Approximation Algorithms

Ruben Hoeksma (University of Bremen)

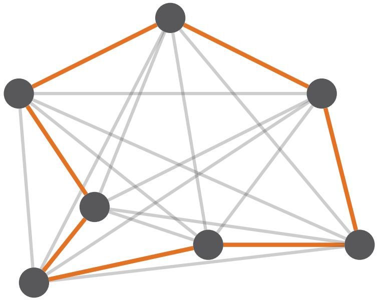
April 11, 2019

The Traveling salesperson problem (TSP)

The Traveling Salesperson Problem (TSP)

Input: complete graph $G = (V, E)$, distances $d : E \to \mathbb{R}_+$

Task: find a Hamiltonian cycle C in G minimizing $d(C) := \sum_{e \in C} d(e)$

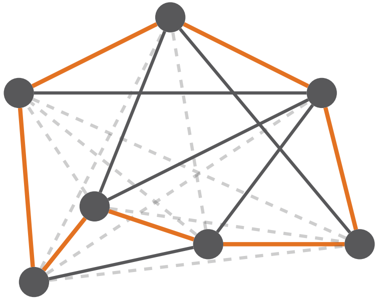


Hardness

Theorem

There is no α-approximation for TSP for any α, unless $P = NP$.

Proof Deciding whether graph has a Hamiltonian cycle is *NP*-hard.



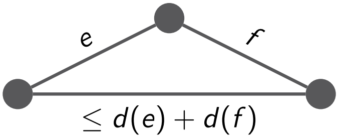
Given graph $G' = (V, E')$ define complete graph $G = (V, E)$ with $d(e) = 0$ if $e \in E'$ and $d(e) = 1$ if $e \notin E'$.

$\textsc{Yes}$ instance: $\text{OPT} = 0$ $\textsc{No}$ instance: $\text{OPT} \geq 1$ $\square$

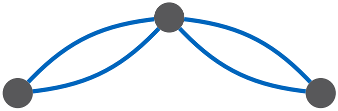
Metric TSP

Input: complete graph $G = (V, E)$, distances $d : E \to \mathbb{R}_+$, with $d(u, w) \leq d(u, v) + d(v, w)$ for all $u, v, w \in V$

Task: find a Hamiltonian cycle C in G minimizing $d(C) := \sum_{e \in C} d(e)$



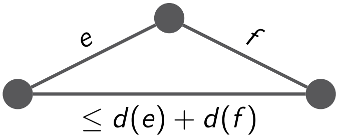
Shortcuts



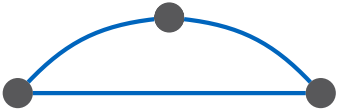
Metric TSP

Input: complete graph $G = (V, E)$, distances $d : E \to \mathbb{R}_+$, with $d(u, w) \leq d(u, v) + d(v, w)$ for all $u, v, w \in V$

Task: find a Hamiltonian cycle C in G minimizing $d(C) := \sum_{e \in C} d(e)$



Shortcuts



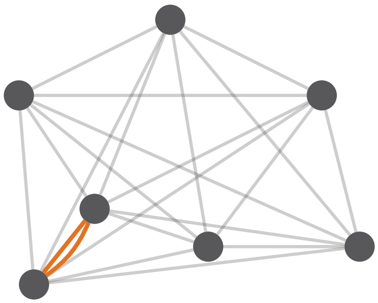
A greedy algorithm

Algorithm: Greedy

1. $C = 2 \times$ shortest edge, $\underset{\{u,v\} \in E}{\operatorname{argmin}} d(u, v)$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u, v)$ is minimal.
4. Add v to S, and $\{u, v\}$ and $\{v, u'\}$ to C, where u' was u's successor.



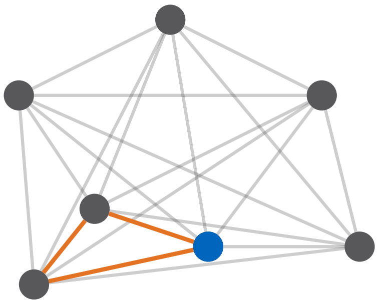
A greedy algorithm

Algorithm: Greedy

1. $C = 2 \times$ shortest edge, $\underset{\{u,v\} \in E}{\operatorname{argmin}} d(u, v)$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u, v)$ is minimal.
4. Add v to S, and $\{u, v\}$ and $\{v, u'\}$ to C, where u' was u's successor.



