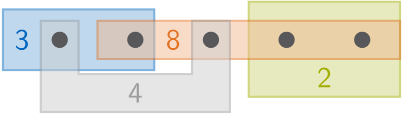
Lecture: Approximation Algorithms

Jannik Matuschke

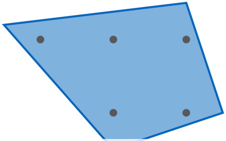
October 24, 2018

Previously ...

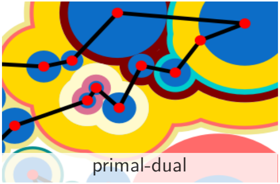
Example: SET COVER



Many techniques:



LP rounding



primal-dual

greedy

to be continued

Randomized LP Rounding

Randomized rounding

Idea: Include set S with probability $x(S)$.

Randomized rounding

Idea: Include set S with probability $x(S)$.

Expected cost: $$\mathbb{E}[\sum_{S \in \mathcal{F}'} w(S)] = \sum_{S \in \mathcal{F}} w(S) x(S) = Z^*$$

Randomized rounding

Idea: Include set S with probability $x(S)$.

Expected cost: $$\mathbb{E}[\sum_{S \in \mathcal{F}'} w(S)] = \sum_{S \in \mathcal{F}} w(S)x(S) = Z^*$$

but probability to produce set cover can be very low!

Randomized rounding

Idea: For each S, throw $c \cdot \ln(n)$ coins, each showing heads with probability $x(S)$. If at least one of them shows heads, include S.

$n := |E|, \; c > 1$

Randomized rounding

Idea: For each S, throw $c \cdot \ln(n)$ coins, each showing heads with probability $x(S)$. If at least one of them shows heads, include S.

$$n := |E|, \; c > 1$$

Probability that element e is not covered:

$$\prod_{S: e \in S} (1 - x(S))^{c \ln(n)} \;\leq\; \exp\left(-c \ln(n) \sum_{S: e \in S} x(S)\right) \;\leq\; \frac{1}{n^c}$$

Randomized rounding

Idea: For each S, throw $c \cdot \ln(n)$ coins, each showing heads with probability $x(S)$. If at least one of them shows heads, include S.

$n := |E|, \ c > 1$

Probability that element e is not covered:

$$\prod_{S:e\in S} (1 - x(S))^{c\ln(n)} \ \leq \ \exp\left(-c\ln(n)\sum_{S:e\in S} x(S)\right) \ \leq \ \frac{1}{n^c}$$

Probability to generate a set cover:

$$1 - \Pr\left[\exists \text{ uncovered } e\right] \ \geq \ 1 - \sum_{e\in E}\frac{1}{n^c} \ = \ 1 - \frac{1}{n^{c-1}}$$

Randomized rounding

Idea: For each S, throw $c \cdot \ln(n)$ coins, each showing heads with probability $x(S)$. If at least one of them shows heads, include S.

$$n := |E|, \ c > 1$$

Probability that element e is not covered:

$$\prod_{S:e\in S} (1 - x(S))^{c\ln(n)} \ \leq \ \exp\left(-c\ln(n) \sum_{S:e\in S} x(S)\right) \ \leq \ \frac{1}{n^c}$$

Probability to generate a set cover:

$$1 - \Pr\left[\exists \text{ uncovered } e\right] \ \geq \ 1 - \sum_{e\in E} \frac{1}{n^c} \ = \ 1 - \frac{1}{n^{c-1}}$$

We say the algorithm outputs a set cover with high probability.

Randomized rounding

Idea: For each S, throw $c \cdot \ln(n)$ coins, each showing heads with probability $x(S)$. If at least one of them shows heads, include S.

$n := |E|, \ c > 1$

Theorem 2.1

The Randomized Rounding Algorithm computes a set cover w.h.p. If it succeeds, its expected cost is at most $2c\ln(n)Z^*$.

Hardness of Approximation

Approximation hardness

Theorem 2.2

There is a $c > 0$ such that there is no $c \ln(n)$-approximation algorithm for SET COVER, unless $P = NP$.

Approximation hardness

Theorem 2.2

There is a $c > 0$ such that there is no $c \ln(n)$-approximation algorithm for Set Cover, unless $P = NP$.

Theorem 2.3

There no α-approximation algorithm for Vertex Cover for any $\alpha < 2$, unless the Unique Games Conjecture is false or $P = NP$.

