The Long Road:
Comparison of Approximation Heuristics for the Traveling Salesperson Problem

Arthur Viegas Eguia, Artem Yushko, Miles Hoene-Langdon, Daniel Chen, Layla Oesper
Carleton College

Problem Formulation

Given a set of cities, and the distance between each pair of cities, what is the smallest distance that a salesperson has to travel so they can visit each city once and return to the origin city [1][2]

Problem details

- There are no efficient known ways to solve the problem (NP-hard)
- Heuristics approximate solutions that are "good enough"
- Commonly a geographical problem, but also has applications in genetics, transportation networks and other fields [1][2]
- Cities are nodes of a graph
- Path connecting two cities are weighted edges (weight = distance)

Heuristics

Nearest Neighbor

- Start at any city -> go to the nearest unvisited city -> Repeat until all cities are visited
- Given a networkx graph, grab the first node.
  - Add it to the visited set
  - While the visited set does not contain all nodes
  - sort the edges of the node you are currently at
  - pick the smallest unvisited edge
- Time complexity of $O(n^2 \log n)$
- Easy to implement, relatively fast, suboptimal solutions

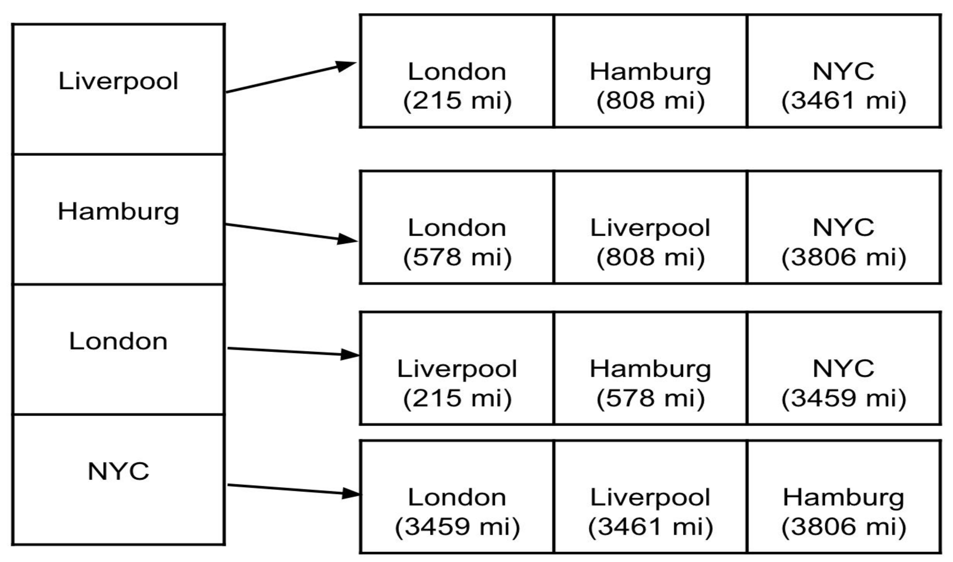


Figure 1: Nearest Neighbor example

- Liverpool -> London -> Hamburg -> NYC -> Liverpool (8060 mi)

Smallest Insertion

- Get two random cities, add a third city in the tour so that the distance increases the least. Repeat until all cities are visited
- Time complexity of $O(n^2)$

Christofides

- Relatively state of the art algorithm for the problem. Solution is guaranteed to be at most 1.5 times worse than optimal
  - Requires that the dataset follows the triangle inequality
- Uses Minimum Spanning Trees, does perfect matching and calculates the Eulerian path
- Runs in $O(n^3)$ [1]

