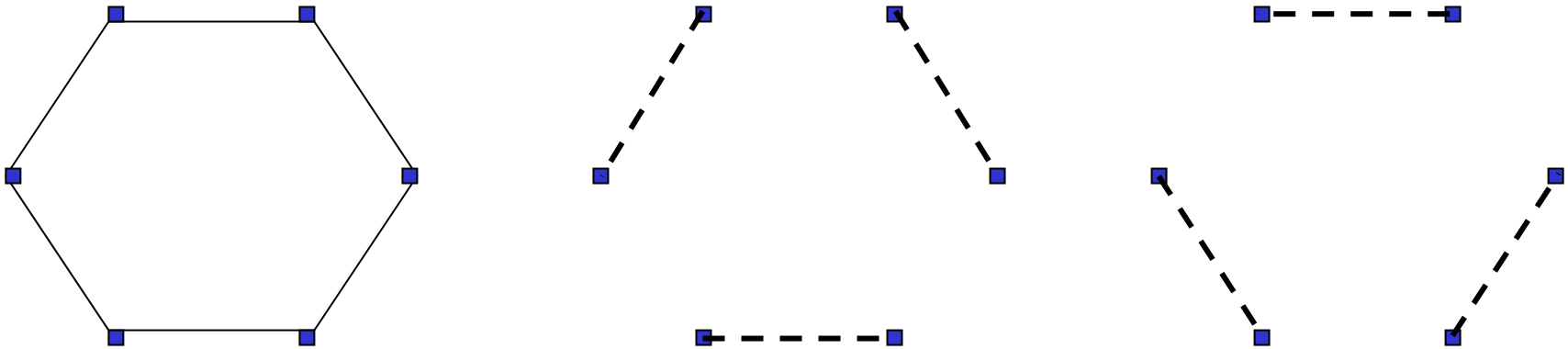
T



Tour on S

Proof of Lemma

A tour in G of cost OPT implies a tour on S of cost at most OPT (why?)

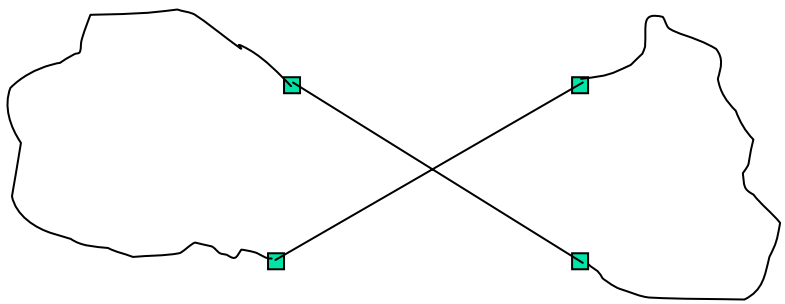


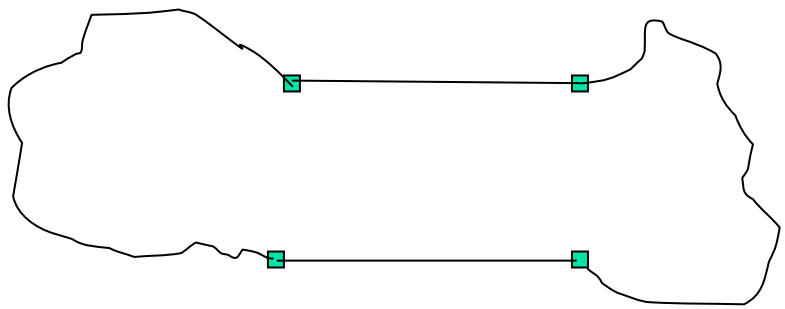
$c(M_1) + c(M_2) \leq OPT$

Practice and local search

Local search heuristics perform extremely well

2-Opt: apply following step if it improves tour





Research Problem

Christofides heuristic: 1976

No improvement in 30 years!