Set Cover: Conclusion & Outlook

Approximation techniques

- LP rounding (deterministic/randomized)
- primal-dual method
- greedy algorithm

Set Cover: Conclusion & Outlook

Approximation techniques

- LP rounding (deterministic/randomized)
- primal-dual method
- greedy algorithm

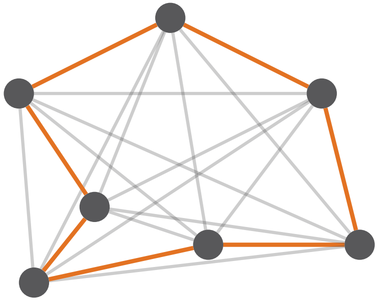
+ combinatorial lower bounds
+ local search
+ rounding data & dynamic programs

Combinatorial Lower Bounds for the Traveling Salesman Problem

Traveling Salesman 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

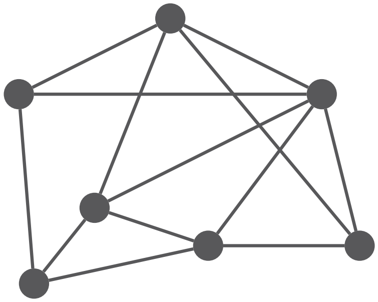
There is no α-approximation for TSP for any α, unless $P = NP$.

Hardness

Theorem

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

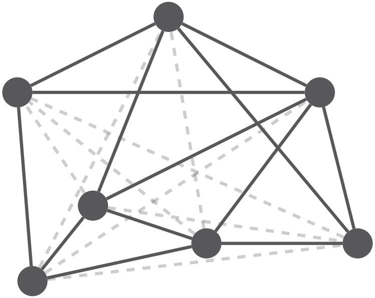
Proof Deciding whether graph has a Hamiltonian cycle is *NP*-hard.



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.



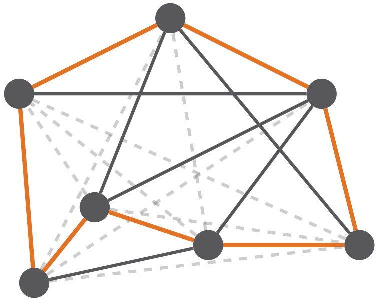
Given graph $G' = (V, E')$ define complete graph $G = (V, E)$ with $d(e) = 0$ if $e \in E'$ and $d(e) = 1$ if $e \notin E'$.

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.



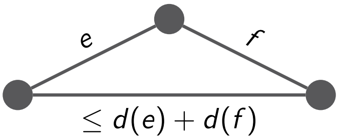
Given graph $G' = (V, E')$ define complete graph $G = (V, E)$ with $d(e) = 0$ if $e \in E'$ and $d(e) = 1$ if $e \notin E'$.

YES instance: $\text{OPT} = 0$ 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 $d(C) := \sum_{e \in C} d(e)$



The tree lower bound

Lemma 2.4

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

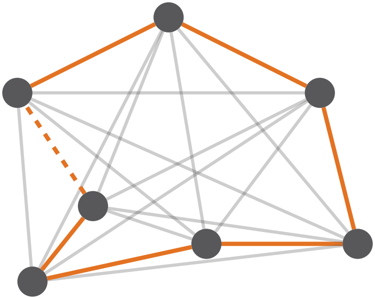
The tree lower bound

Lemma 2.4

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

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

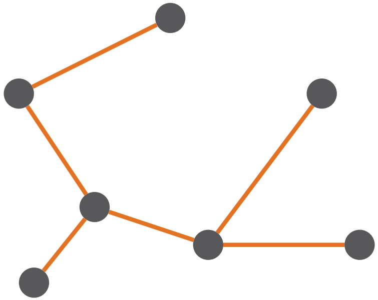


□

The double-tree algorithm

Algorithm:

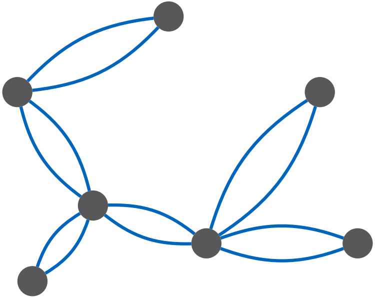
1. Compute MST T.
2. Let $H = (V,\ T \mathbin{\dot\cup} T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



The double-tree algorithm

Algorithm:

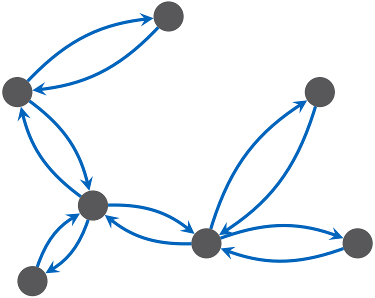
1. Compute MST T.
2. Let $H = (V,\ T \,\dot\cup\, T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