Results

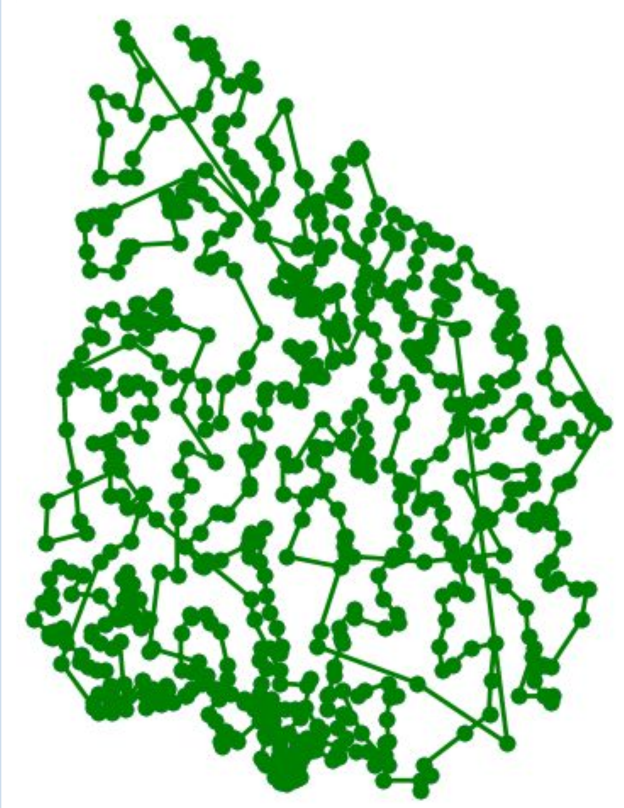


Figure 2: Tour approximation generated by nearest neighbor

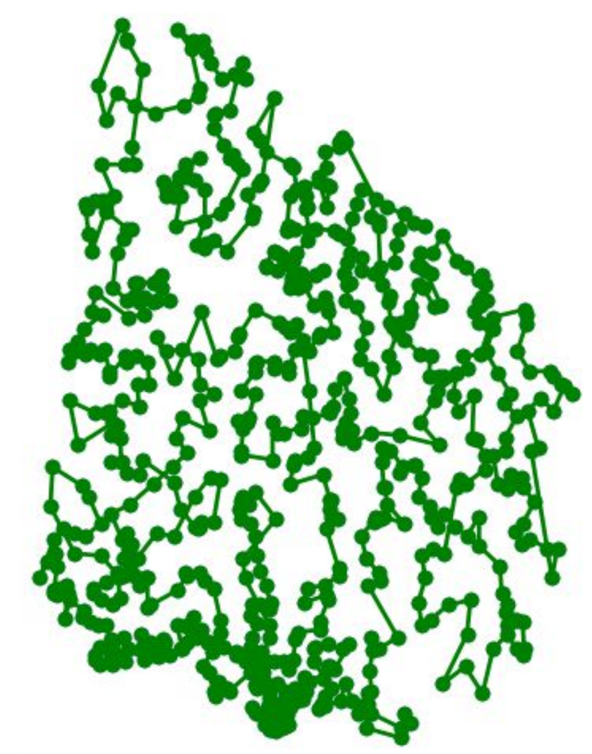


Figure 3: Tour approximation generated by Christofides

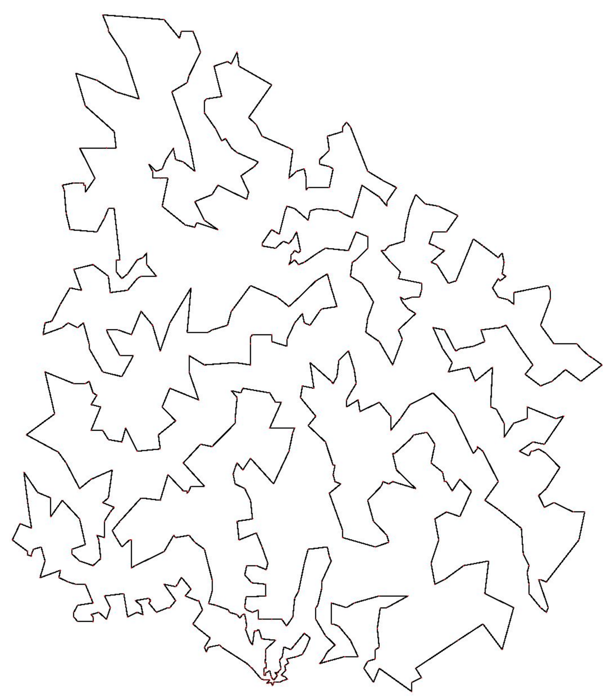


Figure 4: Optimal Path [3]

