Approximating Graphic TSP by Matchings

Tobias Mömke[1] and Ola Svensson[2]

[1]KTH Royal Institute of Technology
Sweden

[2]EPFL
Switzerland

November 29, 2011

Traveling Salesman Problem

Given

- n cities
- distance $d(u, v)$ between cities u and v

Find shortest tour that visits each city once

Traveling Salesman Problem

Given

- n cities
- distance $d(u, v)$ between cities u and v

Find shortest tour that visits each city once

Traveling Salesman Problem

Given

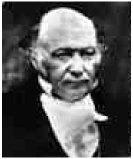
- n cities
- distance $d(u, v)$ between cities u and v

Find shortest tour that visits each city once

Classic Problem both in Practice and Theory

1800's

William Rowan Hamilton and Thomas Penyngton Kirkman studied related mathematical problems.

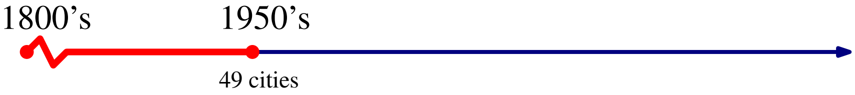


Hamilton

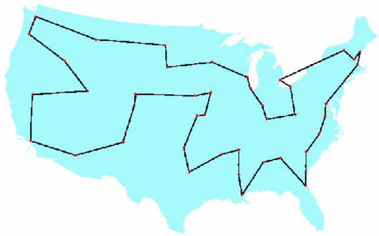
Kirkman

http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory

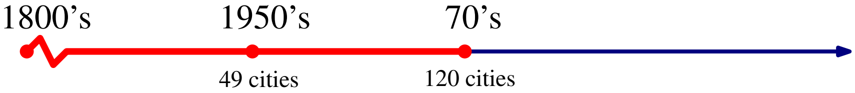


G. Dantzig, R. Fulkerson, and S. Johnson publish a method for solving the TSP and solve a 49-city instance to optimality.

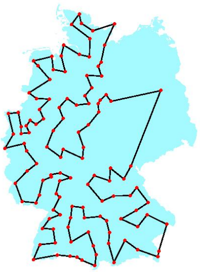


http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory

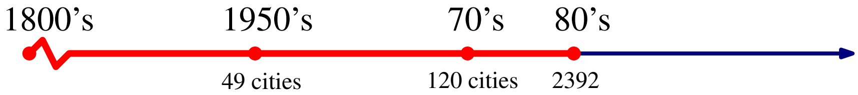


Groetschel (1977) found the optimal tour of 120 cities from what was then West Germany.



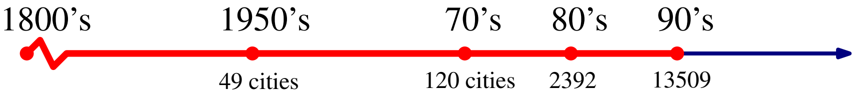
http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory



http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory

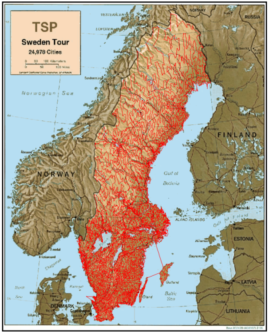


http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory

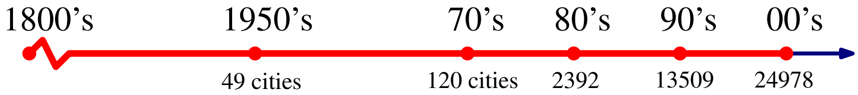
1800's	1950's	70's	80's	90's	00's
	49 cities	120 cities	2392	13509	24978

Applegate, Bixby, Chvtal, Cook, and Helsgaun (2004) found the optimal tour of 24,978 cities in Sweden.



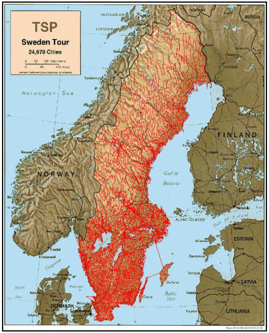
http://www.tsp.gatech.edu

Classic Problem both in **Practice** and Theory



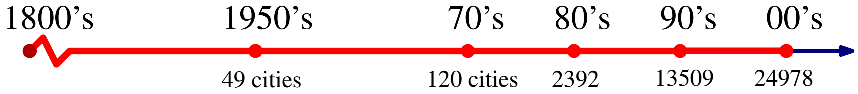
Applegate, Bixby, Chvtal, Cook, and Helsgaun (2004) found the optimal tour of 24,978 cities in Sweden.

Warning: Only 9 million people in Sweden so 360 people in average per "city".

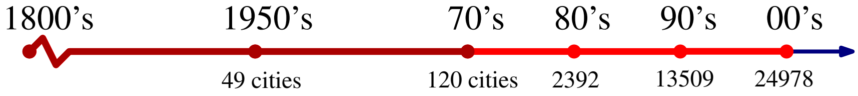


http://www.tsp.gatech.edu

Classic Problem both in Practice and **Theory**



Classic Problem both in Practice and **Theory**



Christofides

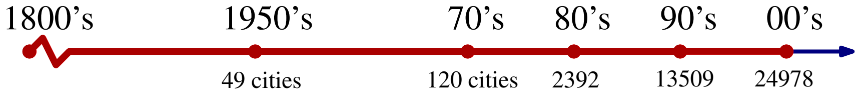
1.5-approximation algorithm for metric distances.

Held-Karp

- Heuristic for calculating a **lower bound on a tour**.
- Coincides with the value of a **linear program** known as

  Held-Karp or **Subtour Elimination relaxation.**

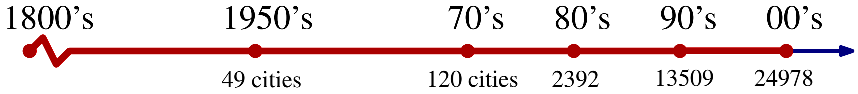
Classic Problem both in Practice and **Theory**



S. Arora and J. S. B. Mitchell independently

PTAS for **Eucledian TSP.**

Classic Problem both in Practice and **Theory**



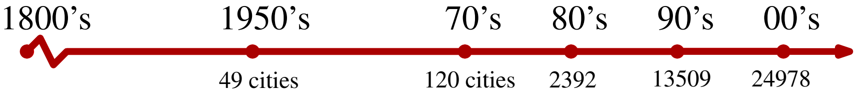
S. Arora and J. S. B. Mitchell independently

PTAS for **Eucledian TSP.**

C. H. Papadimitriou and S. Vempala

NP-hard to approximate **metric** within **220/219**.

Classic Problem both in Practice and **Theory**



Major open problem to understand approximability of metric TSP

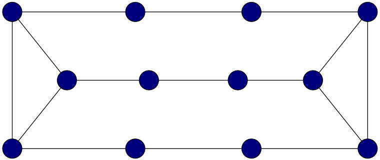
- **NP-hard** to approximate better than $220/219$.
- Christofides' **1.5-approximation algorithm** still best.
- Held-Karp relaxation **conjectured** to have **integrality gap of** $4/3$.

Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

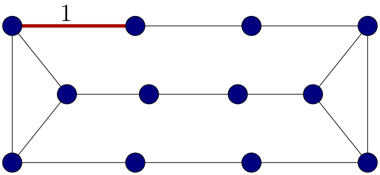


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

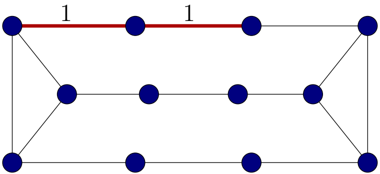


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

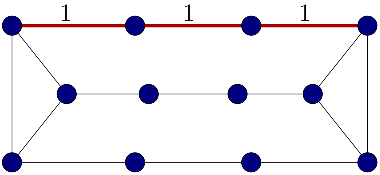


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

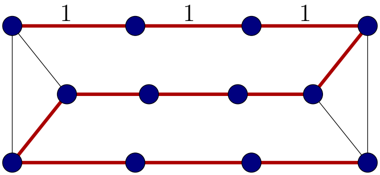


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

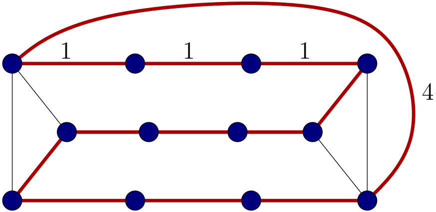


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

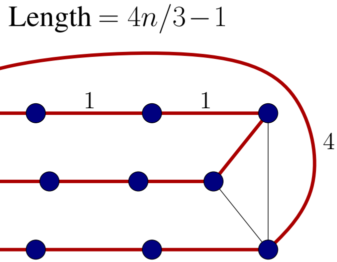


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

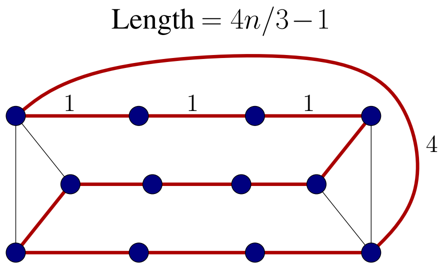


Graphic TSP (graph-TSP)

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

Find shortest tour with respect to distances

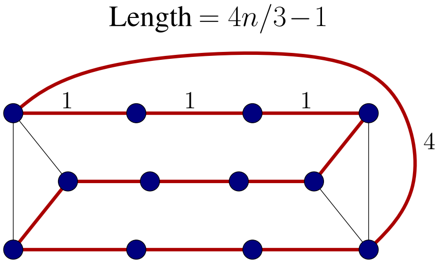
$$d(u, v) = \text{shortest path between } u \text{ and } v.$$

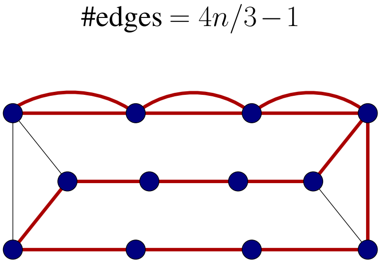


Graphic TSP (graph-TSP)

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

Find spanning Eulerian multigraph with minimum #edges





Important Special Case

- Natural problem to find smallest Eulerian subgraph
  - studied for more than 2 decades.
- Easier to study than general metrics but hopefully shed light on them
  - Still APX-hard
  - Worst instances known for Held-Karp are graphic
  - Until recently, Christofides best approximation algorithm

Recent Advancements on graph-TSP

Major open problem to understand approximability of metric TSP

- **NP-hard** to approximate better than $220/219$.
- Christofides' **1.5-approximation algorithm** still best.
- Held-Karp relaxation **conjectured** to have **integrality gap of** $4/3$.

Recent Advancements on graph-TSP

Gamarnik, Lewenstein & Sviridenko

1.487-approximation algorithm for cubic 3-edge connected graphs.

Recent Advancements on graph-TSP

Boyd, Sitters, van der Star & Stougie

- 4/3-approximation algorithm for cubic graphs
- 7/5-approximation algorithm for subcubic graphs

Recent Advancements on graph-TSP

Boyd, Sitters, van der Star & Stougie

- 4/3-approximation algorithm for cubic graphs
- 7/5-approximation algorithm for subcubic graphs

Conjecture

Subcubic 2-vertex connected graphs have a tour of length at most $\mathbf{4n/3 - 2/3}$

Recent Advancements on graph-TSP

Oveis Gharan, Saberi & Singh

$(\mathbf{1.5} - \epsilon)$-approximation algorithm for graph-TSP.

- First improvement on Christofides
- Similar to Christofides but instead of starting with a MST they sample one from the solution of the Held-Karp relaxation
- Analysis involved and requires several novel ideas

Our Results

Theorem

A **1.461**-approximation algorithm for graph-TSP.

Our Results

Theorem

A **1.461**-approximation algorithm for graph-TSP.

Based on techniques used by Frederickson & Ja'Ja'82 and Monmam, Munson & Pulleyblank'90

+ **novel use of matchings:** instead of only adding edges to make a graph Eulerian we allow for **removal of certain edges**

Our Results

Theorem

A **1.461**-approximation algorithm for graph-TSP.

Based on techniques used by Frederickson & Ja'Ja'82 and Monmam, Munson & Pulleyblank'90

+ **novel use of matchings:** instead of only adding edges to make a graph Eulerian we allow for **removal of certain edges**

Theorem

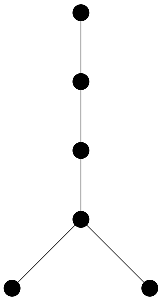
- Subcubic 2-VC graphs have a tour of length at most $\mathbf{4n/3 - 2/3}$
- A $\mathbf{4/3}$-approximation algorithm for subcubic/claw-free graphs (matching the integrality gap)

Christofides' Approach

Our Approach

Christofides' Approach

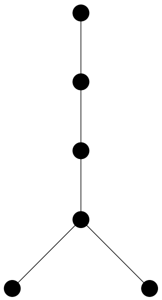
1. Find MST $T = (V, E)$.



Our Approach

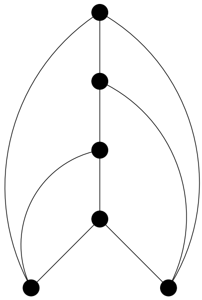
Christofides' Approach

1. Find MST $T = (V, E)$.



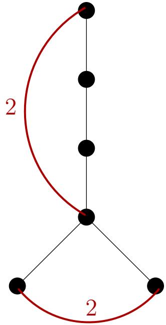
Our Approach

1. Find 2-VC subgraph $G = (V, E)$.



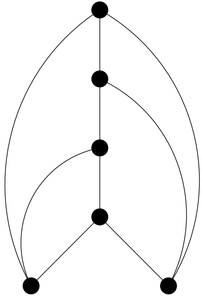
Christofides' Approach

1. Find MST $T = (V, E)$.
2. Find Minimum Matching M of odd degree vertices.



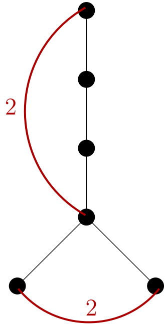
Our Approach

1. Find 2-VC subgraph $G = (V, E)$.



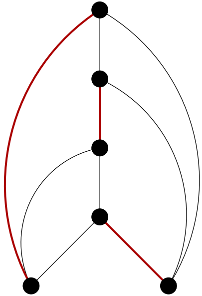
Christofides' Approach

1. Find MST $T = (V, E)$.
2. Find Minimum Matching M of odd degree vertices.



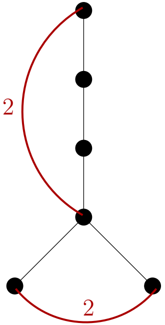
Our Approach

1. Find 2-VC subgraph $G = (V, E)$.
2. Sample perfect matching $M \subseteq E$ on the support.



Christofides' Approach

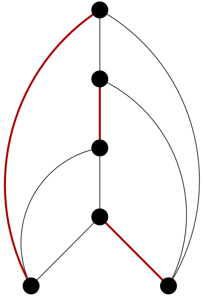
1. Find MST $T = (V, E)$.
2. Find Minimum Matching M of odd degree vertices.
3. Return $E \cup M$.



