The Traveling Salesman: Classical Tools and Recent Advances

Lecture 1: Classical Results and a Conjecture

Anke van Zuylen
XV Summer School in Discrete Mathematics
Valparaiso, January 6-10, 2020

WILLIAM & MARY
CHARTERED 1693

Introduction

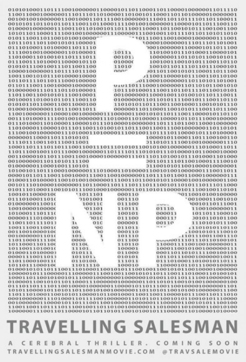
Traveling Salesman Problem (TSP)

The most famous problem in combinatorial optimization: Given n cities and the cost or distance $c(i, j)$ of traveling from city i to city j, find a minimum-cost tour that visits each city exactly once.

Optimal TSP tour through Germany's 15 largest cities (shortest among 43,589,145,600 possible tours) – Wikipedia

Traveling Salesman Problem is NP hard...

... so unless P=NP, there exists no "efficient" (polynomial-time) algorithm for finding the optimal solution.



Movie poster for Travelling Salesman movie from 2012, in which "the US government hires four mathematicians to solve the hardest problem in computer science history".

Unless P=NP...

- ... algorithms that are guaranteed to find an **optimal** solution have (worst case) running time that grows **exponentially** with the input size;

Optimal tour through UK's 24,727 pubs – Cook, Espinoza, Goycoolea and Helsgaun 2015

Unless P=NP...

- ... algorithms that are guaranteed to find an **optimal** solution have (worst case) running time that grows **exponentially** with the input size;
- but, we can develop polynomial time algorithms that have a guarantee on the quality of the solution (the guaranteed quality is just not the optimum)!

Optimal tour through UK's 24,727 pubs – Cook, Espinoza, Goycoolea and Helsgaun 2015

Approximation Algorithm

Definition

An α-approximation algorithm is a polynomial-time algorithm that returns a solution of cost at most α times the cost of an optimal solution.

Classical Problem, but with Lots of News to Report

ERICA KLARREICH SCIENCE 01.30.13 09:30 AM

COMPUTER SCIENTISTS FIND NEW SHORTCUTS FOR INFAMOUS TRAVELING SALESMAN PROBLEM

TRAVELING SALESMAN PROBLEM

Computer Scientists Take Road Less Traveled

By ERICA KLARREICH — 3

An infinitesimal advance in the traveling salesman problem breathes new life into the search for improved approximate solutions.

ABSTRACTIONS BLOG

One-Way Salesman Finds Fast Path Home

By MARK H. KIM — 6

The real-world version of the famous "traveling salesman problem" finally gets a good-enough solution.

Classical Approximation Algorithms for TSP

Traveling Salesman Problem (TSP)

Input:

- A complete graph $G = (V, E)$;
- Edge costs $c(e) \equiv c(i,j) \geq 0$ for all $e = (i,j) \in E$;
- Edge costs satisfy the **triangle inequality**: $c(i,j) \leq c(i,k) + c(k,j)$ for all i, j, k.

Goal: Find a min-cost tour that visits each vertex exactly once.

Edges may be

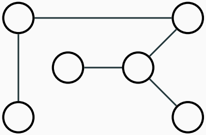
- undirected; then $c(i,j) = c(j,i)$
  (symmetric TSP – focus of this course)
- directed; then $c(i,j)$ may not be equal to $c(j,i)$
  (asymmetric TSP – we will not cover this)

Doubled Spanning Tree Algorithm

- Compute a minimum spanning tree (MST) T on G.

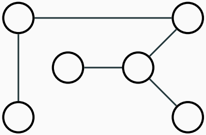
Doubled Spanning Tree Algorithm

- Compute a minimum spanning tree (MST) T on G.



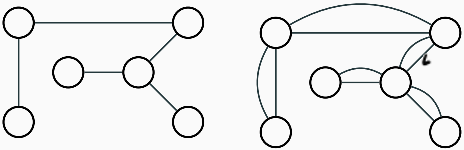
Doubled Spanning Tree Algorithm

- Compute a minimum spanning tree (MST) T on G.
- Double every edge in T.