Open Problem: Improve 3/2 for Metric-TSP

Candidate: Held-Karp linear program relaxation is conjectured to give a ratio of 4/3 – important open problem

TSP in Directed Graphs

G = (V, A), A : arcs
c: A $\rightarrow \mathcal{R}^+$ non-negative arc weights

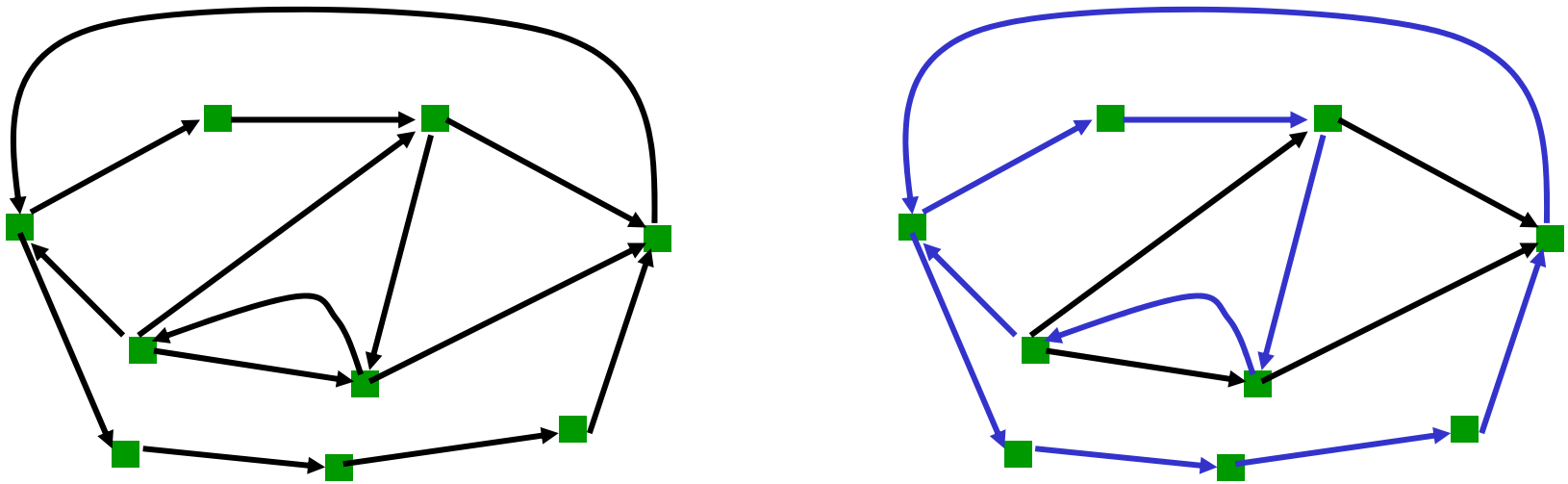
TSP: find min-cost directed Hamiltonian cycle

ATSP: asymmetric TSP (allow vertex to be visited multiple times)

Equivalent to assuming c satisfies
c(u,v) + c(v, w) $\geq$ c(u, w) for all u,v,w

Note: c(u,v) might not equal c(v, u) (asymmetry)

Example



Heuristic ideas?

Metric-TSP heuristics relied on symmetry strongly

Directed cycles and their use

Wlog assume that G is a complete directed graph with $c(uv) + c(vw) \geq c(uw)$ for all u,v,w

