Traveling salesman Problem

TSP: Introduction

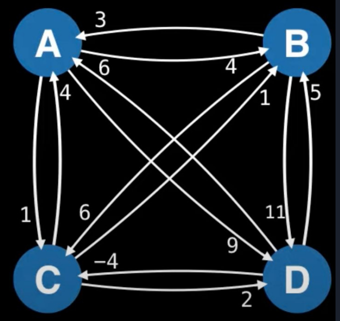
Let L=∑ ∑ x(i, j) d(i, j); We want to find:

$$\mathbf{L^* = \min \sum \sum x(i, j)\, d(i, j)}$$

In other words - given a complete graph, find the shortest **Hamiltonian cycle** in the graph.

NP-complete - no known efficient (i.e., polynomial time) algorithm.

$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{bmatrix} 0 & 4 & 1 & 9 \\ 3 & 0 & 6 & 11 \\ 4 & 1 & 0 & 2 \\ 6 & 5 & -4 & 0 \end{bmatrix} \end{array}$$



Proposed implementation of algorithms

Naive solution/Dynamic Programming

1. *(Exact) Naive: Brute force search over N!*
2. (YZ)(Exact) Held - Karp (Dynamic Programming) O(n^2 2^n)
3. (TY) (Approximate) Nearest Neighbour
4. (SW)(Approximate) Lin - Kernighan
5. (WZ) (Approximate) Christofides - 1.5* L*
6. (NS) (Approximate) Simulated Annealing
7. (NS) (Approximate) Ant colony optimization

Algorithm Main Ideas

1. **Naive - $O(n!)$**

   Try all permutations and see which one is the shortest path (using brute force search).

2. **Held-Karp - $O(n^2 2^n)$**

   Based on dynamic programming and solve TSP “bottom-up”.

   $G(i, s) = \min \{ C_{ik} + G(i, s-\{k\}) \}$

3. **Nearest Neighbour**

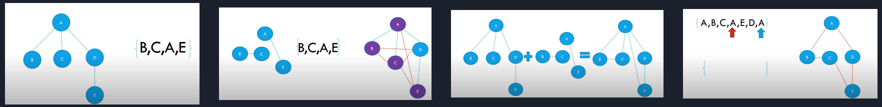
   One of the first algorithms solves TSP approximately. The salesman starts at a random city and repeatedly visits the nearest city until all have been visited.

Algorithm Main Ideas

1. **(Approximate) Lin - Kernighan O(N^2.2)**

   Lin–Kernighan is one of the best heuristics for solving the symmetric travelling salesman problem. It belongs to the class of local search algorithms.

2. **(Approximate) Christofides** - 1.5* L*: If the cities follow the triangle inequality, which is a+b>c, that this algorithm has 3/2 approximation guarantee.



3. **(Approximate) Simulated Annealing** - arrive at solution by (~exhaustively) exploring the state space
   - Random, iterative improvements to the path length
   - Always accept improvements; decaying probability of accepting setbacks to encourages state exploration
4. **(Approximate) Ant colony optimization** - based on behavior of real-life ant colonies
   - Randomly explore space, leaving “pheromones” to help later ants find good paths

Project Design Requirements

Program language: C++, Python (Networkx for visualization).

Dataset: TSPLIB

Test plan: Use (some) of the proposed algorithms to generate approximate solutions to various TSP datasets.

- How far from the optimal are the resulting paths?
- Performance of each algorithm?
- Statistics for the randomized algorithms
- Agreement with theoretical upper and lower bounds?

References:

For some introductory reading:

https://en.wikipedia.org/wiki/Travelling_salesman_problem
https://en.wikipedia.org/wiki/Christofides_algorithm

Detailed Algorithmic Implementations

“Simulated Annealing”, Per Brinch Hansen June 1992
Ant Colony Optimization, Marco Dorigo and Thomas Stützle

Dataset Documentation

https://www.math.uwaterloo.ca/tsp/data/index.html
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Thank You