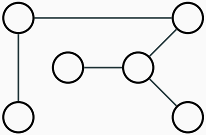
Doubled Spanning Tree Algorithm

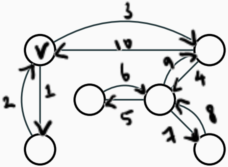
- Compute a minimum spanning tree (MST) T on G.
- Double every edge in T.
  - The result is an Eulerian graph: it is connected, and every vertex has even degree.



Doubled Spanning Tree Algorithm

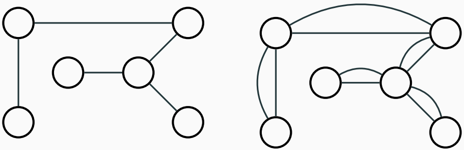
- Compute a minimum spanning tree (MST) T on G.
- Double every edge in T.
  - The result is an Eulerian graph: it is connected, and every vertex has even degree.
  - An Eulerian graph has an traversal that is easy to compute; it starts at any vertex v, visits every edge, and returns to v.





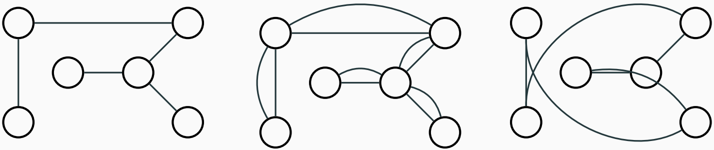
Doubled Spanning Tree Algorithm

- Compute a minimum spanning tree (MST) T on G.
- Double every edge in T.
    - The result is an Eulerian graph: it is connected, and every vertex has even degree.
    - An Eulerian graph has an traversal that is easy to compute; it starts at any vertex v, visits every edge, and returns to v.
- Compute the traversal, and follow it; if the next edge goes back to a previously visited vertex, shortcut it, and go on to the next vertex in the traversal.



Doubled Spanning Tree Algorithm

- Compute a minimum spanning tree (MST) T on G.
- Double every edge in T.
  - The result is an Eulerian graph: it is connected, and every vertex has even degree.
  - An Eulerian graph has an traversal that is easy to compute; it starts at any vertex v, visits every edge, and returns to v.
- Compute the traversal, and follow it; if the next edge goes back to a previously visited vertex, shortcut it, and go on to the next vertex in the traversal.



Analysis

Theorem

The doubled spanning tree algorithm is a 2-approximation algorithm for the TSP.

Let $c(T)$ be the cost of the edges in the MST. Let OPT be the cost of the optimal tour.

Lemma

$$c(T) \leq OPT.$$

Proof: Removing one edge from optimal tour gives a spanning tree T' with $c(T') \leq OPT$.
So $c(T) \leq c(T') \leq OPT$.

□

Proof of Theorem

$2c(T) \leq 2\mathrm{OPT}$

Shortcutting traversal of the Eulerian graph doesn't increase the cost

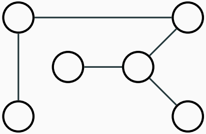
□

Proof of Theorem

□

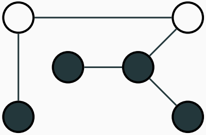
Christofides-Serdyukov's Tree+Matching Algorithm

- Compute minimum spanning tree (MST) T on G. *"Connectivity"*



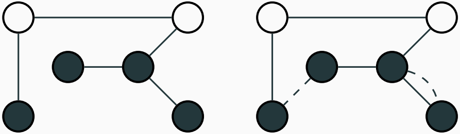
Christofides-Serdyukov's Tree+Matching Algorithm

- Compute minimum spanning tree (MST) T on G. *"Connectivity"*
- Let Odd_T be the odd-degree vertices of T. Compute a minimum-cost matching M of Odd_T. *"Parity correction"*



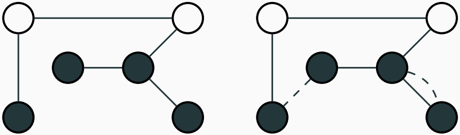
Christofides-Serdyukov's Tree+Matching Algorithm