$Value = 9$

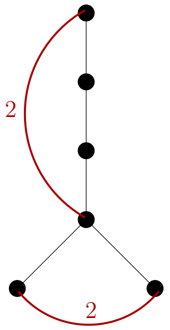
Our Approach

1. Find 2-VC subgraph $G = (V, E)$.
2. Sample perfect matching $M \subseteq E$ on the support.



Christofides' Approach

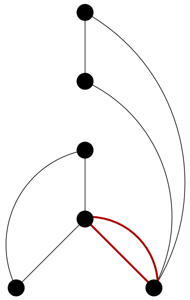
1. Find MST $T = (V, E)$.
2. Find Minimum Matching M of odd degree vertices.
3. Return $E \cup M$.



$Value = 9$

Our Approach

1. Find 2-VC subgraph $G = (V, E)$.
2. Sample perfect matching $M \subseteq E$ on the support.
3. Return $(E \cup M_{\bar{R}}) \setminus M_R$



$Value = 8$

Outline of Remaining Part

Theorem

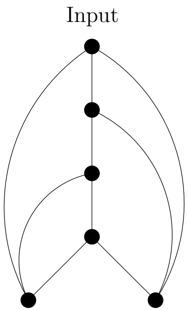
~~Sub~~cubic 2-VC graphs have a tour of length at most $\mathbf{4n/3 - 2/3}$

- Held-Karp Relaxation
- Comments on General Case

Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.



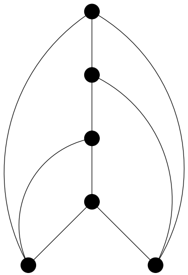
Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{\mathbf{4}}{\mathbf{3}}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$

Input

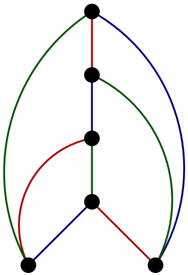


Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$



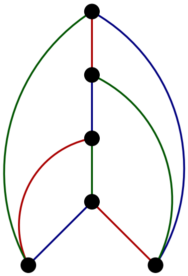
Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$
2. Return graph with edge set $E \cup M$

Input

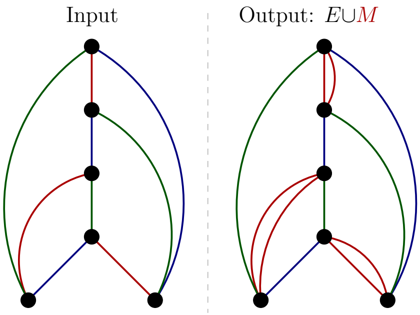


Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$
2. Return graph with edge set $E \cup M$

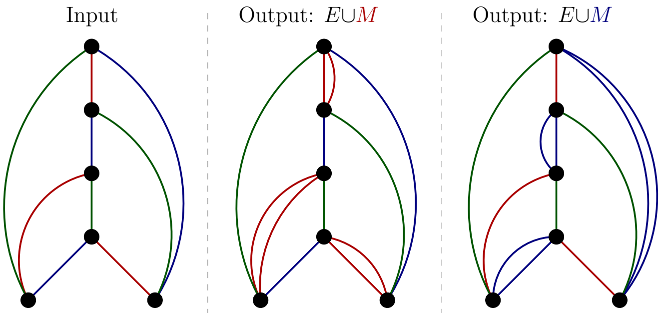


Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$
2. Return graph with edge set $E \cup M$

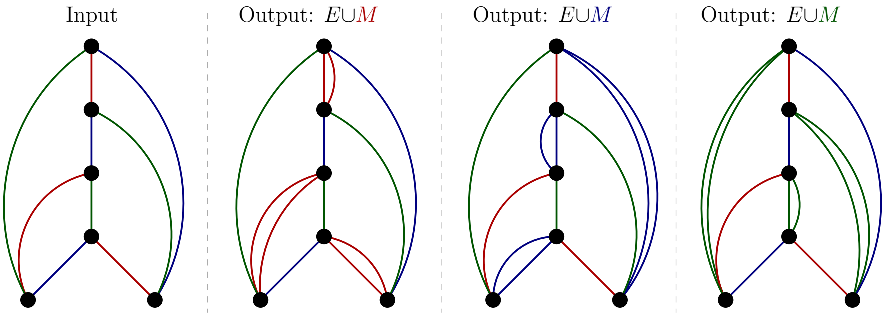


Eulerian subgraph of 2-VC graph

Frederickson & Ja'Ja'82 and Monma, Munson & Pulleyblank'90

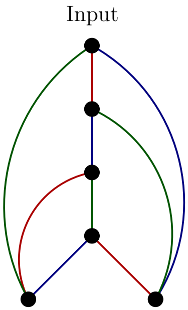
Thm: A 2-VC (cubic) graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}|\mathbf{E}|$.

1. Sample perfect matching M so that each edge is taken with probability $1/3$
2. Return graph with edge set $E \cup M$



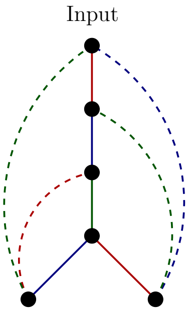
Using Matchings to Remove Edges (First Idea)

- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph



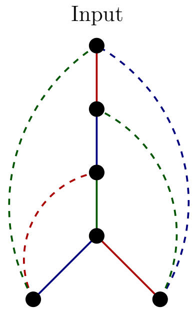
Using Matchings to Remove Edges (First Idea)

- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph



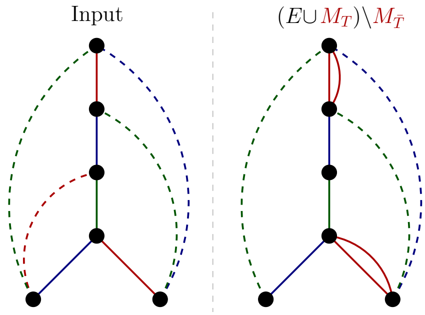
Using Matchings to Remove Edges (First Idea)

- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph
- Same algorithm as before but return $(E \cup M_T) \setminus M_{\bar{T}}$



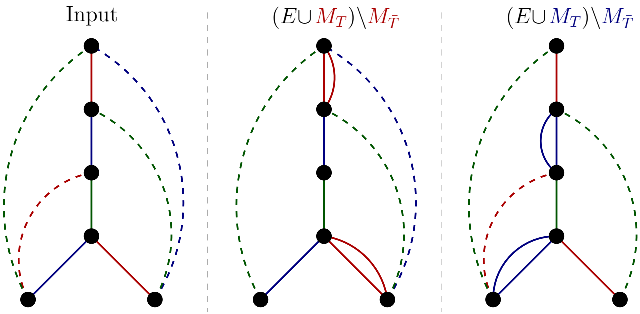
Using Matchings to Remove Edges (First Idea)

- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph
- Same algorithm as before but return $(E \cup M_T) \setminus M_{\bar{T}}$



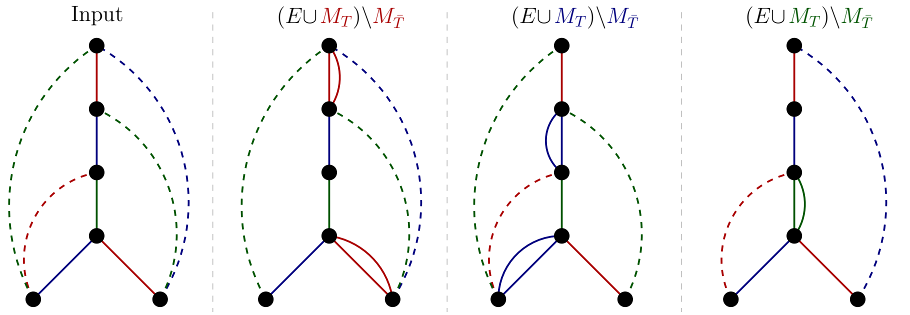
Using Matchings to Remove Edges (First Idea)

- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph
- Same algorithm as before but return $(E \cup M_T) \setminus M_{\bar{T}}$



Using Matchings to Remove Edges (First Idea)