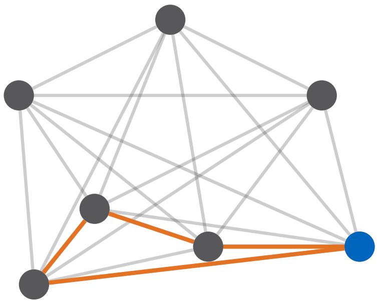
A greedy algorithm

Algorithm: Greedy

1. $C = 2 \times$ shortest edge, $\operatorname{argmin}_{\{u,v\} \in E} d(u,v)$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u,v)$ is minimal.
4. Add v to S, and $\{u, v\}$ and $\{v, u'\}$ to C, where u' was u's successor.

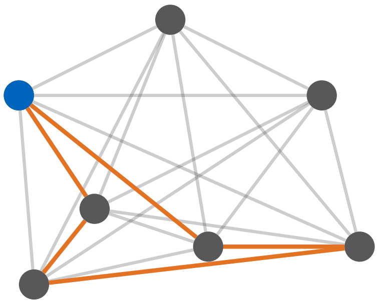


Algorithm: Greedy

1. $C = 2 \times$ shortest edge, $\operatorname*{argmin}_{\{u,v\} \in E} d(u,v)$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u, v)$ is minimal.
4. Add v to S, and $\{u, v\}$ and $\{v, u'\}$ to C, where u' was u's successor.

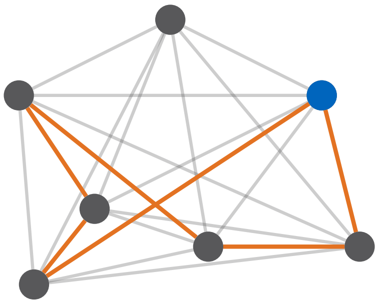


Algorithm: Greedy

1. $C = 2\times$ shortest edge, $\operatorname{argmin}_{\{u,v\}\in E} d(u,v)$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u,v)$ is minimal.
4. Add v to S, and $\{u, v\}$ and $\{v, u'\}$ to C, where u' was u's successor.



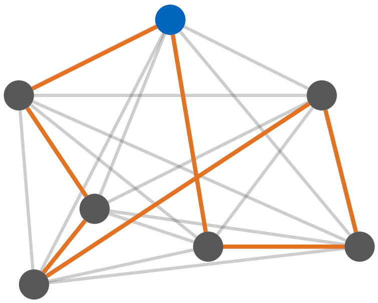
A greedy algorithm

Algorithm: Greedy

1. $C = 2\times$ shortest edge, $\underset{\{u,v\}\in E}{\operatorname{argmin}}\, d(u,v)$.
2. $S = \{u,v\}$; $\bar{S} = V \setminus \{u,v\}$.

While $S \neq V$:

3. Find $u \in S$ and $v \in \bar{S}$, s.t. $d(u,v)$ is minimal.
4. Add v to S, and $\{u,v\}$ and $\{v,u'\}$ to C, where u' was u's successor.

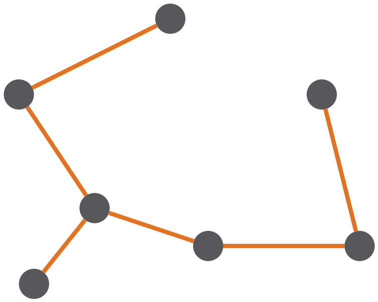


Minimum spanning tree

Minimum spanning tree (MST)

A subset $T \subseteq E$ is a tree if it

1. connects all vertices
2. contains no cycles
3. has minimum total length



Prim's algorithm:

1 $T = \left\{ \operatorname*{argmin}_{\{u,v\} \in E} d(u,v) \right\}$.

2 $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3 $e = \operatorname*{argmin}_{u \in S, v \in \bar{S}} d(u,v)$.

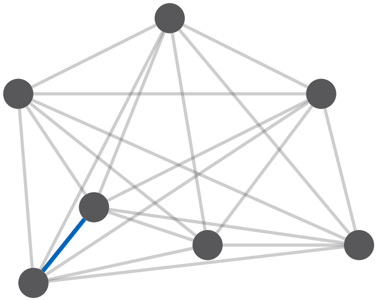
4 $T = T + e$.