- Compute minimum spanning tree (MST) T on G. *"Connectivity"*
- Let Odd_T be the odd-degree vertices of T. Compute a minimum-cost matching M of Odd_T. *"Parity correction"*



Christofides-Serdyukov's Tree+Matching Algorithm

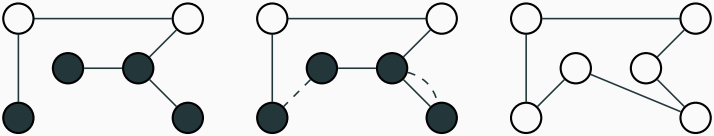
- Compute minimum spanning tree (MST) T on G. *"Connectivity"*
- Let Odd_T be the odd-degree vertices of T. Compute a minimum-cost matching M of Odd_T. *"Parity correction"*
- "Shortcut" Eulerian traversal in resulting Eulerian graph $(V, T \sqcup M)$[1].



[1] $\sqcup$ is the disjoint union, so $T \sqcup M$ has two copies of e if $e \in T$ and $e \in M$.

Christofides-Serdyukov's Tree+Matching Algorithm

- Compute minimum spanning tree (MST) T on G. *"Connectivity"*
- Let Odd_T be the odd-degree vertices of T. Compute a minimum-cost matching M of Odd_T. *"Parity correction"*
- "Shortcut" Eulerian traversal in resulting Eulerian graph $(V, T \sqcup M)$[1].



[1] $\sqcup$ is the disjoint union, so $T \sqcup M$ has two copies of e if $e \in T$ and $e \in M$.

Analysis

Theorem (Christofides (1976), Serdyukov (1978))

The Tree+Matching algorithm is a $\frac{3}{2}$-approximation algorithm for the TSP.

Let $c(T)$ be the cost of the edges in the MST, $c(M)$ the cost of the edges in the matching on Odd_T, and let OPT be the cost of the optimal tour.

Lemma

$$c(T) \leq OPT.$$ ✓

Lemma

$$c(M) \leq \frac{1}{2} OPT.$$

Proof of Second Lemma

Color the edges of the optimal tour with two colors, changing color at a vertex in Odd_T but not at a vertex in $V \setminus \mathrm{Odd}_T$.[2]

[2] The edges of the same color form an Odd_T-join: a set of edges J is an Odd_T-join if each vertex in Odd_T has odd degree in J, and each vertex in $V \setminus \mathrm{Odd}_T$ has even degree in J.

Proof of Second Lemma

Color the edges of the optimal tour with two colors, changing color at a vertex in Odd_T but not at a vertex in $V \setminus \text{Odd}_T$.[2] Shortcutting the paths of edges of the same color gives two matchings on Odd_T.

[2] The edges of the same color form an Odd_T-join: a set of edges J is an Odd_T-join if each vertex in Odd_T has odd degree in J, and each vertex in $V \setminus \text{Odd}_T$ has even degree in J.

Four-Thirds Conjecture

We just showed the following classical result.

Theorem

There exists a polynomial-time algorithm that finds a tour of cost at most $\frac{3}{2}OPT$.

Four-Thirds Conjecture

We just showed the following classical result.

Theorem

There exists a polynomial-time algorithm that finds a tour of cost at most $\frac{3}{2}OPT$.

There is a well-known conjecture that this result can be improved.

Conjecture

There exists a polynomial-time algorithm that finds a tour of cost at most $\frac{4}{3}OPT$.

Why $4/3$??

- Linear programs can be solved in polynomial time, and are a popular tool in developing approximation algorithms.
- In this case, $4/3$ seems like the best we can hope for using a well-known linear program.