- The nearest neighbor tour is 99,247mi (25% worse than optimal) [3]
- The Christofides tour is 86,597mi (9% worse than optimal) [3]
- Both tours generated have 282 edges in common (38.419%)
- The largest path they have in common is of size 13

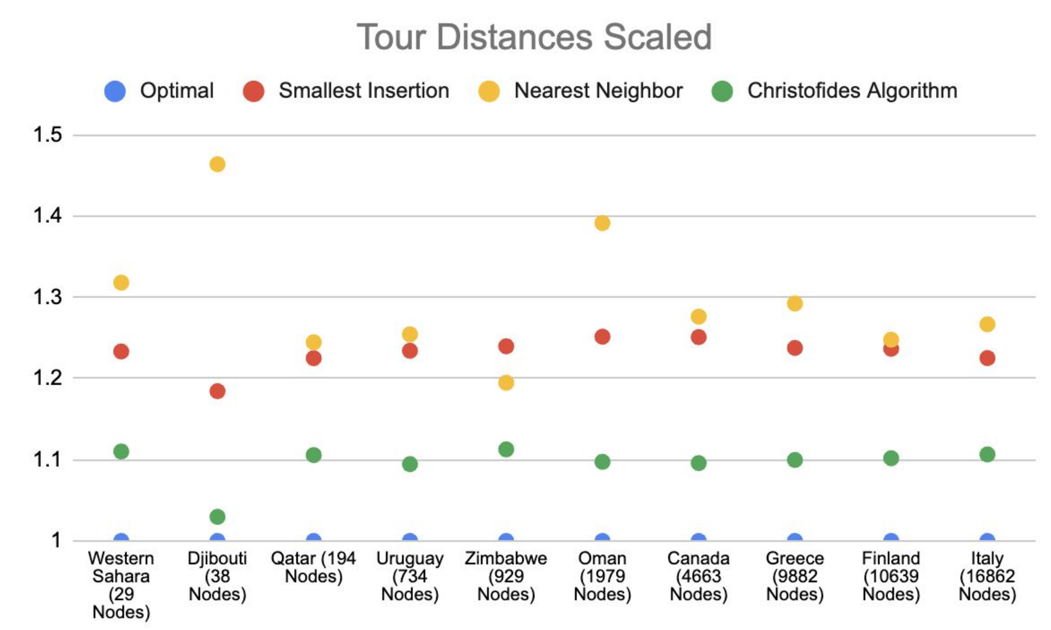
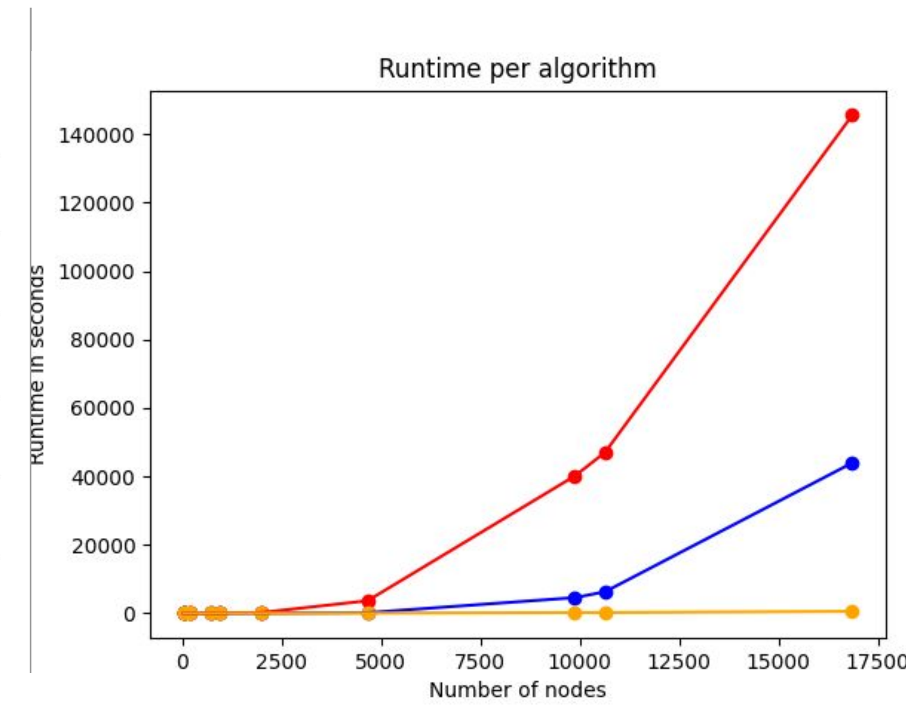