The double-tree algorithm

Algorithm:

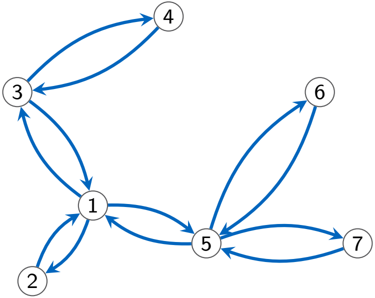
1. Compute MST T.
2. Let $H = (V,\ T \dot\cup T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



The double-tree algorithm

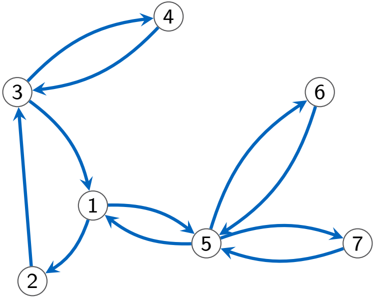
Algorithm:

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.



Algorithm:

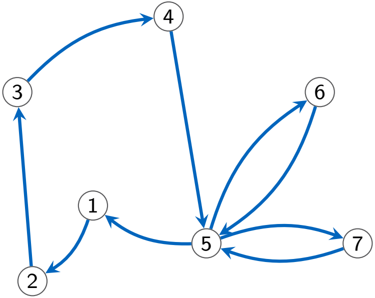
1. Compute MST T.
2. Let $H = (V,\ T \dot{\cup} T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



The double-tree algorithm

Algorithm:

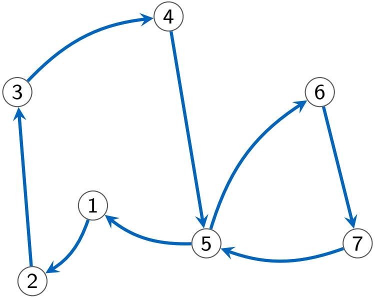
1. Compute MST T.
2. Let $H = (V,\ T \,\dot{\cup}\, T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



The double-tree algorithm

Algorithm:

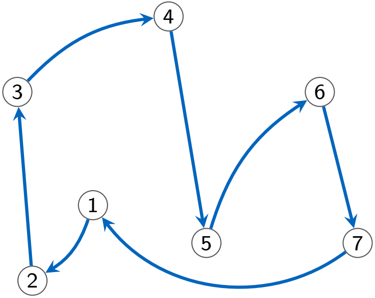
1. Compute MST T.
2. Let $H = (V,\ T \dot{\cup} T)$.
3. Compute Euler tour C' in H.
4. Shortcut C' to HC C.



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.



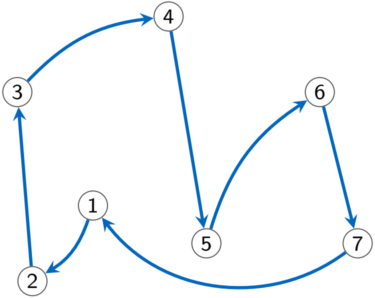
The double-tree algorithm

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 2.5

The Double-Tree Algorithm is a 2-approximation for metric TSP.



The matching lower bound

Lemma 2.6

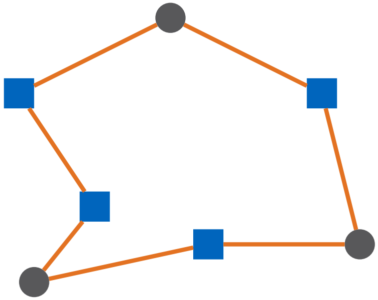
Let C be a HC in G and let $U \subseteq V$ with $|U|$ even. Let M be a minimum weight perfect matching on U. Then $d(M) \leq \frac{1}{2}d(C)$.

The matching lower bound

Lemma 2.6

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.

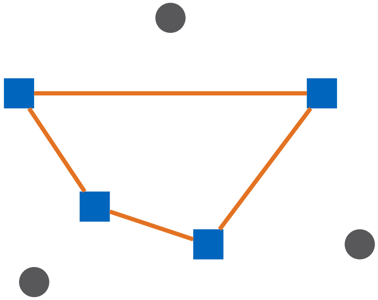


The matching lower bound

Lemma 2.6

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. $d(C') \leq d(C)$



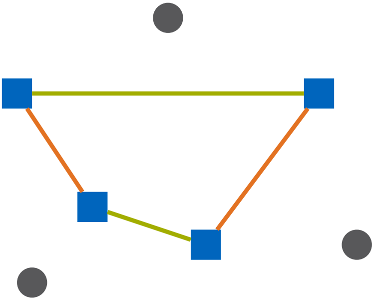
The matching lower bound

Lemma 2.6

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.