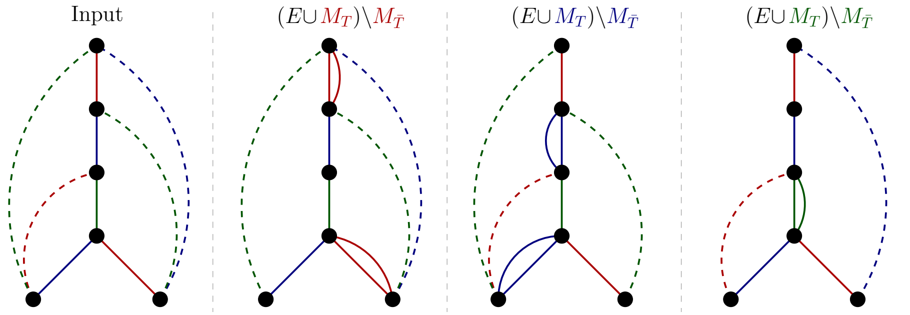
- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph
- Same algorithm as before but return $(E \cup M_T) \setminus M_{\bar{T}}$



Using Matchings to Remove Edges (First Idea)

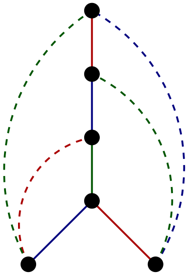
- Removing an edge from the matching will still result in even degree vertices
- If it stays connected we will again have a spanning Eulerian graph
- Same algorithm as before but return $(E \cup M_T) \setminus M_{\bar{T}}$

A 2-VC subcubic graph $G = (V, E)$ has a tour of size at most $\frac{2}{3}|\mathbf{E}| + \frac{2}{3}(\mathbf{n} - \mathbf{1})$.



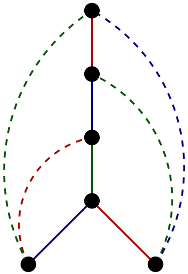
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges



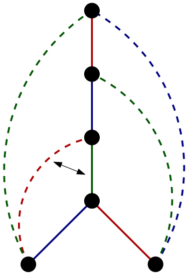
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching



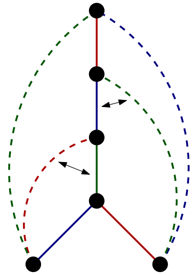
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a “removable pairing”
  - Pair of edges: only one edge in each pair can occur in a matching



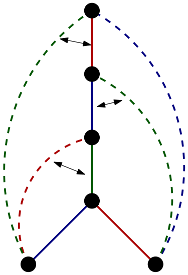
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching



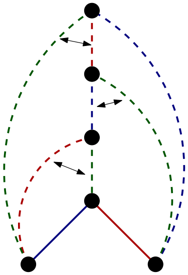
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching



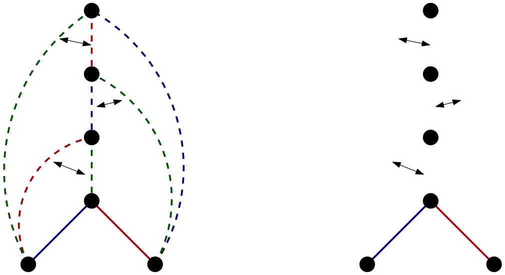
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching



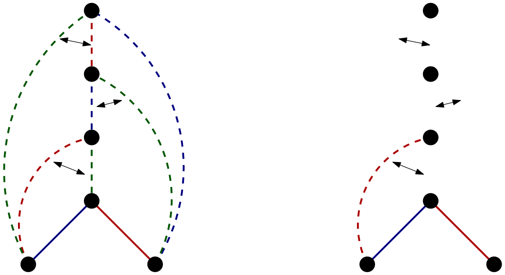
Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected



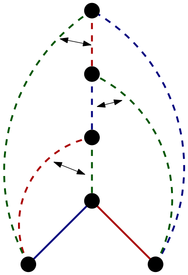
Using Matchings to Remove Edges (Second Idea)

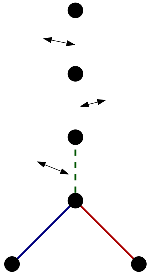
- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected



Using Matchings to Remove Edges (Second Idea)

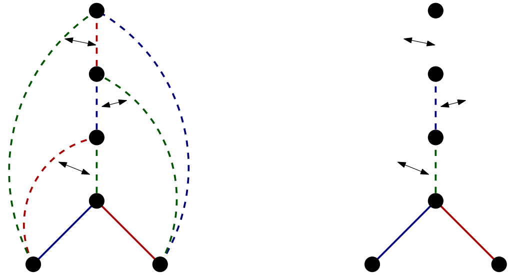
- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected





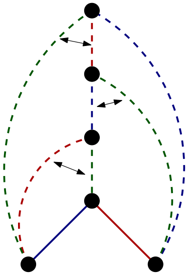
Using Matchings to Remove Edges (Second Idea)

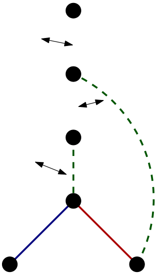
- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected



Using Matchings to Remove Edges (Second Idea)

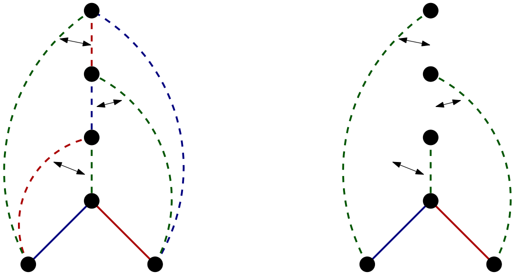
- Use structure of perfect matching to increase the set R of removable edges
- Define a “removable pairing”
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected





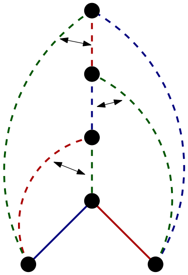
Using Matchings to Remove Edges (Second Idea)

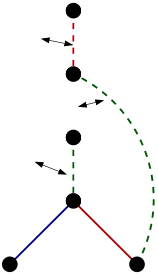
- Use structure of perfect matching to increase the set R of removable edges
- Define a “removable pairing”
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected



Using Matchings to Remove Edges (Second Idea)

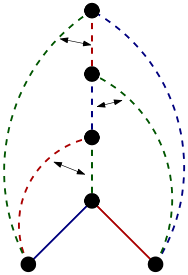
- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected

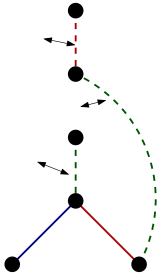




Using Matchings to Remove Edges (Second Idea)

- Use structure of perfect matching to increase the set R of removable edges
- Define a "removable pairing"
  - Pair of edges: only one edge in each pair can occur in a matching
  - Graph obtained by removing removable edges such that at most one edge in each pair is removed is connected





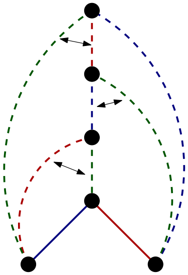
R contains all back-edges and paired tree-edges

If G has degree at most 3 then size of R is at least $2b - 1$

Using Matchings to Remove Edges (Second Idea)

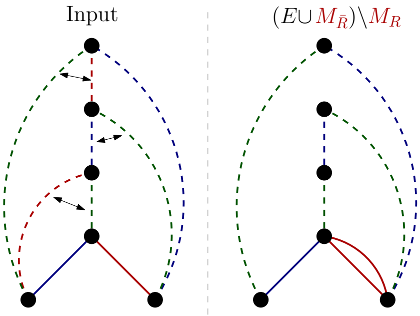
- Same algorithm as before but return $(E \cup M_{\bar{R}}) \setminus M_R$.

