Lab 2 – Traveling Salesman Problem
Advanced Algorithms

Master in Computer Science

Davide Bresolin

2021/2022

Traveling Salesman Problem

Traveling Salesman Problem (TSP)

Input: undirected, complete and weighted graph $G = (V, E)$ with weights $c : V \times V \mapsto \mathbb{N}$

Output: shortest Hamiltonian cycle in G

Approximate TSP: two approaches

- A 2-approximate algorithm based on MST, that guarantees a good approximation for metric TSP
- a family of constructive heuristics that uses a greedy approach to find a solution, with different guarantees on the quality of the approximation

TSP: constructive heuristics

- A large family of heuristics that build the solution one vertex at a time following the same general scheme:
  1. **Initialization:** how to chose the initial circuit (or starting vertex);
  2. **Selection:** how to chose the next vertex to be inserted in the solution;
  3. **Insertion:** how to chose the position where to insert the new vertex.

Nearest Neighbor

1. **Initialization:** start from the single-node path 0
2. **Selection:** let $(v_0, \ldots, v_k)$ be the current path. Find the vertex v_{k+1} not in the path with minimimun distance from v_k;
3. **Insertion:** insert v_{k+1} immedaitely after v_k ;
4. repeat from (2) until all vertices are inserted in the path.

Random Insertion

1. **Initialization:** start from the single-node path 0. Find the vertex j that minimize $w(0, j)$ and build the partial circuit $(0, j)$;
2. **Selection:** randomly select a vertex k not in the circuit;
3. **Insertion:** find the edge $\{i, j\}$ of the partial circuit that minimize $w(i, k) + w(k, j) - w(i, j)$ and insert k between i and j;
4. repeat from (2) until all vertices are inserted in the path.

Cheapest Insertion

1. **Initialization:** start from the single-node path 0. Find the vertex j that minimize $w(0, j)$ and build the partial circuit $(0, j)$;
2. **Selection:** find a vertex k not in the circuit and an edge $\{i, j\}$ of the circuit that minimize $w(i, k) + w(k, j) - w(i, j)$;
3. **Insertion:** insert k between i and j;
4. repeat from (2) until all vertices are inserted in the path.

Circuit-vertex distance

Given a set of vertices $C \subseteq V$, and a vertex $k \notin C$, we define the distance of k from C as the minimum weight of the edges connecting k to C:

$$\delta(k, C) = \min_{h \in C} w(h, k)$$

"Closest Insertion" and "Farthest Insertion" select vertices that minimize/maximizes the distance from the circuit

Closest Insertion

1. **Initialization:** start from the single-node path 0. Find the vertex j that minimize $w(0, j)$ and build the partial circuit $(0, j)$;
2. **Selection:** find a vertex k k not in the circuit C that minimize $\delta(k, C)$;
3. **Insertion:** find the edge $\{i, j\}$ of the partial circuit that minimize $w(i, k) + w(k, j) - w(i, j)$ and insert k between i and j;
4. repeat from (2) until all vertices are inserted in the path.

Farthest Insertion

1. **Initialization:** start from the single-node path 0. Find the vertex j that maximize $w(0, j)$ and build the partial circuit $(0, j)$;
2. **Selection:** find a vertex k k not in the circuit C that maximize $\delta(k, C)$;
3. **Insertion:** find the edge $\{i, j\}$ of the partial circuit that minimize $w(i, k) + w(k, j) - w(i, j)$ and insert k between i and j;
4. repeat from (2) until all vertices are inserted in the path.

Approximation factors

If the triangular inequality is respected:

- Nearest Neighbor, Random Insertion and Farthest Insertion give a $\log(n)$-approximation
- Closest Insertion and Cheapest Insertion find a 2-approssimation