CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Approximation Algorithms 2: Traveling Salesman

Instructor: Xiao Liang[1]

Department of Computer Science and Engineering
The Chinese University of Hong Kong

[1]These slides are primarily based on materials prepared by Prof. Yufei Tao (please refer to Prof. Tao's version from 2024 Fall for the original content). Some modifications have been made to better align with this year's teaching progress, incorporating student feedback, in-class interactions, and my own teaching style and research perspective.

Definition

Setup for the Traveling Salesman Problem (TSP):

- $G = (V, E)$ is a complete[2] undirected graph.
- Each edge $e \in E$ carries a non-negative **weight** $w(e)$.
- A **Hamiltonian cycle** of G is a cycle passing every vertex of V once.

The traveling salesman problem: Find a Hamiltonian cycle with the shortest length.

[2] This is without loss of generality: If there is no "real" direct path between two vertices, we can assign a very large cost (or use a placeholder like ∞) to discourage that route—but the edge still "exists" in the graph.

An Example of TSP

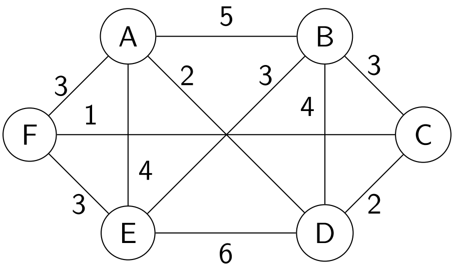


Figure: An Exemplary Graph (missing edges have weight ∞)

The shortest Hamiltonian cycle for the above graph is:

$$A \to D \to C \to B \to E \to F \to A$$

The cost is: $2 + 2 + 3 + 3 + 3 + 3 = 16$

Application: Logistics and Transportation

Courier and Delivery Services

- Companies like UPS, FedEx, and food delivery services use TSP-like optimizations to minimize travel distance and fuel costs.

Ride-Sharing Services

- Apps like Uber and DiDi solve routing subproblems similar to TSP to efficiently match passengers with drivers.

Waste Collection and Street Sweeping

- Cities optimize routes for municipal services to reduce time and operational costs.

Application: Manufacturing and Robotics

Robotic Path Planning

- Robots performing inspections or assembly tasks use TSP to find efficient routes through multiple checkpoints.

3D Printing

- Optimizing the nozzle path to reduce print time and material waste.

Application: Biology and Data Analysis

DNA Sequencing

- In DNA sequencing, there is a famous problem called the Shortest Common Superstring (SCS) Problem. This problem can be modeled as a special version of TSP.

Astronomy and Telescope Scheduling

- Scheduling observations of multiple celestial bodies in the most efficient order.

Clustering and Visualization

- TSP can help in ordering data points for better visualization, such as in heatmaps.

The problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in $|V|$.
- Such algorithms cannot exist if $\mathcal{P} \neq \mathcal{NP}$.

So, let's focus on approximate solutions as usual.

Approximation Algorithm for TSP?

$\mathcal{A}$ = an algorithm that, given any legal input (G, w), returns a Hamiltonian cycle of G.

Denote by $OPT_{G,w}$ the shortest length of all Hamiltonian cycles of G under the weight function w.

> $\mathcal{A}$ is a **ρ-approximate algorithm** for the traveling salesman problem if, for any legal input (G, w), $\mathcal{A}$ can return a Hamiltonian cycle with length at most $\rho \cdot OPT_{G,w}$.

The value ρ is the **approximation ratio**.

Bad news (the proof of it is out of the scope of this course):

- In fact, **TSP is NP-hard to approximate within any constant factor.** That is, achieving a constant ρ is no easier than solving the exact TSP!

TSP with triangle inequality

We will instead focus on graphs with nicer structure.

We additionally assume that the graph G satisfies **triangle inequality**:

- For any $x, y, z \in V$, it holds that $w(x, z) \leq w(x, y) + w(y, z)$.

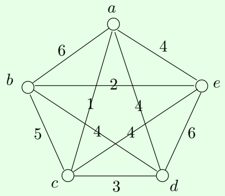
Motivation:

- Ensures that taking a *detour* is never shorter than going directly.
- Reflects realistic distance metrics (e.g., Euclidean distances, road networks).

Our goal now is to find an approximation algorithms $\mathcal{A}$ for TSP on such graphs satisfying triangle inequality.

- We will show such an algorithm for $\rho = 2$.

Exemplary Graph satisfying Triangle Inequality



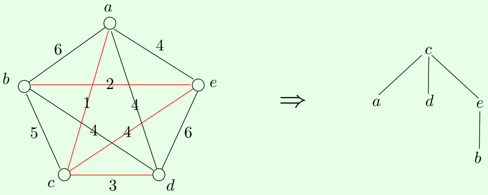
An optimal solution: *acdbea* with length 14.

Algorithm

Next, we will describe a 2-approximate algorithm. The algorithm consists of 3 steps.

Step 1: Obtain an MST (minimum spanning tree) T of G.

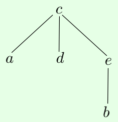
Example:



Algorithm

Step 2: Obtain a **closed walk** π where every edge of T appears on π exactly **twice**.

Example:



A possible closed walk: π = cacdcebec

π can be obtained in $O(|V|)$ time (regular exercise). [Hint: modify DFS.]

Algorithm

Step 3: Construct a sequence σ of vertices as follows. First, add the first vertex of π to σ. Then, go through π; when arriving at a vertex v:

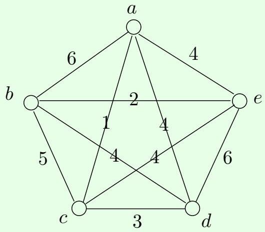
- If v has not been seen before, append v to σ.
- Otherwise, do nothing.

Finally, add the last vertex of π to σ.

The sequence σ now gives a Hamiltonian cycle.

Return this cycle.

Example:



$\pi =$ *cacdcebec*
$\sigma =$ *cadebc*
Weight of the Hamiltonian cycle: 18

Theorem 1: Our algorithm returns a Hamiltonian cycle with length at most $2 \cdot OPT_{G,w}$.

Next, we will prove the theorem.

Let $w(T)$ be the **weight** of (the MST) T:

$$w(T) = \sum_{\text{edge } e \text{ in } T} w(e)$$

Lemma 1: $OPT_{G,w} \geq w(T)$.

Proof: Given any Hamiltonian cycle, we can remove an (arbitrary) edge to obtain a spanning tree of G. The lemma follows from the fact that T is an MST. □

Next, we will show that our Hamiltonian cycle σ has length at most $2 \cdot w(T)$, which will complete the proof of Theorem 1.

Lemma 2: The walk π has length $2 \cdot w(T)$.

Proof: Every edge of T appears twice in π. □

Lemma 3: The length of our Hamiltonian cycle σ is at most the length of π.

Proof: Let the vertex sequence in π be $u_1 u_2 ... u_t$ for some $t \geq 1$.
Let σ be the vertex sequence $u_{i_1} u_{i_2} ... u_{i_{|V|+1}}$ where

$$i_1 = 1 < i_2 < ... < i_{|V|} < i_{|V|+1} = t.$$

By triangle inequality, we have for each $j \in [1, |V|]$:

$$w(u_{i_j}, u_{i_{j+1}}) \leq \sum_{k=i_j}^{i_{j+1}-1} w(u_k, u_{k+1})$$

Hence:

$$\text{length of } \sigma = \sum_{j=1}^{|V|} w(u_{i_j}, u_{i_{j+1}}) \leq \sum_{k=1}^{t-1} w(u_k, u_{k+1}) = \text{length of } \pi.$$

□