Given $S \subseteq V$, $G[S]$ induced graph on S

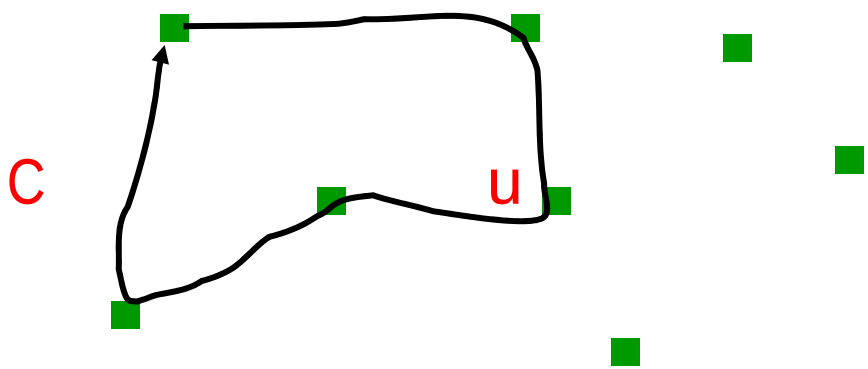
$OPT(G)$: cost of tour in G

Observation: $OPT(G[S]) \leq OPT(G)$ for any $S \subseteq V$

Directed cycles and their use

Consider a directed cycle C on S
Pick an arbitrary vertex $u \in S$

$V' = (V - S) \cup \{u\}$

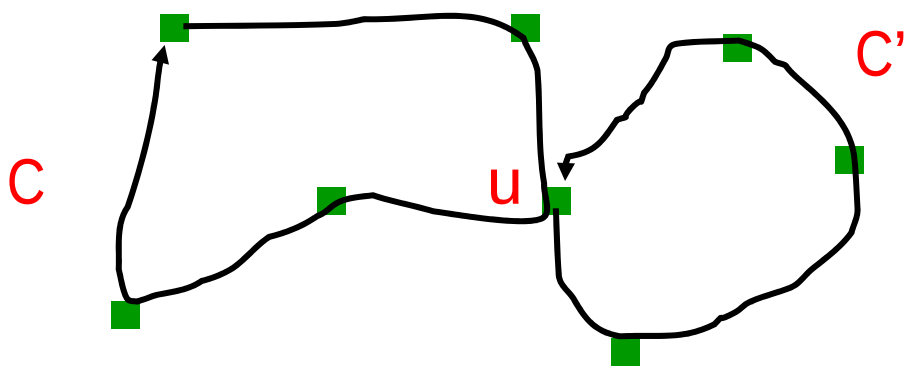


Directed cycles and their use

Consider a directed cycle C on S
Pick an arbitrary vertex $u \in S$

$V' = (V - S) \cup \{u\}$

Find a solution C' in G[V']
Can extend solution to G using $C' \cup C$

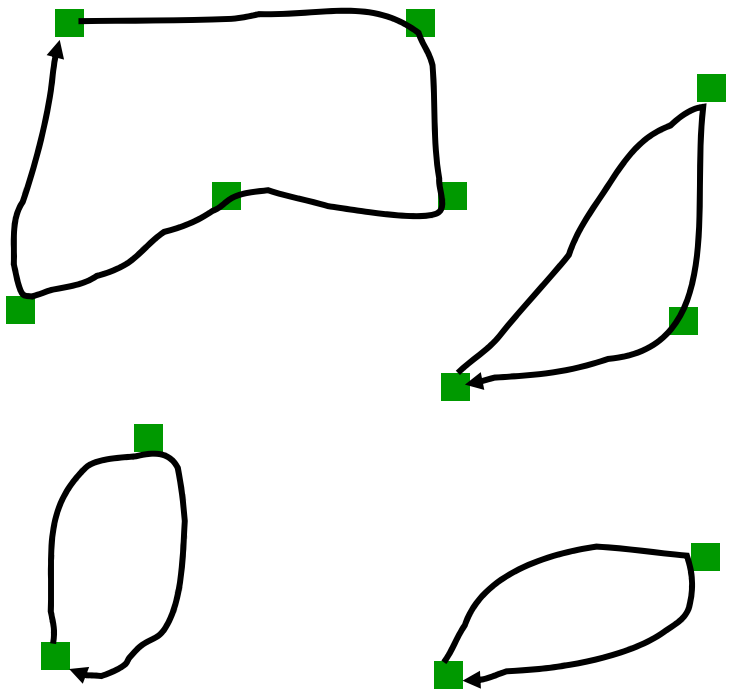


Cycle cover

Cycle cover of G

collection of cycles C_1, C_2, …, C_k

such that each vertex is in exactly one cycle



Min-cost cycle cover

cycle cover cost: sum of costs of edges in cover

Claim: a minimum cost cycle cover in a directed graph can be computed in polynomial time