Linear Programs Related to the TSP

An Integer Programming Formulation

For each edge $e \in E$, we introduce a decision variable $x(e)$, for which we want

$$x(e) = \begin{cases} 1 & \text{if tour uses edge } e \\ 0 & \text{otherwise} \end{cases}$$

An Integer Programming Formulation

For each edge $e \in E$, we introduce a decision variable $x(e)$, for which we want

$$x(e) = \begin{cases} 1 & \text{if tour uses edge } e \\ 0 & \text{otherwise} \end{cases}$$

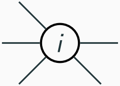
Then our <u>objective function</u> is to

$$\text{Minimize} \sum_{e \in E} c(e)x(e).$$

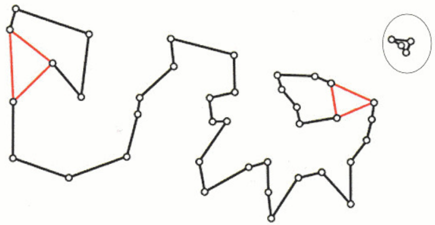
Constraints

Let $\delta(i)$ represent the set of all edges e that have $i \in V$ as one endpoint. Then we want

$$\sum_{e \in \delta(i)} x(e) = 2$$



for all $i \in V$.



From Cook, p. 128

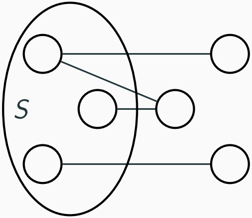
We need additional constraints...

Constraints

For $S \subseteq V$, let $\delta(S)$ be the set of edges with one endpoint in S. We also want

$$\sum_{e \in \delta(S)} x(e) \geq 2$$

for any set S.



Integer Program (IP) for the TSP

Minimize $\sum_{e \in E} c(e)x(e)$

subject to:

$$\sum_{e \in \delta(i)} x(e) = 2 \qquad \forall i \in V,$$

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

$$x(e) \in \{0, 1\} \qquad \forall e \in E.$$

Integer Program (IP) for the TSP

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{e \in E} c(e)x(e) \\
\text{subject to:} \quad & \\
& \sum_{e \in \delta(i)} x(e) = 2 && \forall i \in V, \\
& \sum_{e \in \delta(S)} x(e) \geq 2 && \forall S \subset V, S \neq \emptyset, \\
& x(e) \in \{0,1\} && \forall e \in E.
\end{aligned}
$$

Replace $x(e) \in \{0,1\}$ by $0 \leq x(e) \leq 1$ to obtain a linear programming (LP) relaxation called the Subtour LP.

The Subtour LP

- Subtour LP can be solved in polynomial time.

Minimize $\sum_{e \in E} c(e)x(e)$

subject to:

$$\sum_{e \in \delta(i)} x(e) = 2 \quad \forall i \in V,$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \quad \forall S \subset V, S \neq \emptyset,$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

The Subtour LP

Minimize $\sum_{e \in E} c(e)x(e)$

subject to:

$$\sum_{e \in \delta(i)} x(e) = 2 \quad \forall i \in V,$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \quad \forall S \subset V, S \neq \emptyset,$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

- Subtour LP can be solved in polynomial time.
- If we are “lucky” and the optimal solution is integer, then it gives us an optimal TSP tour.

The Subtour LP

Minimize $\sum_{e \in E} c(e)x(e)$

subject to:

$$\sum_{e \in \delta(i)} x(e) = 2 \quad \forall i \in V,$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \quad \forall S \subset V, S \neq \emptyset,$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

- Subtour LP can be solved in polynomial time.
- If we are "lucky" and the optimal solution is integer, then it gives us an optimal TSP tour.
- Whether or not we are "lucky", let OPT be the cost of the min-cost tour, and OPT_{LP} the LP optimal value, then:

$$OPT_{LP} \leq OPT$$

The Subtour LP

Minimize $\sum_{e \in E} c(e)x(e)$

subject to:

$$\sum_{e \in \delta(i)} x(e) = 2 \quad \forall i \in V,$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \quad \forall S \subset V, S \neq \emptyset,$$

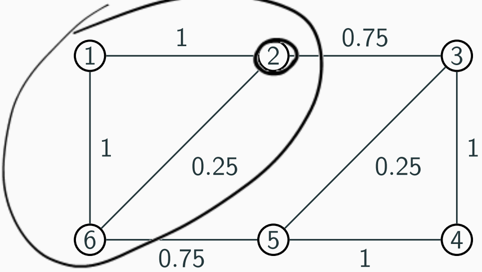
$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

- Subtour LP can be solved in polynomial time.
- If we are "lucky" and the optimal solution is integer, then it gives us an optimal TSP tour.
- Whether or not we are "lucky", let OPT be the cost of the min-cost tour, and OPT_{LP} the LP optimal value, then:

$$OPT_{LP} \leq OPT.$$

Shorthand Notation

We'll sometimes use the shorthand $x(F) = \sum_{e \in F} x(e)$, or $c(F) = \sum_{e \in F} c(e)$.