Input



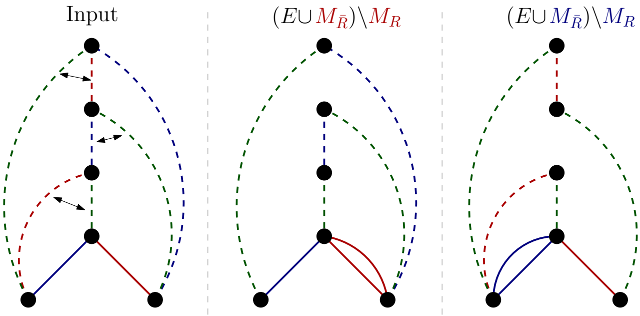
Using Matchings to Remove Edges (Second Idea)

- Same algorithm as before but return $(E \cup M_{\bar{R}}) \setminus M_R$.



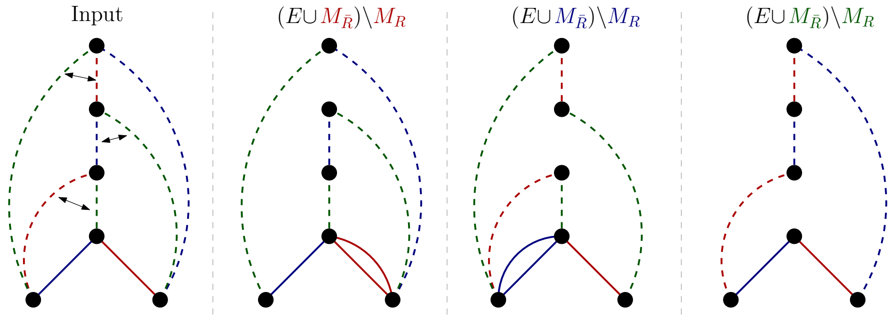
Using Matchings to Remove Edges (Second Idea)

- Same algorithm as before but return $(E \cup M_{\bar{R}}) \setminus M_R$.



Using Matchings to Remove Edges (Second Idea)

- Same algorithm as before but return $(E \cup M_{\bar{R}}) \setminus M_R$.

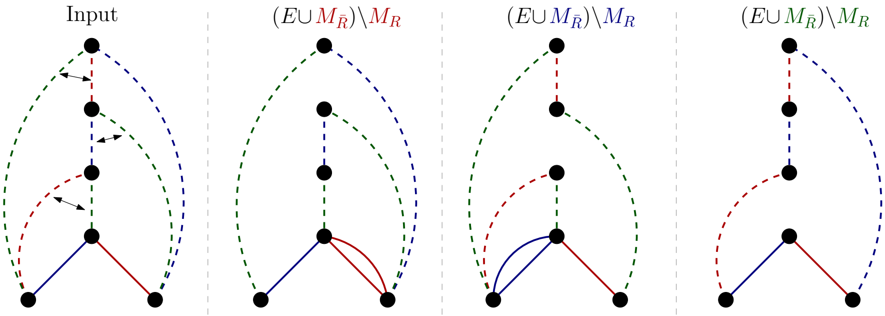


Using Matchings to Remove Edges (Second Idea)

- Same algorithm as before but return $(E \cup M_{\bar{R}}) \setminus M_R$.

Theorem

A 2-VC subcubic graph $G = (V, E)$ has a tour of size at most $\frac{4}{3}\mathbf{n} - \frac{2}{3}$.



Outline of Remaining Part

Theorem

~~Sub~~cubic 2-VC graphs have a tour of length at most $\mathbf{4n/3 - 2/3}$

- Held-Karp Relaxation
- Comments on General Case

General Statement of What We Proved

Theorem

A 2-VC graph $G = (V, E)$ with a removable pairing (R, P) has a tour of length at most $\frac{4}{3}|E| - \frac{2}{3}|R|$.

- Defining R large enough led to tight bound $4n/3 - 2/3$ for subcubic graphs.

General Statement of What We Proved

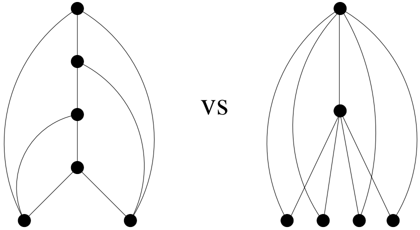
Theorem

A 2-VC graph $G = (V, E)$ with a removable pairing (R, P) has a tour of length at most $\frac{4}{3}|E| - \frac{2}{3}|R|$.

- Defining R large enough led to tight bound $4n/3 - 2/3$ for subcubic graphs.

Problem with general graphs:

1. To find a large enough removable pairing is more involved



Held-Karp Relaxation (Definition)

- A variable x_e for each edge $e \in E$
  - intuiton: value 1 if e in tour and 0 otherwise

Held-Karp Relaxation (Definition)

- A variable x_e for each edge $e \in E$
  - intuiton: value 1 if e in tour and 0 otherwise

$$\text{minimize} \sum_{e \in E} x_e$$

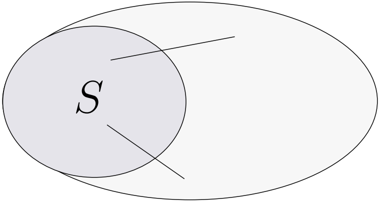
Held-Karp Relaxation (Definition)

- A variable x_e for each edge $e \in E$
  - intuiton: value 1 if e in tour and 0 otherwise

$$\text{minimize} \sum_{e \in E} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$



Held-Karp Relaxation (Useful Structure)

$$\text{minimize} \sum_{e \in E} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

- W.l.o.g. the graph is 2-VC
  - otherwise decompose instance

Held-Karp Relaxation (Useful Structure)

$$\text{minimize} \sum_{e \in E} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

- W.l.o.g. the graph is 2-VC
  - otherwise decompose instance
- The support $\{e : x_e > 0\}$ of an extreme point has size at most $2n - 1$
  - we can concentrate on very sparse graphs

Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.

Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e

Finding a Removable Pairing for General Graphs

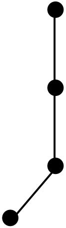
1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e

Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x_e^*

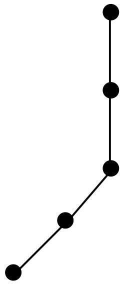
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e



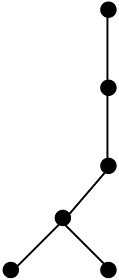
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e



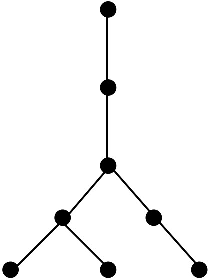
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x_e^*



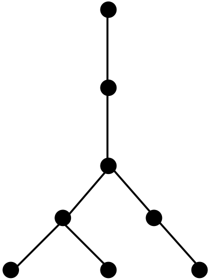
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e



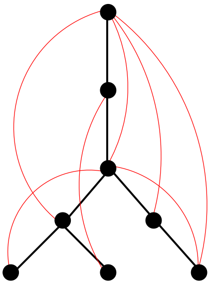
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x_e^*
3. Extend DFS tree to a 2-VC graph of minimum cost



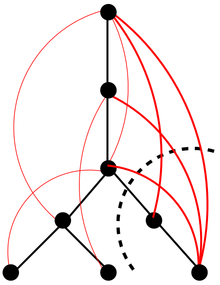
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost



Finding a Removable Pairing for General Graphs

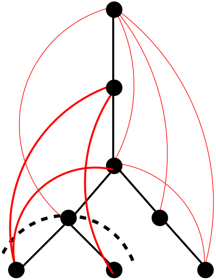
1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost



Linear constraints ensuring we pick enough red edges

Finding a Removable Pairing for General Graphs

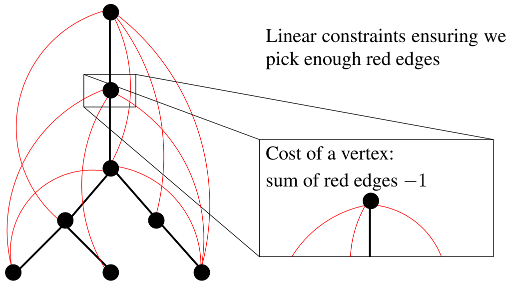
1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost



Linear constraints ensuring we pick enough red edges

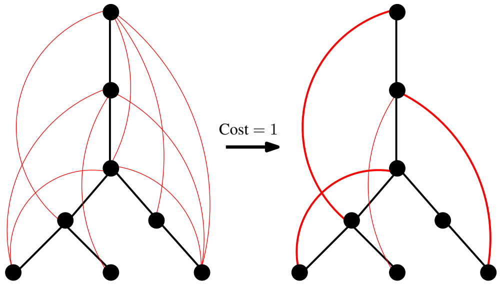
Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x_e^*
3. Extend DFS tree to a 2-VC graph of minimum cost



Finding a Removable Pairing for General Graphs

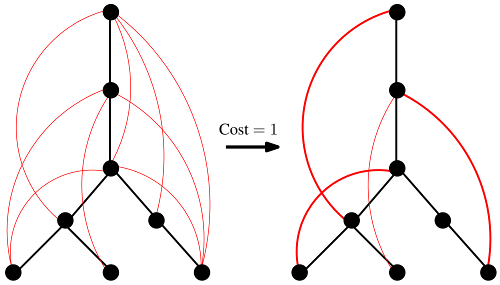
1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost



This is an integral linear program!

Finding a Removable Pairing for General Graphs

1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost
4. Sample perfect matching and remove/add edges as before

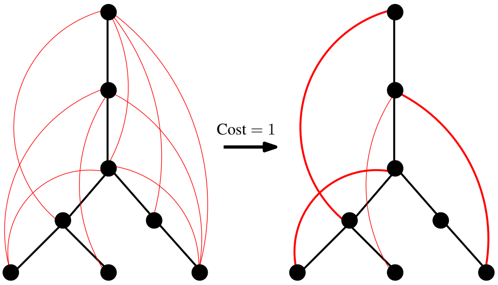


This is an integral linear program!

Finding a Removable Pairing for General Graphs

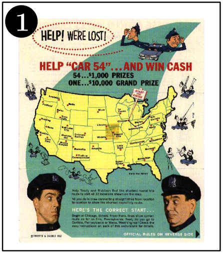
1. Solve linear program to obtain x^*.
2. Build DFS tree by in every step choosing the heaviest possible edge with respect to x^*_e
3. Extend DFS tree to a 2-VC graph of minimum cost
4. Sample perfect matching and remove/add edges as before

$$\mathbb{E}[\#\text{edges in tour}] = 4n/3 - 2/3 + \mathbf{2/3} \cdot \mathbf{c(Extension)}$$



This is an integral linear program!

Overview of Algorithm for General Graphs



Overview of Algorithm for General Graphs

1

2

$$\text{minimize} \sum_{e \in E} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

Overview of Algorithm for General Graphs

1

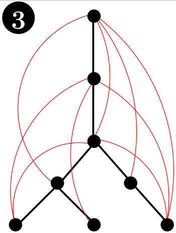
2

$$\text{minimize} \sum_{e \in E} x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

3



Overview of Algorithm for General Graphs

1

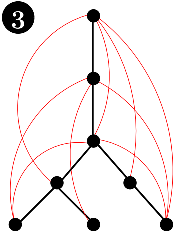
2

$$\text{minimize} \sum_{e\in E} x_e$$

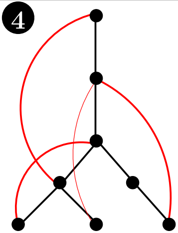
$$\sum_{e\in\delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

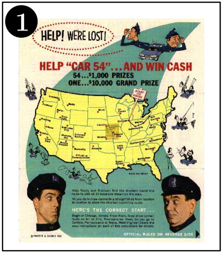
3



4



Overview of Algorithm for General Graphs

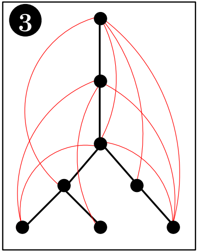


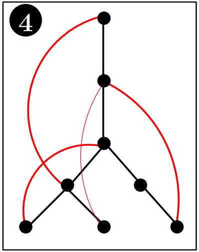
2

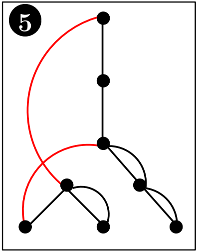
$$\text{minimize} \sum_{e\in E} x_e$$

$$\sum_{e\in\delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

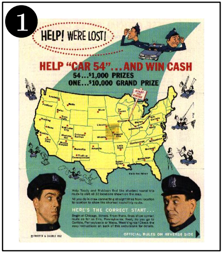
$$x \geq 0$$







Overview of Algorithm for General Graphs



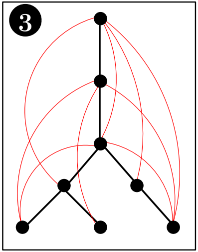
2

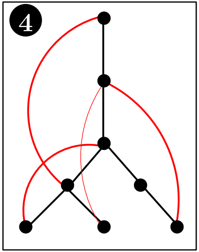
$$\text{minimize} \sum_{e \in E} x_e$$

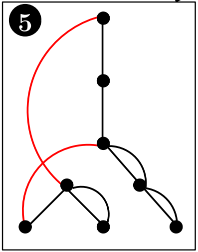
$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall \emptyset \neq S \subset V$$

$$x \geq 0$$

$$\text{Cost} = \frac{4n}{3} - \frac{2}{3} + \frac{\mathbf{2}}{\mathbf{3}} \cdot \mathbf{c(extension)}$$







Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)

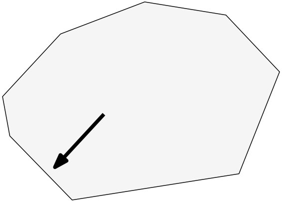
- Simplifying assumption: **cost of Held-Karp solution** x^* **is** n**.**

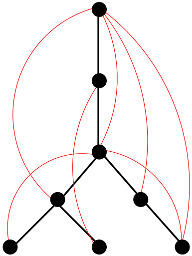
- Since

$$\sum_{\mathbf{e}\in\mathbf{E}} \mathbf{x}^*_{\mathbf{e}} = \frac{\sum_{\mathbf{v}\in\mathbf{V}} \mathbf{x}^*(\delta(\mathbf{v}))}{\mathbf{2}} \qquad \text{and} \qquad \mathbf{x}^*(\delta(\mathbf{v})) \geq \mathbf{2}$$

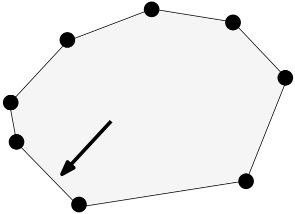
$$\mathbf{x}^*(\delta(\mathbf{v})) = 2$$

Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)

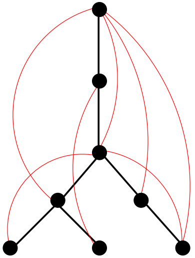




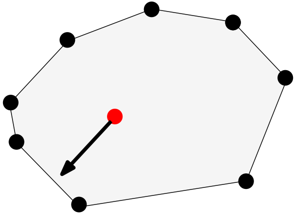
Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



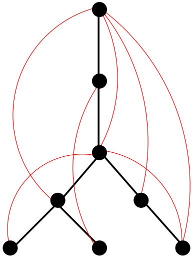
- Any extremepoint corresponds to 2-VC graph (extended from the DFS)



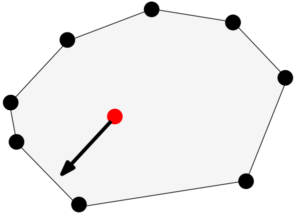
Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