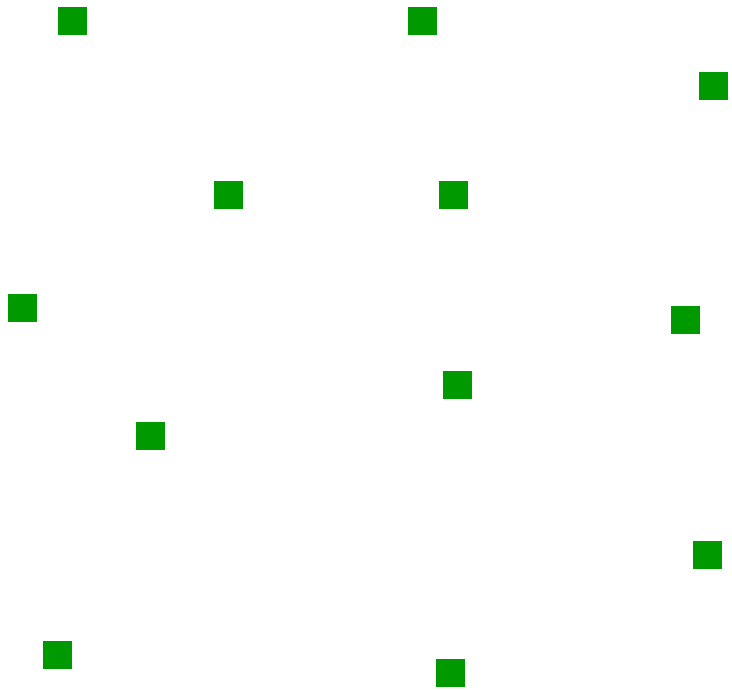
See Hw 0, Prob 5

Cycle Shrinking Algorithm

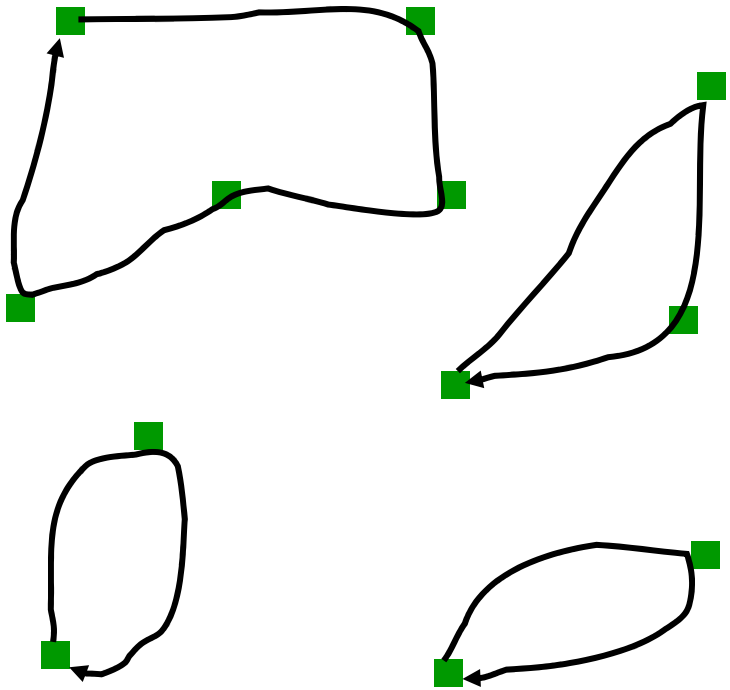
[Frieze-Galbiati-Maffioli'82] $\log_2 n$ approx

1. Find a *minimum cost* cycle cover (poly-time computable)
2. Pick a proxy node for each cycle
3. *Recursively* solve problem on proxies – extend using cycles

