Approximation schemes

Traveling salesman problem (TSP)

- inapproximability
- 2-approximation with metric costs
- 1.5-approximation with metric costs (Christofides)

Approximation schemes (subset sum)
Traveling Salesman Problem (TSP)

- **Given** complete graph $K_n$, with edge weights $w : E \to \mathbb{R}_+$
- **Find** cycle $v_1 \to v_2 \to \ldots \to v_n \to v_1$ visiting each vertex exactly once
- **Minimize** weight of cycle
  
  $$w(v_1, v_2) + w(v_2, v_3) + \ldots + w(v_{n-1}, v_n) + w(v_n, v_1)$$

Suppose we have an $\alpha$-approximation. Cost of $OPT$ is $n$. 

Solving exactly is NP-hard.

Approximation?
Traveling Salesman Problem (TSP)

- Solving exactly is NP-hard. Approximation?
- weights on edges

Suppose we have an $\alpha$-approximation. Cost of OPT is $n$.

Any solution that uses at least one expensive edge has cost at least $T$. Choose $T > \alpha n$.

If $\text{cost}(\text{ALG}) \leq \alpha \text{cost}(\text{OPT})$, then no expensive edge is used.

Traveling salesman problem (TSP)

- inapproximability
- 2-approximation with metric costs
- 1.5-approximation with metric costs (Christofides)

- Approximation schemes (subset sum)

Approximate TSP NP-hard in full generality
- Make reasonable assumptions on input!
- Assume that triangle inequality is satisfied:
  \[ w(a, b) \leq w(a, c) + w(c, b) \]
Traveling Salesman Problem (TSP)

- Approximate TSP NP-hard in full generality
- Make reasonable assumptions on input!
- Assume that triangle inequality is satisfied:
  \[ w(a, b) \leq w(a, c) + w(c, b) \]

Examples:
- \( w(a, b) = \) Euclidean distance from \( a \) to \( b \)
- \( w(a, b) = \) shortest path distance in a graph

Inapproximability: \(< 1 + \frac{1}{2} \) not possible unless \( P \neq NP \)

[Papadimitriou-Yannakakis'1993]

TODAY:
- 2-approximation
- 1.5-approximation

2-Approximation for TSP

Main idea:
- Need to find minimum weight cycle, but do not know how
- Compute minimum weight spanning tree instead!
- Convert spanning tree into walk that visits every node \( \geq 1 \) times
- Convert walk into a cycle! (visits every node once)
2-Approximation for TSP

Main idea:
- Need to find minimum weight cycle, but do not know how
  - Compute minimum weight spanning tree instead!
  - Convert spanning tree into walk that visits every node \( \geq 1 \) times
  - Convert walk into a cycle! (visits every node once)

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal
2-Approximation for TSP
- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

Weights on edges:

```
a e d b c b d e a f a g a
```
2-Approximation for TSP

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

Let \( \text{OPT} = \text{cost of minimum TSP tour.} \)

First, 

\[
\text{cost}(MST) \leq \text{OPT}
\]
2-Approximation for TSP

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

Let $OPT =$ cost of minimum TSP tour.

First,

$$\text{cost}(\text{MST}) \leq OPT$$

Q1: Why?

Cost of preorder traversal $\leq 2OPT$

Q2: Are we done?

Shortcutting: skip vertices that were already visited

weights on edges

a e d b c b d e a f a g a

$$w(c, f) \leq w(c, b) + w(b, d) + w(d, e) + w(e, a) + w(a, f)$$
2-Approximation for TSP

Shortcutting: skip vertices that were already visited

weights on edges

\[ w(c, f) \leq w(c, b) + w(b, d) + w(d, e) + w(e, a) + w(a, f) \]

\[ w(f, g) \leq w(f, a) + w(a, g) \]
2-Approximation for TSP

Shortcutting: skip vertices that were already visited

\[ w(f,g) \leq w(f,a) + w(a,g) \]

2-Approximation for TSP (summary)

- Compute minimum spanning tree
- Pick arbitrary root
- Output preorder traversal

Let \( \text{OPT} = \text{cost of minimum TSP tour} \).

First,

\[ \text{cost}(MST) \leq \text{OPT} \]

Cost of preorder traversal \( \leq 2\text{OPT} \)

Cost of final tour only smaller (triangle inequality)

A source of inefficiency: doubling of the tree
1.5-Approximation for TSP

Can we find a small weight subgraph that looks more like a cycle?

Which graphs admit such walks?

Can find a closed walk that visits each edge exactly once!
Can we find a small weight subgraph that looks more like a cycle?

Can find a closed walk that visits each edge exactly once!

Which graphs admit such walks?
1.5-Approximation for TSP

Can we find a small weight subgraph that looks more like a cycle?

Can find a closed walk that visits each edge exactly once!