Example: if the values indicated on the edges in the graph above are $x(e)$'s, then

- $x(\delta(2)) =$ $x(1,2) + x(2,6) + x(2,3) = 1 + 0.25 + 0.75 = 2$
- For $S = \{1, 2, 6\}$, $x(\delta(S)) =$ $x(2,3) + x(6,5) = 0.75 + 0.75 = 1.5$

Shorthand Notation

This allows us to rewrite the Subtour LP as:

Minimize $\sum_{e \in E} c(e)\, x(e)$

subject to:

$x(\delta(i)) = 2 \quad \forall i \in V$

$x(\delta(S)) \geq 2 \quad \forall S \subset V, S \neq \emptyset$

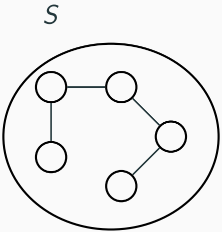
$0 \leq x(e) \leq 1 \quad \forall e \in E.$

Equivalent Constraints

An equivalent way to write the subtour elimination constraints is via a constraint that says no cycles in any strict subset. Let $E(S)$ be the set of edges with both endpoints in S; then

$$x(E(S)) \leq |S| - 1$$

for all $S \subset V, |S| \geq 2$.



Equivalent LP

So an LP that's equivalent to the subtour LP is the following:

$$\text{Minimize} \quad \sum_{e \in E} c(e)x(e)$$

subject to:

$$\begin{aligned} x(\delta(i)) &= 2 && \forall i \in V \\ x(E(S)) &\leq |S| - 1 && \forall S \subset V, |S| \geq 2 \\ 0 \leq x(e) &\leq 1 && \forall e \in E. \end{aligned}$$

The Tree+Matching Algorithm Again

Theorem (Wolsey (1980), Cunningham (1986), Shmoys & Williamson (1990))

The Tree+Matching algorithm returns a tour of cost at most $\frac{3}{2} OPT_{LP}$ *for the TSP.*

The Tree+Matching Algorithm Again

Theorem (Wolsey (1980), Cunningham (1986), Shmoys & Williamson (1990))

The Tree+Matching algorithm returns a tour of cost at most $\frac{3}{2}OPT_{LP}$ for the TSP.

To prove this, we need to show :

Lemma

$$c(T) \leq OPT_{LP}$$

Lemma

$$c(M) \leq \frac{1}{2}OPT_{LP}$$

where T is the MST, and M is a minimum-cost matching on Odd_T.

Proof of First Lemma

The minimum spanning tree can be found as the solution to the following LP (Edmonds 1971):

$$\text{Minimize} \quad \sum_{e \in E} c(e) z(e)$$

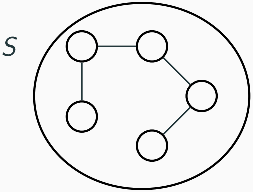
subject to:

$$z(E) = |V| - 1$$

$$z(E(S)) \leq |S| - 1 \qquad \forall S \subset V, |S| \geq 2$$

$$z(e) \geq 0 \qquad \forall e \in E.$$

Optimal extreme point is integer & a MST



Proof of First Lemma: $c(T) \leq OPT_{LP}$

Subtour LP

$$\text{Min} \sum_{e \in E} c(e)x(e)$$

$$\text{s.t.} \quad x(\delta(i)) = 2 \quad \forall i \in V,$$

$$x(E(S)) \leq |S| - 1 \quad \forall S \subset V, |S| \geq 2,$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

Spanning Tree LP

$$\text{Min} \sum_{e \in E} c(e)z(e)$$

$$\text{s.t.} \quad z(E) = |V| - 1,$$

$$z(E(S)) \leq |S| - 1 \quad \forall S \subset V, |S| \geq 2,$$

$$z(e) \geq 0 \quad \forall e \in E.$$

Let x^* be an optimal subtour LP solution.