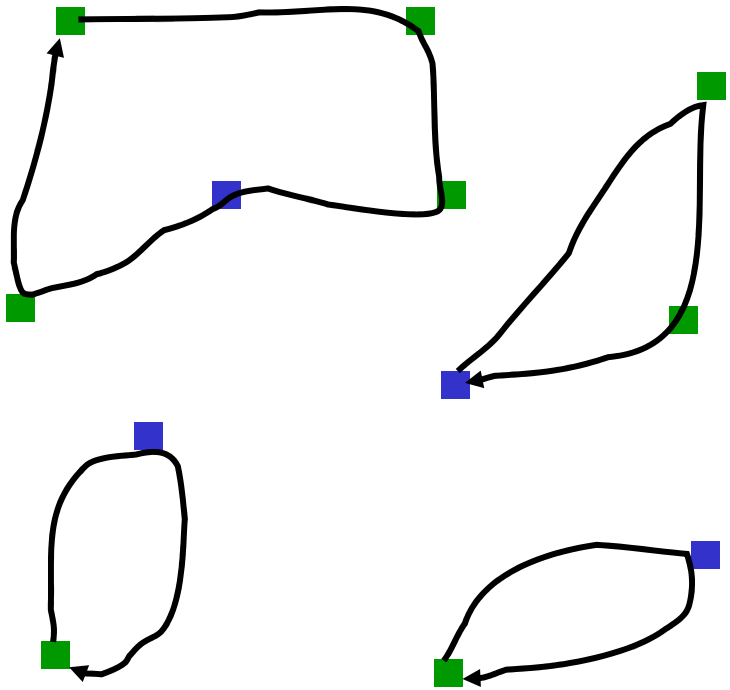
Example



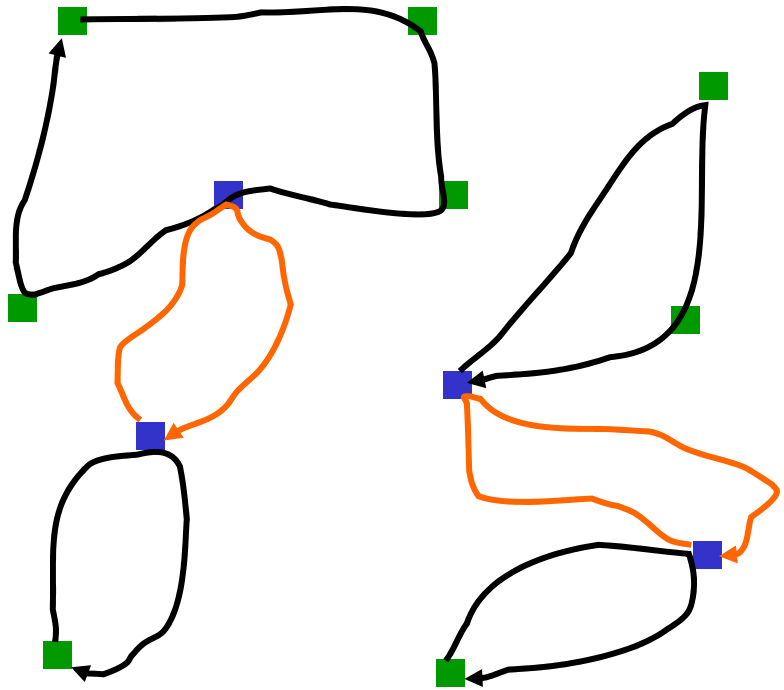
Example



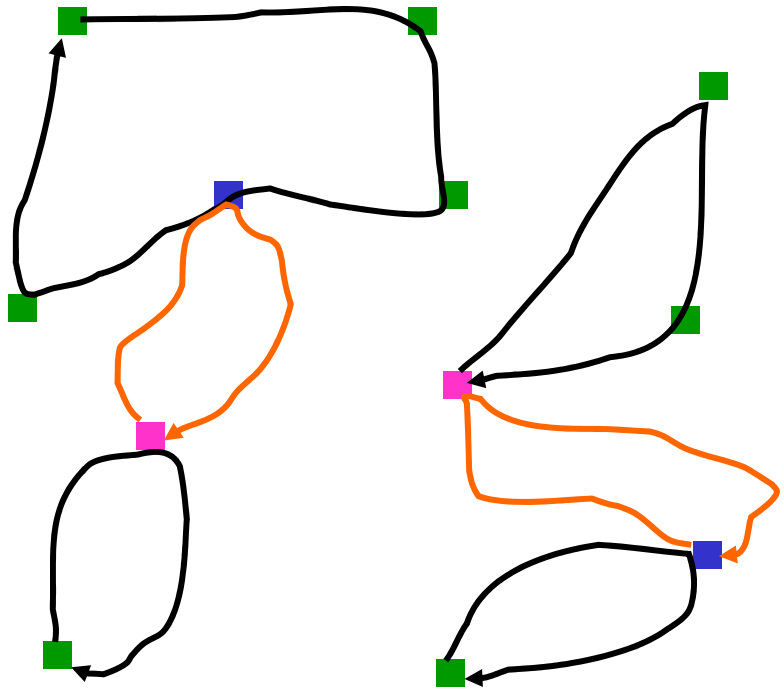
Example



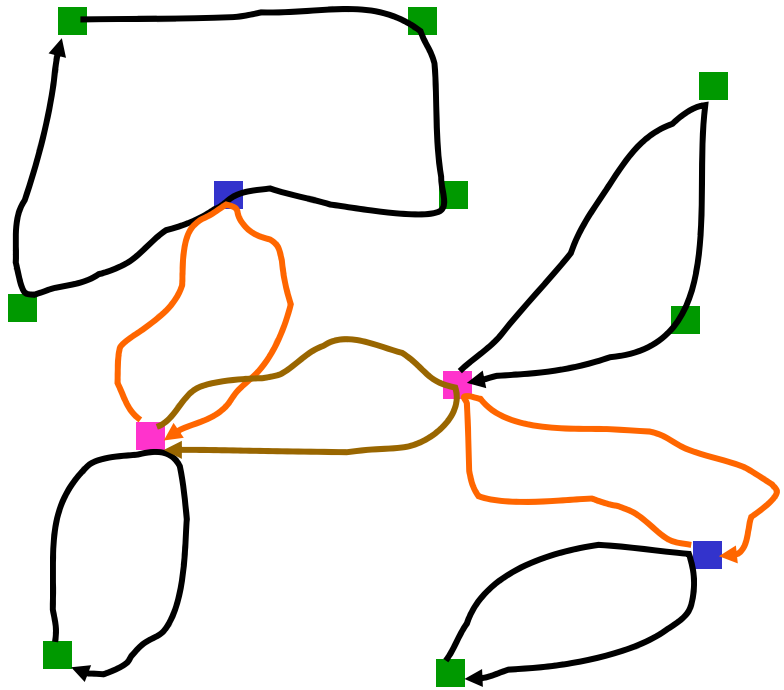
Example



Example



Example



Analysis

$V_0 = V$

V_i : set of proxy nodes after iteration i

$G_i = G[V_i]$

x_i : cost of cycle cover in G_i

Claim: $x_i \leq OPT$ (why?)

Analysis

x_i : cost of cycle cover in G_i

Claim: $x_i \leq$ OPT (why?)

Total cost = $x_1 + x_2 + \ldots + x_k$ where k is number of iterations

$\leq$ OPT log n

Why is $k \leq \log n$?

Analysis

Conclusion: log n approximation for ATSP

Running time:

each iteration computes a min cost cycle cover

log n iterations
$|V_i| \leq |V|/2^i$

time dominated by first iteration: $O(n^{2.5})$

Can be reduced to $O(n^2)$ with a small loss in approximation ratio

ATSP

Best known ratio: 0.842 log n

Open problem: Is there a constant factor approximation for ATSP? (open for 25 years!)

Formal definition of NPO problems

P, NP: language/decision classes

NPO: NP Optimization problems, function class

Optimization problem

Π is an optimization problem

- Π is either a *min* or *max* type problem
- Instances I of Π are a subset of Σ^*
- $|I|$ size of instance
- for each I there are feasible solutions $\mathcal{S}(I)$