5 $S = S + v$.

6 $\bar{S} = \bar{S} - v$.

Theorem

Prim's algorithm computes a MST.



Prim's algorithm:

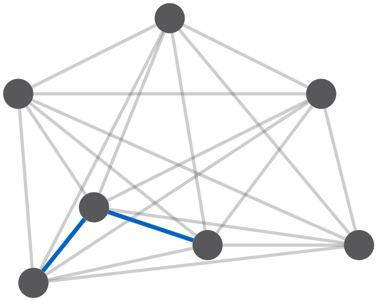
1. $T = \left\{ \operatorname*{argmin}_{\{u,v\} \in E} d(u, v) \right\}$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. $e = \operatorname*{argmin}_{u \in S, v \in \bar{S}} d(u, v)$.
4. $T = T + e$.
5. $S = S + v$.
6. $\bar{S} = \bar{S} - v$.

Theorem

Prim's algorithm computes a MST.



Prim's algorithm:

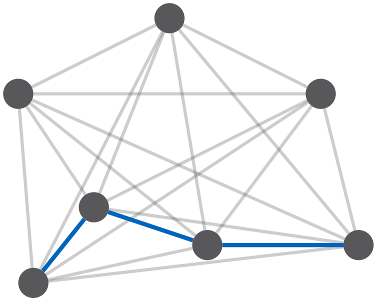
1. $T = \left\{ \operatorname*{argmin}_{\{u,v\} \in E} d(u,v) \right\}$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. $e = \operatorname*{argmin}_{u \in S, v \in \bar{S}} d(u,v)$.
4. $T = T + e$.
5. $S = S + v$.
6. $\bar{S} = \bar{S} - v$.

Theorem

Prim's algorithm computes a MST.



Prim's algorithm

Prim's algorithm:

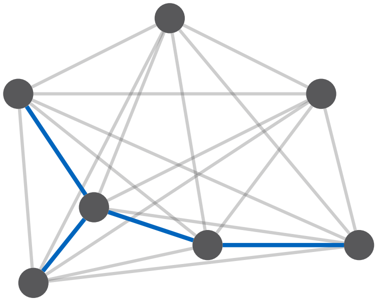
1. $T = \left\{ \operatorname{argmin}_{\{u,v\} \in E} d(u, v) \right\}$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. $e = \operatorname{argmin}_{u \in S, v \in \bar{S}} d(u, v)$.
4. $T = T + e$.
5. $S = S + v$.
6. $\bar{S} = \bar{S} - v$.

Theorem

Prim's algorithm computes a MST.



Prim's algorithm:

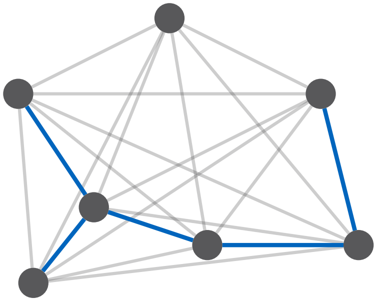
1. $T = \left\{ \underset{\{u,v\} \in E}{\operatorname{argmin}} \, d(u,v) \right\}$.
2. $S = \{u, v\}$; $\bar{S} = V \setminus \{u, v\}$.

While $S \neq V$:

3. $e = \underset{u \in S, v \in \bar{S}}{\operatorname{argmin}} \, d(u,v)$.
4. $T = T + e$.
5. $S = S + v$.
6. $\bar{S} = \bar{S} - v$.

Theorem

Prim's algorithm computes a MST.



Prim's algorithm:

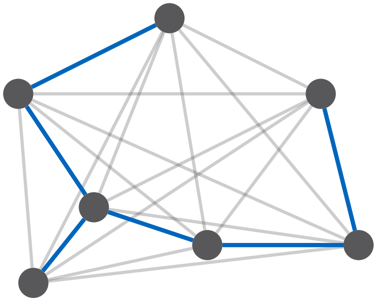
1. $T = \left\{ \underset{\{u,v\} \in E}{\operatorname{argmin}} d(u, v) \right\}.$
2. $S = \{u, v\};\ \bar{S} = V \setminus \{u, v\}.$

While $S \neq V$:

3. $e = \underset{u \in S, v \in \bar{S}}{\operatorname{argmin}} d(u, v).$
4. $T = T + e.$
5. $S = S + v.$
6. $\bar{S} = \bar{S} - v.$

Theorem

Prim's algorithm computes a MST.