Then $x^*(E) = \frac{1}{2} \sum_{i \in V} x^*(\delta(i)) = \frac{1}{2} \cdot 2|V| = |V|$

Let $z(e) = \frac{|V|-1}{|V|} x^*(e)$ for every $e \in E$.

Then z is a feasible solution to the spanning tree LP.

$$c(T) \leq \sum_{e \in E} c(e) z(e) = \frac{|V|-1}{|V|} \sum_{e \in E} c(e) x^*(e)$$

$$= \frac{|V|-1}{|V|} OPT_{LP} \leq OPT_{LP}$$

□

Proof of Second Lemma

The minimum-cost matching on W can be found as the solution to the following LP (Edmonds 1965):

$$\text{Minimize} \quad \sum_{e \in E} c(e) z(e)$$

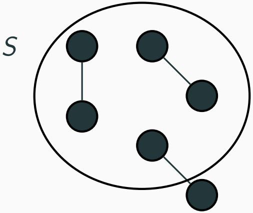
subject to:

$$z(\delta(i)) = \begin{cases} 1 & \forall i \in W, \\ 0 & \forall i \in V \setminus W, \end{cases}$$

$$z(\delta(S)) \geq 1 \qquad \forall S \subset V, |S \cap W| \text{ odd},$$

$$z(e) \geq 0 \qquad \forall e \in E.$$

Optimal extreme point is a matching of W



Proof of Second Lemma

If edge costs are non-negative and satisfy triangle inequality, we can omit the first set of constraints , so the minimum-cost matching on W can be found as the solution to the following LP:

$$\text{Minimize} \quad \sum_{e \in E} c(e)z(e)$$

subject to:

$$z(\delta(S)) \geq 1 \qquad \forall S \subset V, |S \cap W| \text{ odd},$$
$$z(e) \geq 0 \qquad \forall e \in E.$$

(Feasible region of this LP is called the (dominant of the) W-join polytope, but we will abuse terminology and call this the W-matching LP.)

Proof of Second Lemma: $c(M) \leq \frac{1}{2} OPT_{LP}$

Subtour LP

$$\text{Min} \sum_{e \in E} c(e)x(e)$$

$$x(\delta(i)) = 2 \quad \forall i \in V$$

$$x(\delta(S)) \geq 2 \quad \forall S \subset V, S \neq \emptyset$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

Odd_T-Matching LP (if edge costs non-negative, and satisfy triangle inequality)

$$\text{Min} \sum_{e \in E} c(e)z(e)$$

$$z(\delta(S)) \geq 1 \quad \forall S \subset V, |S \cap \text{Odd}_T| \text{ odd}$$

$$z(e) \geq 0 \quad \forall e \in E.$$

Let x^* be an optimal solution to the Subtour LP.
Let $z(e) = \frac{1}{2} x^*(e)$ for every $e \in E$.
Then z is a feasible solution to the Odd_T-matching LP. So $c(M) \leq \sum_{e \in E} c(e) z(e)$

$$= \frac{1}{2} OPT_{LP}$$

□

Back to the Four-Thirds Conjecture

We have shown

Theorem

The Christofides-Serdyukov Tree+Matching algorithm returns a solution of cost at most $\frac{3}{2}OPT_{LP}$.

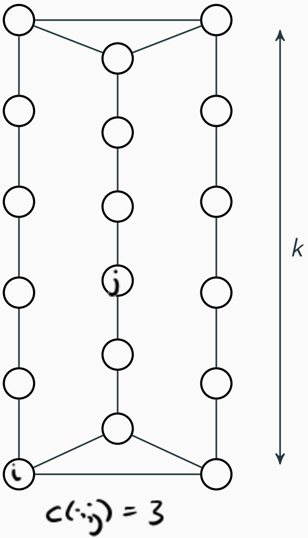
And the four-thirds conjecture can be restated as

Conjecture

There exists an algorithm that returns a solution of cost at most $\frac{4}{3}OPT_{LP}$.