- for each I and solution $S \in \mathcal{S}(I)$ there is a real/rational number $val(S, I)$
- Goal: given I, find $OPT(I) = \min_{S \in \mathcal{S}(I)} val(S, I)$

NPO: NP Optimization problem

Π is an NPO problem if

- Given $x \in \Sigma^*$, can check if x is an instance of Π in $\text{poly}(|x|)$ time
- for each I, and $S \in \mathcal{S}(I)$, $|S|$ is $\text{poly}(|I|)$
- there exists a poly-time decision procedure that for each I and $x \in \Sigma^*$, decides if $x \in \mathcal{S}(I)$
- $\text{val}(I, S)$ is a poly-time computable function

NPO and NP

Π in NPO, minimization problem

For a rational number B
define $L(\Pi, B) = \{ I \mid OPT(I) \leq B \}$

Claim: $L(\Pi, B)$ is in NP

$L(\Pi, B)$: decision version of Π

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

  val $(\mathcal{A}(I)) \leq \alpha$ val (OPT(I))

(Note: $\alpha \geq 1$)

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

Example: $\alpha(|I|) = \log n$

Maximization problems

Maximization 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

  val $(\mathcal{A}(I)) \geq \alpha$ val (OPT(I))

(Note: $\alpha \leq 1$)

Very often people use $1/\alpha$ (≥ 1) as approximation ratio

Relative vs Additive approximation

Approximation ratio defined as a relative measure

Why not additive?

