Approximation Algorithms 2: Traveling Salesman

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

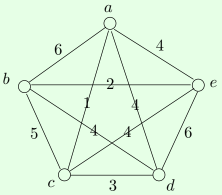
$G = (V, E)$ is a complete undirected graph.

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.

G satisfies **triangle inequality**: for any $x, y, z \in V$, it holds that $w(x, z) \le w(x, y) + w(y, z)$.

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



An optimal solution: *acdbea* with length 14.

The problem is NP-hard.

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

$\mathcal{A}$ = 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 $\rho \cdot OPT_{G,w}$.

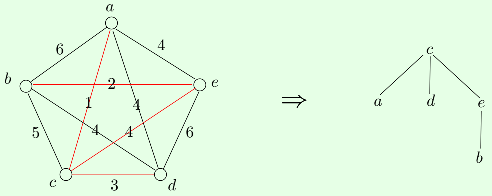
The value ρ is the **approximation ratio**.
We say that $\mathcal{A}$ achieves an approximation ratio of ρ.

Algorithm

Next, we will describe a 2-approximate algorithm.

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

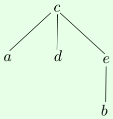
Example:



Algorithm

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

Example:



A possible closed walk: $\pi =$ cacdcebec

π can be obtained in $O(|V|)$ time (regular exercise).

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 crossing an edge (u, v):

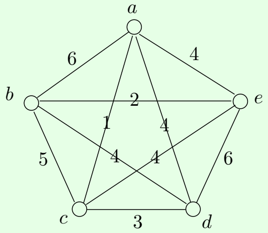
- If v has not been seen before, append v to σ.
- Otherwise, do nothing.

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:

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

□