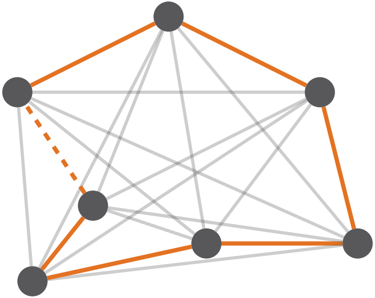
The tree lower bound

Lemma

Let C be a Hamiltonian cycle in G and T be a minimum spanning tree in G. Then $d(T) \leq d(C)$.

Proof. Let $e \in C$. Then $C \setminus \{e\}$ is a spanning tree in G. Hence

$$d(T) \leq d(C \setminus \{e\}) \leq d(C).$$



□

The Double-tree algorithm

Algorithm:

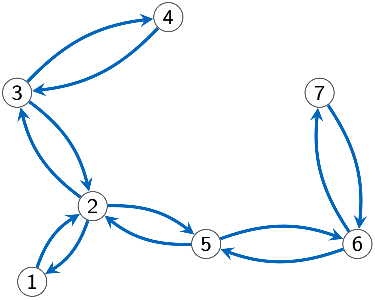
1. Compute MST T.
2. Let $H = (V,\ T \dot\cup T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.

Theorem

The Double-tree algorithm is a 2-approximation for metric TSP.

Theorem

The Greedy algorithm for metric TSP is a 2-approximation.



The Double-tree algorithm

Algorithm:

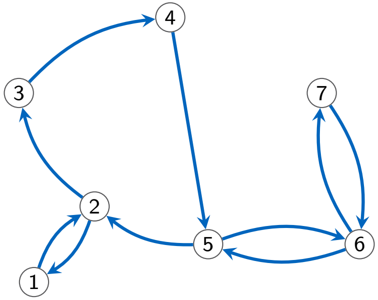
1. Compute MST T.
2. Let $H = (V,\ T \dot{\cup} T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.

Theorem

The Double-tree algorithm is a 2-approximation for metric TSP.

Theorem

The Greedy algorithm for metric TSP is a 2-approximation.



The Double-tree algorithm

Algorithm:

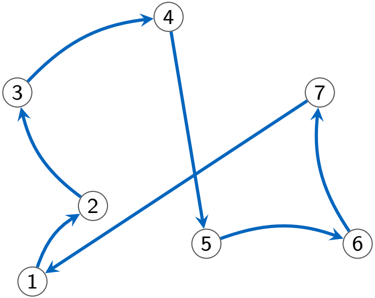
1. Compute MST T.
2. Let $H = (V,\ T \dot\cup T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.

Theorem

The Double-tree algorithm is a 2-approximation for metric TSP.

Theorem

The Greedy algorithm for metric TSP is a 2-approximation.



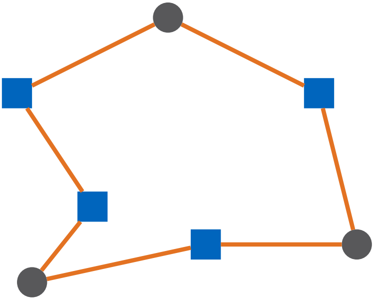
The Matching lower bound

Lemma

Let C be a HC in G and let $U \subseteq V$ with $|U|$ even. Let M be a minimum weight perfect matching on U. Then $d(M) \leq \frac{1}{2}d(C)$.

Proof. Shortcut C to cycle C' on U. $\qquad d(C') \leq d(C)$

C' contains two disjoint perfect matchings on U. $\qquad 2d(M) \leq d(C')$



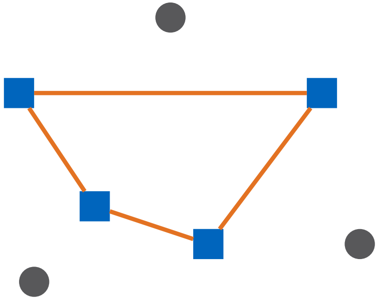
The Matching lower bound

Lemma

Let C be a HC in G and let $U \subseteq V$ with $|U|$ even. Let M be a minimum weight perfect matching on U. Then $d(M) \leq \frac{1}{2} d(C)$.