α–additive approximation implies

for all I, val($\mathcal{A}$(I)) $\leq$ OPT(I) + α

For most NPO problems no additive approximation ratio possible because OPT(I) has scaling property

Scaling property

Example: Metric-TSP

Take instance I, create instance I' with all edge costs increased by a factor of β

for each $S \in \mathcal{S}(I) = \mathcal{S}(I')$, $\quad val(S, I') = \beta\, val(S, I)$

$OPT(I') = \beta\, OPT(I)$

Suppose there exists an *additive* α approximation for Metric-TSP then by choosing β sufficiently large, can obtain an *exact* algorithm!

Scaling property

Given I, run additive approx alg on I' to get a solution $S \in \mathcal{S}(I')$, return S for I

$val(S, I') \leq OPT(I') + \alpha$

$val(S, I) = val(S, I')/\beta = OPT(I')/\beta + \alpha/\beta$

$= OPT(I) + \alpha/\beta$

Choose β s.t $\alpha/\beta < 1$
for integer data, $val(S,I) < OPT(I) + 1 \Rightarrow val(S,I) = OPT(I)$

Some simple additive approximations

Planar-graph coloring:

Fact: NP-complete to decide if a planar graph adimits 3-coloring

Fact: can always color using 4 colors

Edge coloring:

Vizing's thm: edge coloring number is either $\Delta(G)$ or $\Delta(G) + 1$

Fact: NP-complete to decide!

Hardness of approximation

Given optimization problem Π (say minimization)

Q: What is the smallest α such that Π has an approximation ratio of α?

$\alpha^*(\Pi)$: approximability threshold of Π

$\alpha^*(\Pi) = 1$ implies Π is polynomial time solvable

Hardness of Approximation

P = NP $\Rightarrow$ *all* NPO problems have exact algorithms (Why?)

P $\neq$ NP $\Rightarrow$ *many* NPO problems have $\alpha^*(\Pi) > 1$

(Which ones?)

Approximation algorithms: *upper bounds* on $\alpha^*(\Pi)$

Hardness of approximation: *lower bounds on* $\alpha^*(\Pi)$

Proving hardness of approximation

Unless P = NP, $\alpha(\Pi)$ > ??

Direct: from an NP-Complete problem

Indirect: via gap reductions

Example: k-center

(metric) k-center problem:

given undirected graph $G = (V, E)$ and integer k

choose k centers/vertices in V

Goal: minimize maximum distance to a center

$$\min_{S \subset V, |S| = k} \max_{v \in V} \text{dist}_G(v, S)$$

$$\text{dist}_G(v, S) = \min_{u \in S} \text{dist}_G(u,v)$$

Example: k-center

k-center problem:

given undirected graph $G = (V, E)$ and integer k

choose k centers/vertices in V

Goal: minimize maximum distance to a center

$$\min_{S \subset V, |S| = k} \max_{v \in V} \text{dist}_G(v, S)$$

Theorem: Unless P=NP there is no $2-\varepsilon$ approximation for k-center for any $\varepsilon > 0$

Theorem: There is a 2-approximation for k-center

k-center hardness of approximation

Given undirected graph $G=(V,E)$

Dominating set: $S \subseteq V$ s.t for all $v \in V$, either $v \in S$ or v is adjacent to some u in S

Dominating Set Problem:

Given $G = (V, E)$ is there a dominating set in G of size k?

NP-Complete

k-center hardness of approximation

Dominating Set: decision version

Given $G = (V, E)$ is there a dominating set in G of size k?

Use DS to prove hardness for k-center

k-center hardness of approximation

Dominating Set:

Given G = (V, E) is there a dominating set in G of size k?

Given instance of DS instance I create instance I' of k-center – graph and k remain the same

k-center hardness of approximation

If I has DS of size k (that is I is in the language) then OPT(I') = 1 (why?)

If I does not have a DS of size k (that is I is not in the language) then OPT(I') $\geq$ 2 (why?)