Which graphs admit such walks?

Euler tours

Euler tour = walk that visits each edge exactly once

Theorem
A graph $G$ admits an Euler tour (is Eulerian) iff the degree of every vertex is even.

Seven bridges of Königsberg

Euler tour = walk that visits each edge exactly once

Seven bridges of Königsberg
**Theorem**
A graph $G$ admits an Euler tour (is Eulerian) iff the degree of every vertex is even.

Our plan:
- compute minimum spanning tree $T$
- add some set of edges $M$ to $T$ so that $T + M$ is Eulerian
- find an Euler tour
- shortcut!

**Step 2 needs to ensure that all vertex degrees are even!**

Add a matching on odd-degree nodes of $T$!
(increases degrees by 1)

---

1.5-Approximation for TSP (Christofides)
- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour

---

1.5-Approximation for TSP (Christofides)
- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour
1.5-Approximation for TSP (Christofides)

- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour

1.5-Approximation for TSP (Christofides)

- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour
1.5-Approximation for TSP (Christofides)

- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)

![Graph 4](image4)
1.5-Approximation for TSP (Christofides)

- Compute MST
- Find min-cost matching on odd-degree vertices
- Find Euler tour of $T + M$
- Shortcut the Euler tour

Theorem

Christofides algorithm gives $3/2$ approximation.

Lemma

Cost of min-cost matching is $\leq OPT/2$

Proof.
Consider TSP tour $C_{odd}$ on the odd nodes only. We have

$$cost(C_{odd}) \leq OPT$$

(1) Odd edges on $C_{odd}$ form a matching and (2) even edges on $C_{odd}$ form a matching.

- Compute MST (cost at most $OPT$)
- Find min-cost matching on odd-degree vertices (cost at most $OPT/2$)
- Find Euler tour of $T + M$ (cost at most $OPT/2$)
- Shortcut the Euler tour (cost can only decrease)

Thus

$$ALG \leq (3/2)OPT$$
1.5-Approximation for TSP (Christofides)

- Compute MST (cost at most \( OPT \))
- Find min-cost matching on odd-degree vertices (cost at most \( OPT/2 \))
- Find Euler tour of \( T + M \) (cost at most \( 3/2OPT \))
- Shortcut the Euler tour (cost can only decrease)

Thus \( ALG \leq (3/2)OPT \)

Approximation schemes

Approximation schemes: approximation algorithm with additional input \( \epsilon \) satisfying that the output is a \( 1 + \epsilon \) approximation

PTAS (polynomial time approximation scheme)
=polynomial time for any fixed \( \epsilon > 0 \)
=\( O(n^{f(\epsilon)}) \) time

EPTAS (efficient PTAS)
=\( O(f(\epsilon) \cdot n^C) \) time
Approximation schemes

**Approximation schemes**: approximation algorithm with additional input \( \varepsilon \) satisfying that the output is a \( 1 + \varepsilon \) approximation.

PTAS (polynomial time approximation scheme) = polynomial time for any fixed \( \varepsilon > 0 \) = \( O(n^{f(\varepsilon)}) \) time

EPTAS (efficient PTAS) = \( O(f(\varepsilon) \cdot n^C) \) time

FPTAS (fully polynomial time approximation scheme) = \( O((n/\varepsilon)^C) \) time

---

Subset sum

**Input**: positive integers \( w_1, w_2, \ldots, w_n \) and positive integer \( t \)

**Output**: index set \( I \subseteq \{1, 2, 3, \ldots, n\} \) such that

\[
\sum_{i \in I} w_i \leq t
\]

**Objective**: maximize

\[
\sum_{i \in I} w_i
\]

---

Dynamic program

\( O(nt) \) time

**Subproblem**: find \( I \subseteq \{1, 2, \ldots, k\} \) (prefix) to maximize \( \sum_{i \in I} w_i \) subject to \( \sum_{i \in I} w_i \leq w \)

**Q**: How many subproblems?

**A**: \( n(t+1) \)
Dynamic program

\( O(nt) \) time

**Subproblem**: find \( I \subseteq \{1, 2, \ldots, k\} \) (prefix) to maximize \( \sum_{i \in I} w_i \)
subject to \( \sum_{i \in I} w_i \leq w \)

**Q**: How many subproblems?
**A**: \( n(t + 1) \)

**Guessing**: is \( k \in OPT? \)

\[
OPT(k, w) = \max\{OPT(k - 1, w), OPT(k - 1, w - w_k) + w_k\}
\]
assuming \( OPT(j, <0) = -\infty \).

