Analysis of Algorithms
Final Project

Andrés Calderón, Ph.D.

April 29, 2025

1 Introduction

This project focuses on the development of a system designed to optimize the visitation order of a set of locations within a defined road network. The system will include a proof-of-concept web application for loading network and location data, applying different algorithmic approaches to solve the routing problem, and visualizing the results. Additionally, the project requires the preparation of a technical report analyzing the performance and theoretical characteristics of the implemented algorithms.

2 Problem Statement and Scope

The local government, through the IT Office, is issuing a call for proposals to implement a system that enables the upload of a file representing the current road network for a specific area, along with a set of locations within that area. The goal is to determine the optimal order in which the locations should be visited to minimize time and distance as much as possible. The IT Office has identified this as an ideal use case for the well-known Traveling Salesman Problem (TSP).

The call seeks proposals capable of implementing at least two different approaches, in addition to the base case scenario (brute-force algorithm), to enable a well-informed decision on which algorithm to implement in the subsequent phase of the project. The IT Office expects a well-organized report containing the algorithmic analysis of the three options, along with a proof-of-concept web application that allows for the loading of input files and the visualization of the results.

The specific implementation details are discussed in the following section.

3 Use Case Specifications

3.1 Road Network Load

- Pre-condition: The user has a valid representation of the road network (nodes and edges).
- Post-condition: The road network is visible on the web map.

Basic Flow:

1. The user presses a button to select the input file from disk.
2. The system processes the input file and is able to identify nodes and edges.
3. The system draws the network on the web map using points and lines.

3.2 Pointset Load

- Pre-condition: The user has a valid representation of the pointset (ID, latitude, and longitude).
- Post-condition: The updated version of the road network, with the points integrated into it, is visible on the web map.

Basic Flow:

1. The user presses a button to select the input file from disk.
2. The system processes the input file and is able to identify the points.
3. For each point, the system finds the closest edge of the road network using the perpendicular distance.
   (a) The system computes the intersection where the point should be integrated into the network.
   (b) The system updates the network by adding the point as a new node and splitting the corresponding edge.
4. The system visualizes the updated network, showing the new nodes with a different shape and color.

3.3 Algorithms Evaluation

- Pre-condition: The system has a valid network representation and the set of points integrated into it. The system is able to extract the set of points from the network.
- Post-condition: The system displays the resulting path for each algorithm in a different color, along with information about the length of the path and the response time.

Basic Flow:

1. The user presses a button to start the evaluation.
2. The system runs the implementation of the three algorithms one by one, keeping track of the execution time.
3. The system draws the resulting paths over the network in such a way that it is easy to identify each of them.
4. The system displays the length of each path and the execution time of each alternative to the user.

Special requirements:

- The user should be able to download the update of the network and the resulting paths in a valid format.

4 General Requirements

- The IT Office recommends the use of open-source web mapping libraries such as Leaflet[1] or OpenLayers[2], but the technical decisions regarding the implementation are open.
- The following reports collect different approaches and pseudocode for the TSP:
  - "Different Approaches to Travelling Salesman Problem" by Roger Nogales Giné. https://diposit.ub.edu/dspace/bitstream/2445/186678/3/tfg_nogales_gine_roger.pdf.
  - "Different Approaches to Solve Traveling Salesman Problem" by Revant Kumar, Xin Wei, and Aashu Singh. https://hal.science/hal-04348868v1/document.

  However, these reports explore the problem using Euclidean distance between points and not the shortest path over a network, as required by this call.
- It is expected that the outputs for both the algorithm and implementation use cases be supported by an appropriate set of unit tests (i.e., through a unit testing process).
- The implementation must be accompanied by a well-structured report that includes the asymptotic analysis of the three alternatives and an empirical analysis of their performance using random synthetic data of varying sizes.

5 Additional Resources

Useful resources to explore and understand the details of TSP are:

- Traveling Salesman Problem by Technical University of Munich. https://algorithms.discrete.ma.tum.de/graph-games/tsp-game/index_en.html.
- TSP Game by University of Vienna. https://prolog.univie.ac.at/tspGame/game.html
- The TSPLIB library by University of Heidelberg. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html

6 Conclusions

This project outlines the development of a system capable of solving a real-world routing problem through the integration of multiple algorithmic approaches. By enabling the loading, processing, and visualization of road networks and location data, the proposed solution supports a comprehensive evaluation of different methods for route optimization. The inclusion of both theoretical and empirical analyses ensures a well-rounded assessment of the algorithms' performance across various data sizes and conditions. The deliverables: **a proof-of-concept web application** and **a detailed technical report**, will provide a strong foundation for selecting the most suitable approach for future implementation phases.

[1] https://leafletjs.com/

[2] https://openlayers.org/