Therefore a 2-ε approximation for k-center can be used to solve DS

$\Rightarrow$ unless P=NP, α^*(k-center) $\geq$ 2-ε

Algorithm for k-center

Given $G = (V, E)$

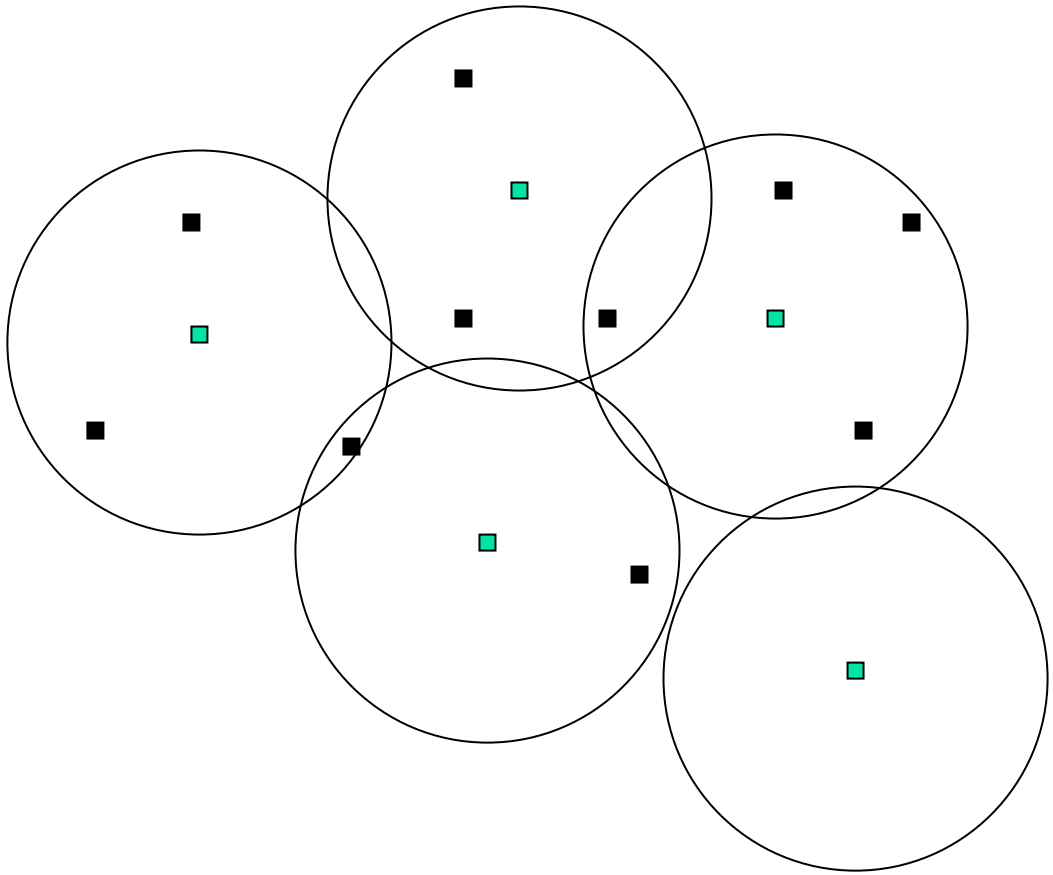
integer k

Pick k centers from V so that the maximum distance of any vertex to a center is minimized

Let R^* be the optimal distance

Example

Let R^* be the optimal distance



2-approx for k-center

Assume we know R^*

Initialize all vertices to be *uncovered*

for $i = 1$ to k do

  if no uncovered vertex, BREAK

  pick an uncovered vertex u as a *center*

  set all vertices within $2R^*$ from u as covered

endif

Output set of centers

2-approx for k-center

Assume we know R^*

Initialize all vertices to be *uncovered*

for $i = 1$ to k do

  if no uncovered vertex, BREAK

  pick an uncovered vertex u as a *center*

  set all vertices within $2R^*$ from u as covered

end if

Output set of centers

Claim: All vertices covered by chosen centers

Another 2-approx for k-center

$S = \emptyset$

for $i = 1$ to k do

  let u be vertex in G farthest from S

  add u to S

Output S

Also known as Gonzalez's algorithm

Exercises

Prove that the k-center algorithms yield a 2-approximation

Compare running times of algorithms

Which one would you prefer in practice? Why?