**Initialization**:

\( OPT(0,0) = 0 \) and \( OPT(0,>0) = 0 \)

---

FPTAS for subset sum

Let \( M \) denote the maximum item weight:

\[
M := \max_{i=1,\ldots,n} w_i
\]

Define

\[
\bar{w}_i := \lceil w_i \cdot \frac{n}{\epsilon M} \rceil
\]
and

\[
\bar{t} := \lceil t \cdot \frac{n}{\epsilon M} \rceil
\]

---

Pseudopolynomial runtime: \( n(t + 1) \)

(1 + \( \varepsilon \))-approximate solution in \( poly(n, 1/\varepsilon) \) time?

Use approximate weights with a smaller range (rounding!)

---

FPTAS for subset sum

**Subproblem**: find \( I \subseteq \{1, 2, \ldots, k\} \) (prefix) to maximize \( \sum_{i \in I} \tilde{w}_i \)
subject to \( \sum_{i \in I} \tilde{w}_i \leq \tilde{w} \)

**Q**: How many subproblems?
**A**: Don't ask.
Why does this work?

Claim
For any $\varepsilon > 0$ can compute $(1 - \varepsilon)$-approximate solution.

How different are the new weights from old weights?

Let $S \subseteq \{1, 2, \ldots, n\}$.

$$\frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i \leq \sum_{i \in S} w_i \leq \frac{\varepsilon M}{n} \sum_{i \in S} (\tilde{w}_i + 1)$$

So

$$\frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i \leq \sum_{i \in S} w_i \leq \frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i + |S| \frac{\varepsilon M}{n}$$

But $|S| \frac{\varepsilon M}{n} \leq \varepsilon M \leq w(OPT)$, so

\[OPT(k, \tilde{w}) = \max\{OPT(k - 1, \tilde{w}), OPT(k - 1, \tilde{w} - \tilde{w}_k) + \tilde{w}_k\}\]
Why does this work?

Claim
For any $\varepsilon > 0$ can compute $(1 - \varepsilon)$-approximate solution.

How different are the new weights from old weights?

Let $S \subseteq \{1, 2, \ldots, n\}$. 

$$\frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i \leq \sum_{i \in S} w_i \leq \frac{\varepsilon M}{n} \sum_{i \in S} (\tilde{w}_i + 1)$$

So

$$\frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i \leq \sum_{i \in S} w_i \leq \frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i + |S| \varepsilon M$$

But $|S| \varepsilon M \leq \varepsilon M \leq w(OPT)$, so

$$\frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i \leq \sum_{i \in S} w_i \leq \frac{\varepsilon M}{n} \sum_{i \in S} \tilde{w}_i + \varepsilon \cdot w(OPT)$$
Why does this work?

Claim
For any \( \varepsilon > 0 \) can compute \((1 - \varepsilon)\)-approximate solution.

Let \( S \subseteq \{1,2,\ldots,n\} \) denote the output of our algorithm.

\[
\sum_{i \in S} w_i = \frac{\varepsilon M}{n} \sum_{i \in S} \frac{w_i}{\varepsilon M} + 1
\]

\[
\leq \frac{\varepsilon M}{n} \sum_{i \in S} (\frac{w_i}{\varepsilon M} + 1)
\]

\[
\leq \frac{\varepsilon M}{n} \sum_{i \in S} (\bar{w}_i + 1)
\]

\[
\leq \frac{\varepsilon M}{n} \left( \sum_{i \in S} \bar{w}_i \right) + \frac{\varepsilon M}{n} |S|
\]

\[
\leq \frac{\varepsilon M}{n} t + \varepsilon M
\]

\[
\leq \frac{\varepsilon M}{n} t + \frac{\varepsilon M}{n} + \varepsilon M \leq (1 + \varepsilon)t
\]

So the set that our algorithm returns is approximately feasible.

What about the weight?

Let \( S \subseteq \{1,2,\ldots,n\} \) denote the output of our algorithm.

\[
\frac{\varepsilon M}{n} \sum_{i \in S} \bar{w}_i \geq \frac{\varepsilon M}{n} \sum_{i \in S} \frac{n}{\varepsilon M}
\]

But

\[
\frac{\varepsilon M}{n} \sum_{i \in S} \bar{w}_i = \frac{\varepsilon M}{n} \sum_{i \in S} \frac{n}{\varepsilon M} \leq \sum_{i \in S} w_i
\]

And

\[
\frac{\varepsilon M}{n} \sum_{i \in S} \bar{w}_i \geq \frac{\varepsilon M}{n} \sum_{i \in S} (\frac{w_i}{\varepsilon M} - 1) \geq (1 - \varepsilon)w(OPT)
\]

So

\[
\sum_{i \in S} w_i \geq (1 - \varepsilon) \sum_{i \in OPT} w_i
\]

Euclidean TSP

PTAS due to Arora'96:
- \( O(n(\log n)^{O(1/\varepsilon)}) \) time for planar TSP
- \( O(n(\log n)^{O(\sqrt{\varepsilon})^{d-1}}) \) in dimension \( d \)