Proof. Shortcut C to cycle C' on U. $\quad d(C') \leq d(C)$

C' contains two disjoint perfect matchings on U. $\quad 2d(M) \leq d(C')$

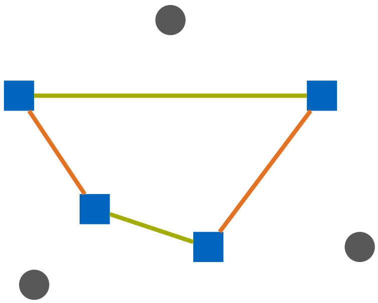


The Matching lower bound

Lemma

Let C be a HC in G and let $U \subseteq V$ with $|U|$ even. Let M be a minimum weight perfect matching on U. Then $d(M) \leq \frac{1}{2}d(C)$.

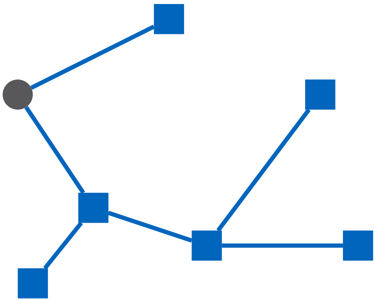
Proof. Shortcut C to cycle C' on U. $d(C') \leq d(C)$
C' contains two disjoint perfect matchings on U. $2d(M) \leq d(C')$



Christofides' algorithm

Algorithm:

1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V, T \dot{\cup} M)$.
4. Shortcut C' to HC C.



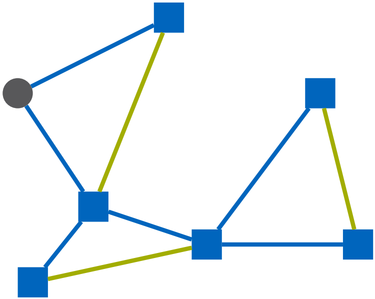
Theorem

Christofides' algorithm is a 3/2-approximation for metric TSP.

Christofides' algorithm

Algorithm:

1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \dot{\cup} M)$.
4. Shortcut C' to HC C.



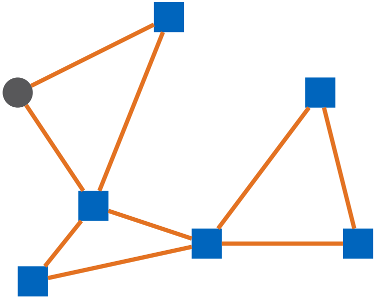
Theorem

Christofides' algorithm is a 3/2-approximation for metric TSP.

Christofides' algorithm

Algorithm:

1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \,\dot\cup\, M)$.
4. Shortcut C' to HC C.



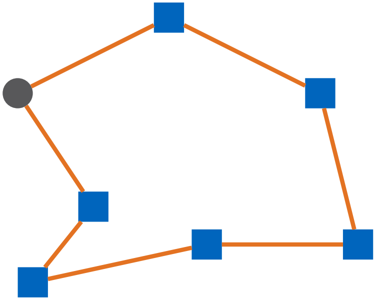
Theorem

Christofides' algorithm is a 3/2-approximation for metric TSP.

Christofides' algorithm

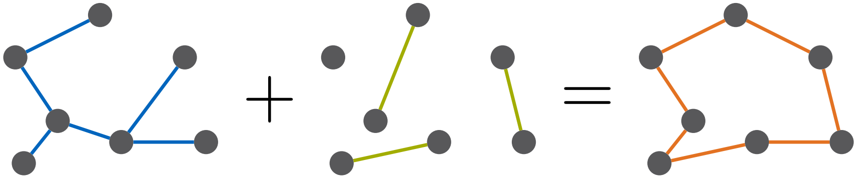
Algorithm:

1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V, T \dot\cup M)$.
4. Shortcut C' to HC C.



Theorem

Christofides' algorithm is a 3/2-approximation for metric TSP.



3/2-approximation for metric TSP