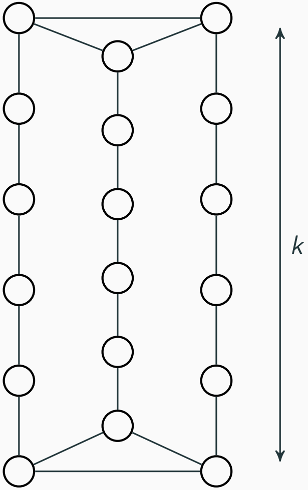
Why 4/3??? Because an example is known that shows we cannot do better (because of the integrality gap of the LP).

Integrality Gap



"Graph-TSP" instance:
$c(i,j)$ is the shortest path distance in unweighted graph.

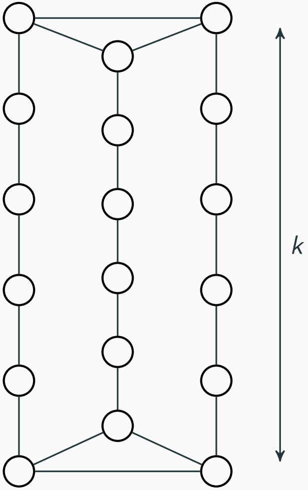
Integrality Gap



"Graph-TSP" instance:
$c(i, j)$ is the shortest path distance in unweighted graph.

Consider the Graph-TSP for this particular graph on $3k$ nodes, what can we say about OPT?

Integrality Gap

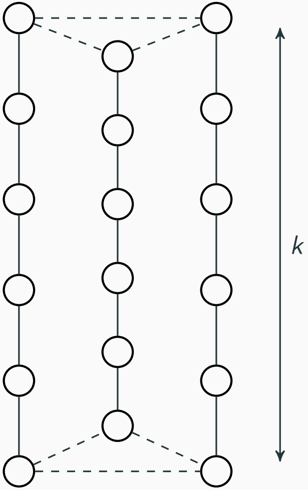


"Graph-TSP" instance:
$c(i, j)$ is the shortest path distance in unweighted graph.

Consider the Graph-TSP for this particular graph on $3k$ nodes, what can we say about OPT?

$OPT \approx 4k$
(in fact, can show that $OPT = 4k - 1$).

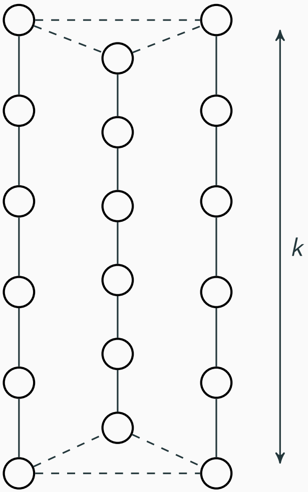
Integrality Gap



Let $x(e) = 1$ for the non-dashed edges, and $x(e) = \frac{1}{2}$ for the dashed edges.

This is a feasible (and optimal) Subtour LP solution!

Integrality Gap

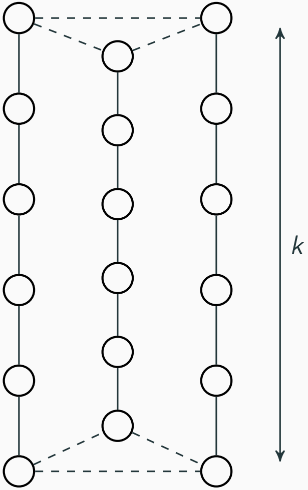


Let $x(e) = 1$ for the non-dashed edges, and $x(e) = \frac{1}{2}$ for the dashed edges.

This is a feasible (and optimal) Subtour LP solution!

So $OPT_{LP} = 3(k-1) + 6(\frac{1}{2}) = 3k$, whereas $OPT = 4k - 1$.

Integrality Gap



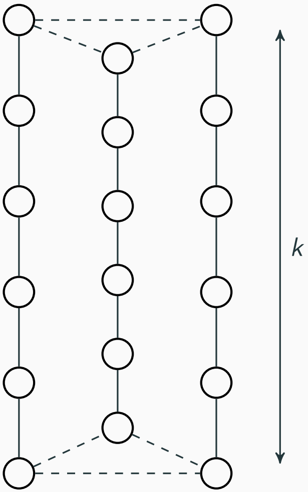
Let $x(e) = 1$ for the non-dashed edges, and $x(e) = \frac{1}{2}$ for the dashed edges.