Figure 5: How far is each algorithm from the optimal solution per dataset



- All algorithms had a similar run time for small datasets (up to 200 nodes)
- Nearest Neighbor and Smallest Insertion heuristics had a similar run time up to ~4500 nodes
- Christofides became slow at ~10000 nodes
- Nearest neighbor became slow at ~16000 nodes
- Slowest computation with Christofides took over 40 hours, while nearest neighbor took only 12
- Christofides is consistently the best algorithm, around 10% worse than optimal
  - This beats the theoretical bound of being at most 50% worse than optimal
- Nearest neighbor and Smallest insertion have a similar performance, with smallest insertion being marginally better

Datasets

Input

- A set of points S = (x, y), where each point in the set represents the coordinates of a city.

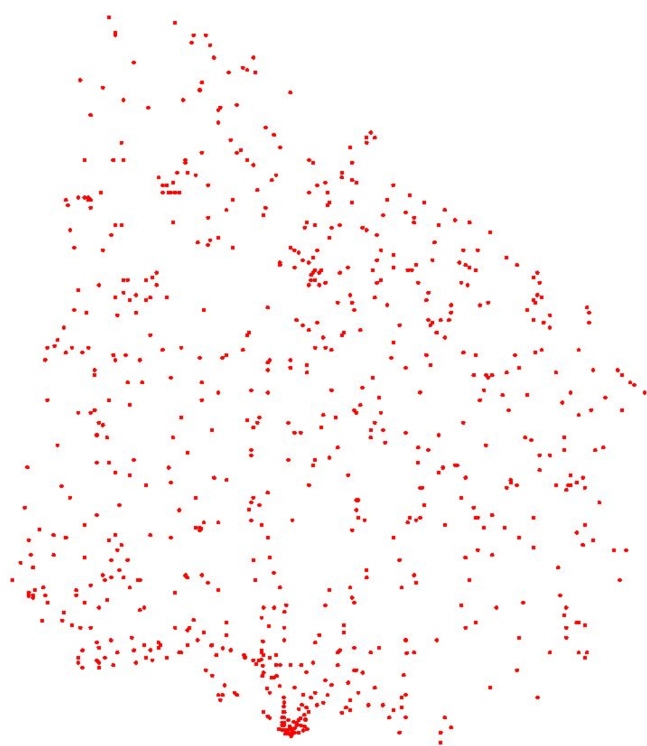


Figure 4: Set of points represented by Coordinates [3]

Output

- An ordered set $G = (p_1, p_2, p_3, \dots, p_{n-1}, p_n, p_1)$ where each p_i is a city, and the total length of G is minimized

Dataset

- 10 geographic datasets ranging from 29 up to 16862 nodes were used. Also included protein and electrical grid datasets
- Given a set of (x, y) coordinate points
  - For every point
    - Add point to a HashSet
  - Match every point to every other point
    - Calculate the distance between them (in miles)
  - Create networkx object
  - Add the recently calculated edges to the graph

Discussion

- The algorithm is very resource intensive
  - The Italy dataset takes from 45-65GB of RAM to run
  - Greece and Finland take around 15-25GB of RAM to run
- While Christofides got the best results, it is significantly slower
  - Might not be the best choice for larger datasets
- Computation time and usage tradeoff
  - Is the tour/nodes are going to change frequently
  - What device is it running on

Sources

[1] Wikimedia Foundation. (2024c, December 29). Travelling salesman problem. Wikipedia. https://en.wikipedia.org/wiki/Travelling_salesman_problem
[2] GeeksforGeeks. (2024b, November 26). Travelling salesman problem using Dynamic Programming. https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/
[3] UWaterloo. (2022, February 8). National traveling salesman problems. UWaterloo. https://www.math.uwaterloo.ca/tsp/world/countries.html

March 6th, 2025

Department of Computer Science

Correspondence: viegaseguiaa@carleton.edu