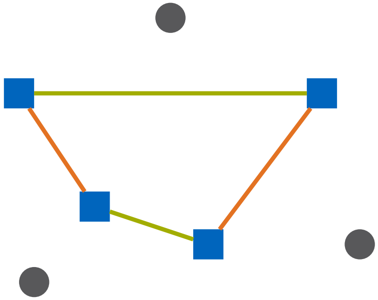


The matching lower bound

Lemma 2.6

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 $d(M) \leq \frac{1}{2}d(C)$.

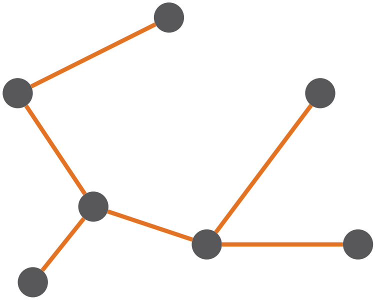
Proof. Shortcut C to cycle C' on U. $d(C') \leq d(C)$
C' contains two disjoint perfect matchings on U. $2d(M) \leq d(C')$



Christofides' algorithm

Algorithm:

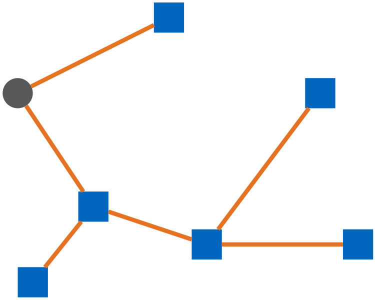
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \,\dot{\cup}\, M)$.
4. Shortcut C' to HC C.



Christofides' algorithm

Algorithm:

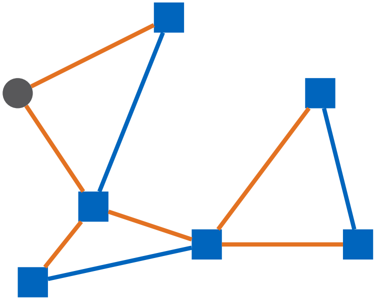
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \mathbin{\dot\cup} M)$.
4. Shortcut C' to HC C.



Christofides' algorithm

Algorithm:

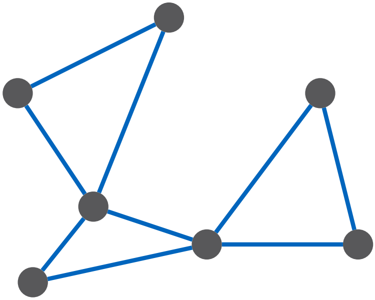
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \dot\cup M)$.
4. Shortcut C' to HC C.



Christofides' algorithm

Algorithm:

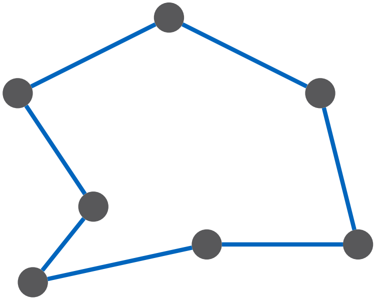
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \,\dot{\cup}\, M)$.
4. Shortcut C' to HC C.



Christofides' algorithm

Algorithm:

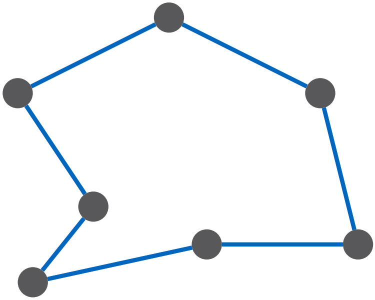
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V, T \dot\cup M)$.
4. Shortcut C' to HC C.



Christofides' algorithm

Algorithm:

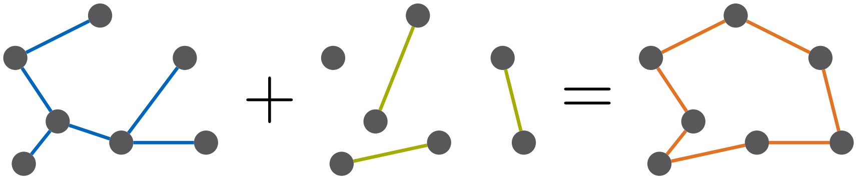
1. Compute MST T.
2. Compute minimum weight perfect matching M on $U := \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. Compute Euler tour C' in $H := (V,\ T \dot\cup M)$.
4. Shortcut C' to HC C.



Theorem 2.7

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

Summary: TSP



3/2-approximation for metric TSP