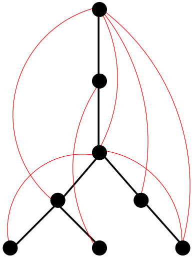
- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution



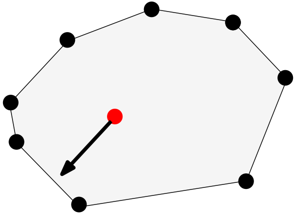
Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution
- Let the red edges have the same fractional values as from Held-Karp
  - This defines a fractional solution

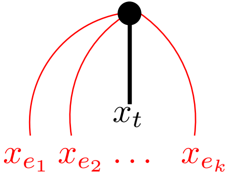


Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)

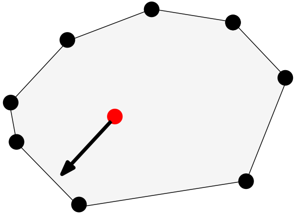


- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution
- Let the red edges have the same fractional values as from Held-Karp
  - This defines a fractional solution

The cost of fractional solution

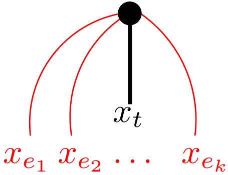


Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



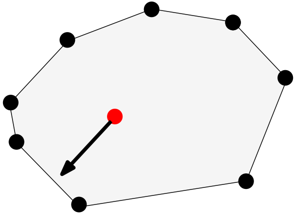
- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution
- Let the red edges have the same fractional values as from Held-Karp
  - This defines a fractional solution

The cost of fractional solution



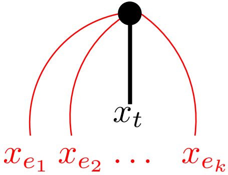
- Selection of DFS $\Rightarrow x_t \geq x_{e_i}$

Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



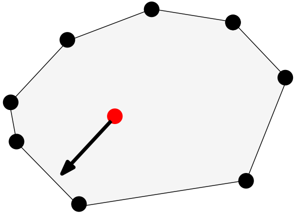
- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution
- Let the red edges have the same fractional values as from Held-Karp
  - This defines a fractional solution

The cost of fractional solution



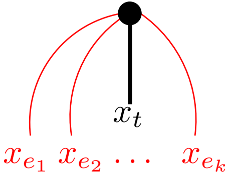
- Selection of DFS $\Rightarrow x_t \geq x_{e_i}$
- If cost is high, i.e., $\sum x_{e_i} - 1 \gg 0$
  $\Rightarrow$ many red edges

Analyzing the Cost of Extending DFS to 2-VC Graph (1/2)



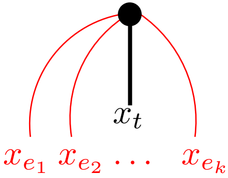
- Any extremepoint corresponds to 2-VC graph (extended from the DFS)
- Bound cost by analyzing fractional solution
- Let the red edges have the same fractional values as from Held-Karp
  - This defines a fractional solution

The cost of fractional solution



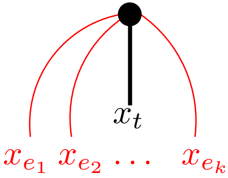
- Selection of DFS $\Rightarrow x_t \geq x_{e_i}$
- If cost is high, i.e., $\sum x_{e_i} - 1 \gg 0$
  $\Rightarrow$ many red edges
- But only $2n - 1 - (n - 1)$ red edges in total
  $\Rightarrow$ Not many vertices of high cost

Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)



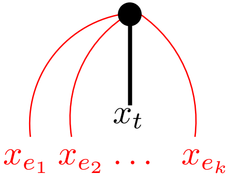
- Cost is $\sum x_{e_i} - 1$

Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)



- Cost is $\sum x_{e_i} - 1$ and $\sum x_{e_i} \leq x^*(\delta(v)) - x_t$

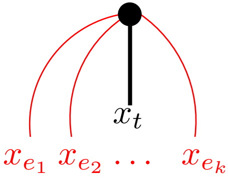
Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)



- Cost is $\sum x_{e_i} - 1$ and $\sum x_{e_i} \leq x^*(\delta(v)) - x_t$

  $\Rightarrow x_t \leq 1 - Cost$ and $\sum x_{e_i} = Cost + 1$

Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)

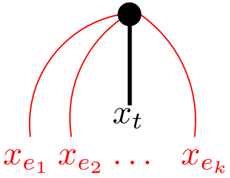


- Cost is $\sum x_{e_i} - 1$ and $\sum x_{e_i} \leq x^*(\delta(v)) - x_t$

  $$\Rightarrow x_t \leq 1 - Cost \qquad \text{and} \quad \sum x_{e_i} = Cost + 1$$

- Selection of DFS $\Rightarrow x_{e_i} \leq x_t \leq 1 - Cost$

Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)



- Cost is $\sum x_{e_i} - 1$ and $\sum x_{e_i} \leq x^*(\delta(v)) - x_t$

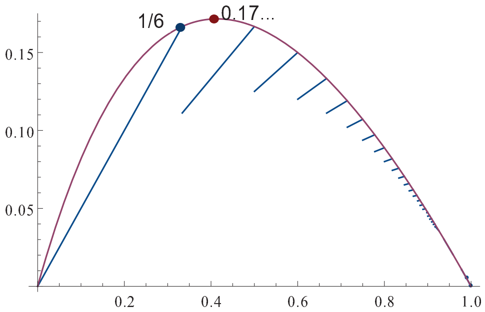
$$\Rightarrow x_t \leq 1 - Cost \qquad \text{and} \qquad \sum x_{e_i} = Cost + 1$$

- Selection of DFS $\Rightarrow x_{e_i} \leq x_t \leq 1 - Cost$
- number of red edges to vertex at least

$$\left\lceil \frac{Cost + 1}{1 - Cost} \right\rceil$$

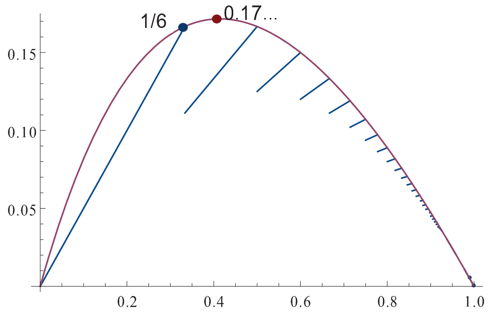
Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)

- Maximum cost per red edge (at most n many)



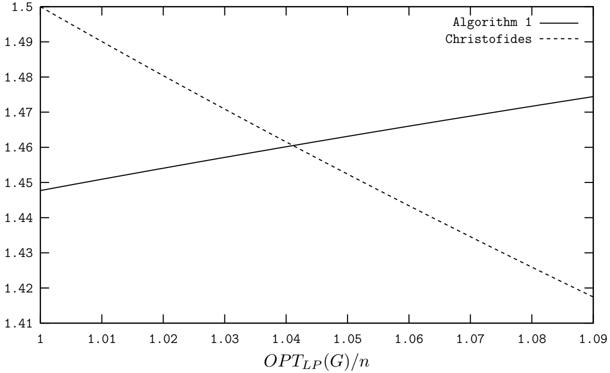
Analyzing the Cost of Extending DFS to 2-VC Graph (2/2)

- Maximum cost per red edge (at most n many)



- Cost of tour: $4n/3 + 2/3 \cdot 1/6n = (4/3 + 1/9)n$
- Cost of tour: $4n/3 + 2/3 \cdot 0.17n \approx 1.45n$

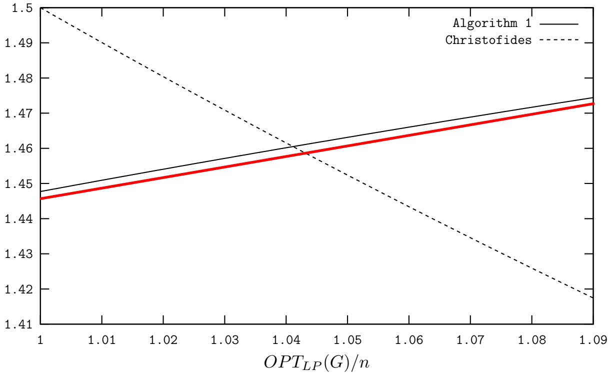
Final Result



Theorem

A 1.461-approximation algorithm for graph-TSP.