This is a feasible (and optimal) Subtour LP solution!

So $OPT_{LP} = 3(k-1) + 6(\frac{1}{2}) = 3k$, whereas $OPT = 4k - 1$.

By choosing k large enough, the example shows for any $\epsilon > 0$ that there does not always exist a solution of cost at most $(\frac{4}{3} - \epsilon)OPT_{LP}$.

Integrality Gap



Definition

The integrality gap is $\sup \frac{OPT}{OPT_{LP}}$.

The example shows that the integrality gap is at least $\frac{4}{3}$.

The analysis of the Tree+Matching algorithm shows that the integrality gap is at most $\frac{3}{2}$.

Four-Thirds Conjecture

Let γ be the smallest value such that we can find a tour of cost at most γOPT_{LP}.

Then today we showed:

$$\frac{4}{3} \leq \gamma \leq \frac{3}{2}$$

Four-Thirds Conjecture

Let γ be the smallest value such that we can find a tour of cost at most γOPT_{LP}.

Then today we showed:

$$\frac{4}{3} \leq \gamma \leq \frac{3}{2}$$

Conjecture

$$\gamma = \frac{4}{3}.$$

Recent Developments

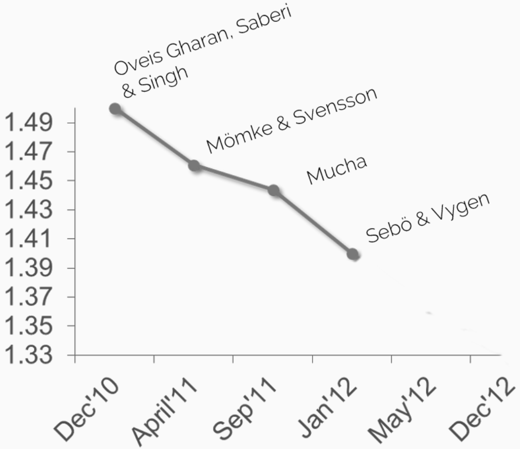
Symmetric TSP

$$\frac{4}{3} \leq \gamma \leq \frac{3}{2}$$

No news to report since 1980s.

Graph-TSP

$$\frac{4}{3} \leq \gamma \leq \frac{7}{5}$$



[3]

[3] Date on x-axis is date the result was first available (arXiv or conference proceedings).

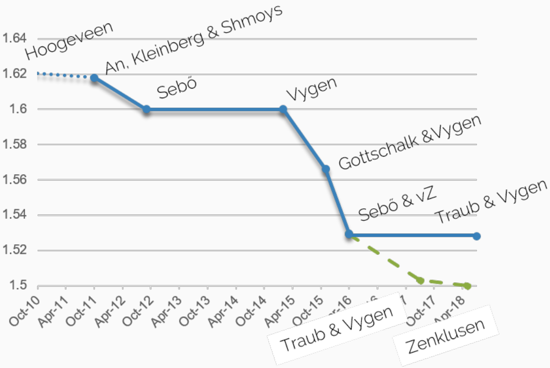
Asymmetric TSP

$$2 \leq \gamma \leq 22$$

- Frieze, Galbiati and Maffioli'82: $\gamma \in O(\log(n))$
- Asadpour, Goemans, Madry, Oveis-Gharan, Saberi'10: $\gamma \in O(\log(n)/\log\log(n))$
- Anari and Oveis-Gharan'15: $\gamma \in O(\mathrm{poly}\log\log(n))$
- Svensson, Tarnawski, Vegh'17: $\gamma \leq 319$
- Traub, Vygen'19: $\gamma \leq 22$

s-t Path TSP

$$\frac{3}{2} \leq \gamma \leq 1.53$$



[4] Date on x-axis is date the result was first available (arXiv or conference proceedings).

Plan for the Lectures

Lecture 2: graph-TSP
Lecture 3: s-t path TSP in graph-TSP instances
Lecture 4: Best-of-Many algorithms for s-t path TSP
Lecture 5: Dynamic programming for s-t path TSP, open questions