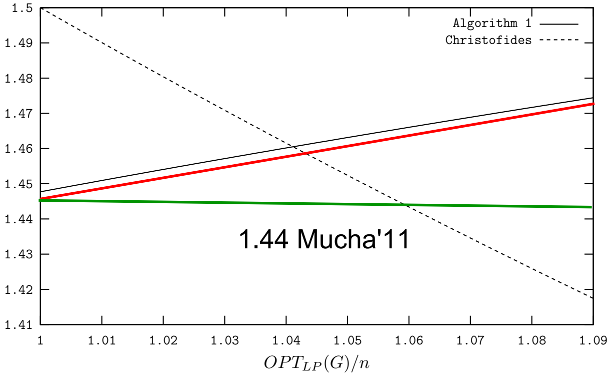
Final Result



Theorem

A 1.461-approximation algorithm for graph-TSP.

Final Result



Theorem

A 1.461-approximation algorithm for graph-TSP.

Summary

- Novel use of matchings
  - allow us to remove edges leading to decreased cost
- Bridgeless subcubic graphs have a tour of size $4n/3 - 2/3$
  - Tight analysis of Held-Karp for these graphs
- 1.461-approximation algorithm for graph-TSP
  - 13/9 analysis by Mucha'11

Summary

- Novel use of matchings
  - allow us to remove edges leading to decreased cost
- Bridgeless subcubic graphs have a tour of size $4n/3 - 2/3$
  - Tight analysis of Held-Karp for these graphs
- 1.461-approximation algorithm for graph-TSP
  - $13/9$ analysis by Mucha'11
- Generalizes to the TSP path problem on graphic metrices
  - 1.586-approximation improving on $5/3$-approximation by Hoogeveen'91
  - Tight analysis for subcubic graphs

General Metrics

No progress for TSP yet but two recent papers

A Proof of the Boyd-Carr Conjecture (Schalekamp, Williamson, and Anke van Zuylen)

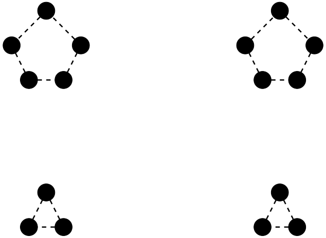
Tight analysis of cost of 2-matching vs Held-Karp Relaxation

Improving Christofides' Algorithm for the s-t Path TSP (An, Kleinberg, Shmoys)

1.62-approximation for general metrics improving upon 1.67-approximation

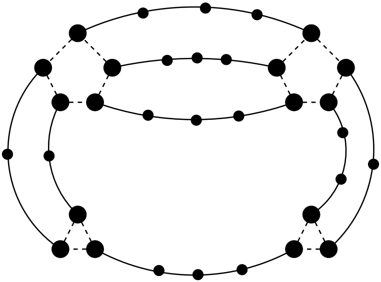
Conjectured Hardest Extreme Points

Schalekamp, Williamson, and Anke van Zuylen



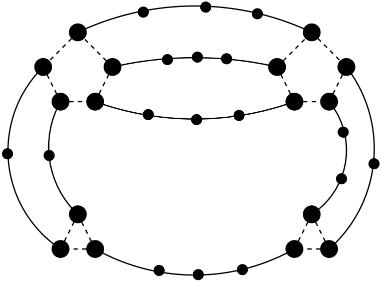
Conjectured Hardest Extreme Points

Schalekamp, Williamson, and Anke van Zuylen



Conjectured Hardest Extreme Points

Schalekamp, Williamson, and Anke van Zuylen



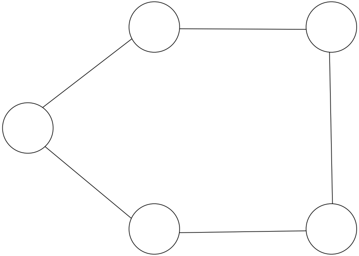
Theorem

Half-integral solutions of graph-TSP have integrality gap $\leq 4/3$

Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V, E)$ where distance of $\{u, v\} \in E$ is $f(u) + f(v)$

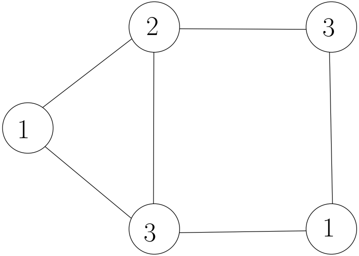
  for some $f : V \mapsto \mathbb{R}^+$



Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V, E)$ where distance of $\{u, v\} \in E$ is $f(u) + f(v)$

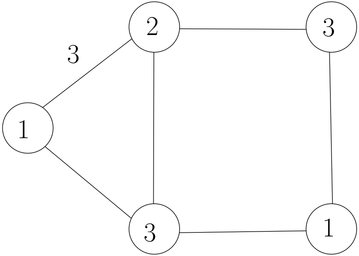
  for some $f : V \mapsto \mathbb{R}^+$



Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V, E)$ where distance of $\{u, v\} \in E$ is $f(u) + f(v)$

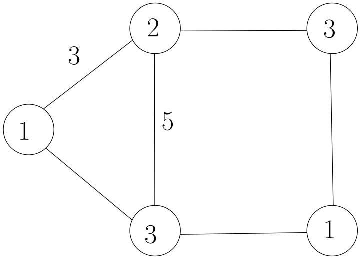
  for some $f : V \mapsto \mathbb{R}^+$



Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V, E)$ where distance of $\{u, v\} \in E$ is $f(u) + f(v)$

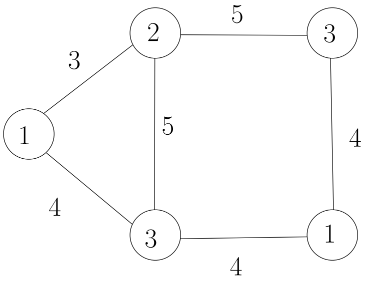
  for some $f : V \mapsto \mathbb{R}^+$



Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V, E)$ where distance of $\{u, v\} \in E$ is $f(u) + f(v)$

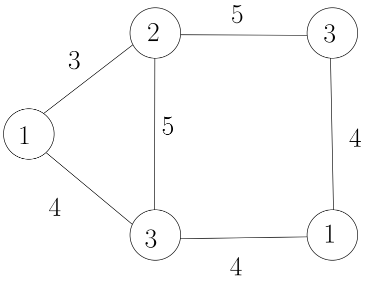
  for some $f : V \mapsto \mathbb{R}^+$



Somewhere in between graph-TSP and General Metrics

- Shortest path metric on $G(V,E)$ where distance of $\{u,v\} \in E$ is $f(u) + f(v)$

  for some $f : V \mapsto \mathbb{R}^+$



- graph-TSP if f is constant

Open Problems

- Find better removable pairing and analysis
  - If $LP = n$ is there always a $2 - VC$ subgraph of degree at most 3?
- Removable pairings straight forward to generalize to any metric
  - However, finding a large enough one remains open
- One idea is to sample an extremepoint, for example:
  - Sample two spanning trees with marginals x_e such that all edges are removable $\Rightarrow$ 4/3-approximation algorithm.

Thank You!


Stockholm
12785 km
1437 km
6237 km
Amsterdam
506 km
Paris
538 km
Lausanne
1154 km
2710 km
203 km
Beirut
Teheran
Tel Aviv