Deepankar Singh, Tanaya Chitaldurg, Mohinder Goyal, Mehul Zawar, Shruti Gupta, Shikhar Jamuar

Krannert School of Management

Abstract

Determining shortest path to reach from one city to another in USA.

Final project

MGMT 590 – COMPUTING FOR ANALYTICS

**TABLE OF CONTENTS**

Introduction Pg. 2

Computational Setup/ Steps Pg. 3

Results Pg. 4

Conclusion Pg. 5

References Pg. 6

1. **INTRODUCTION –**

Through this project, we are determining the best path that can be taken to reach from one city of the USA to another city of the USA. The model calculates all possible paths that can be taken to reach a Destination city from an Origin city and then returns an optimum path with shortest distance. This is useful for a transportation companies that deliver posts, goods, and even people from one city to another. For example, USPS can use this model to find the shortest path to deliver posts, E-Commerce companies can use it to connect sellers with buyers and provide estimated time frame by which the product can be delivered. Travel and Tourism companies can use this model to design tour packages in effective costs.

Computing shortest paths over a network is an important task in many network and transportation related analyses. There are numerous studies and research ongoing to determine algorithms involving shortest path involving real road networks. In recent studies, a set of three shortest path algorithm that run fastest on real road networks has been identified. These three algorithms are: 1) the graph growth algorithm implemented

with two queues, 2) the Dijkstra algorithm implemented with approximate buckets, and 3) the Dijkstra algorithm implemented with double buckets.

In this project, we have focused on finding shortest path and reduced the time taken by the algorithm to search and report the same. Average time taken to process the data, calculate the shortest path and plot visualization is 21 seconds. To reduce the time complexity, we have made use of different data structures and mathematical logics without impacting the output. We can visualize the shortest distance when input for Origin and Destination city are given.

1. **COMPUTATIONAL SETUP/ STEPS –**

The project was carried out in a logical and sequential manner. The computational steps are given below:

* Preprocessing the Data – We obtained the list of cities in U.S.A. and their latitude and longitude details. The file had the details of 28,889 cities, their longitude and latitude location. We created a random subset of 1600 cities on which we built our model. For every iteration we generate a new subset of cities.
* Creation of routes – Since we did not have the data on different routes from one city to another, we randomly created routes from one city to another. Each city is connected to minimum of 2 other cities and maximum of 5 other cities except itself. We have stored the output in sparse matrix with 0 denoting “no route exist” and a number denotes the distance between row i and column j. After doing this we realized that route from A🡪B did not have same distance as route from B 🡪A in order to fix this we added the transpose of the matrix to the matrix itself and then subtracted twice diagonal.

Anew = A+AT-2D

* Calculation of distance between two geographical points - Distance has been calculated between two cities where the routes are created. Library geopy.distance is used to calculate the straight line distance between two cities using their longitude and latitude data point. The output data is stored in Dictionary format. Storing data in dictionary made it easier to input in algorithm and sustainably reduced the retrieval and processing time.
* Defining function for algorithm – We have defined a function to calculate the optimum path. The inputs for this function are data in dictionary format, initial point, destination point. We used Dijkstra algorithm to compute the shortest path. The faction we defied goes through all the possible paths to travel and the most optimum path is stored in tuple for visualization.
* Visualization of shortest path – All the cities are plotted on the map using folium library based on their geographic coordinates in grey dots. The initial point and destination point are marked in red. Using output from the optimum path function a line is drawn from the from the origin city through intermediate cities to the destination city. For making the map readable, all the cities from where the route passes through are labelled in bold.
* Plotting time complexity through various phases of the project – We have divided the entire project into 3 phases – Data Processing (Data Pre-Processing 🡪 Creation of routes🡪 Calculation of Distance), Optimization( Shortest route calculation ) and Visualization. For the three phases we have calculated and compared time complicity. The comparison is show using a bar chart.

**Dijkstra’s Algorithm:**

Dijkstra’s algorithm is a greedy algorithm to find the shortest path between two nodes in a graph.

Following are the steps of the algorithm:

Let distance of start node from start vertex be 0. Call the starting node as the current node.

Consider all nodes to be in an unvisited list.

Repeat the following steps (until no unvisited node left in the neighbor of the current node or we have reached the end node)

• Visit the unvisited node with the smallest known distance from the start node. Call that current node. Put starting node in a visited list.

• For the current node examine the unvisited neighbors (i.e. the nodes connected to the start node).

• For current node, calculate the distance of each neighbor from the start node. Using the distance of its connecting current node (from the start node) in addition to the distance between current node to the neighbor.

• If the calculated distance of a node is lesser than its current stored distance from the start node, update the distance.

• Put the current node in the visited list and then make the neighbor with the smallest distance as the current node.

**Computational challenge? How did we overcome?**

The most time taking part of the algorithm was route optimization. Using depth or breath first algorithm was not very effitinet for our paticular case. If the number of nodes are very high the processingtime for these algorithms bevcome very high.

Using Dijkstra’s algorithm reduces time complexity to some extenet but here also if the number fo nodes start exceeding 2000 we observe a slight in the processing time. Further, Dijkstra’s algorithm has a unique way of data storage, it not only stores the shortest distance but also stores the previous node. We can say Dijkstra’s uses memoization and hence is much more effecttive as compared to depth or breath fisrt algorithm.

1. **RESULTS –**

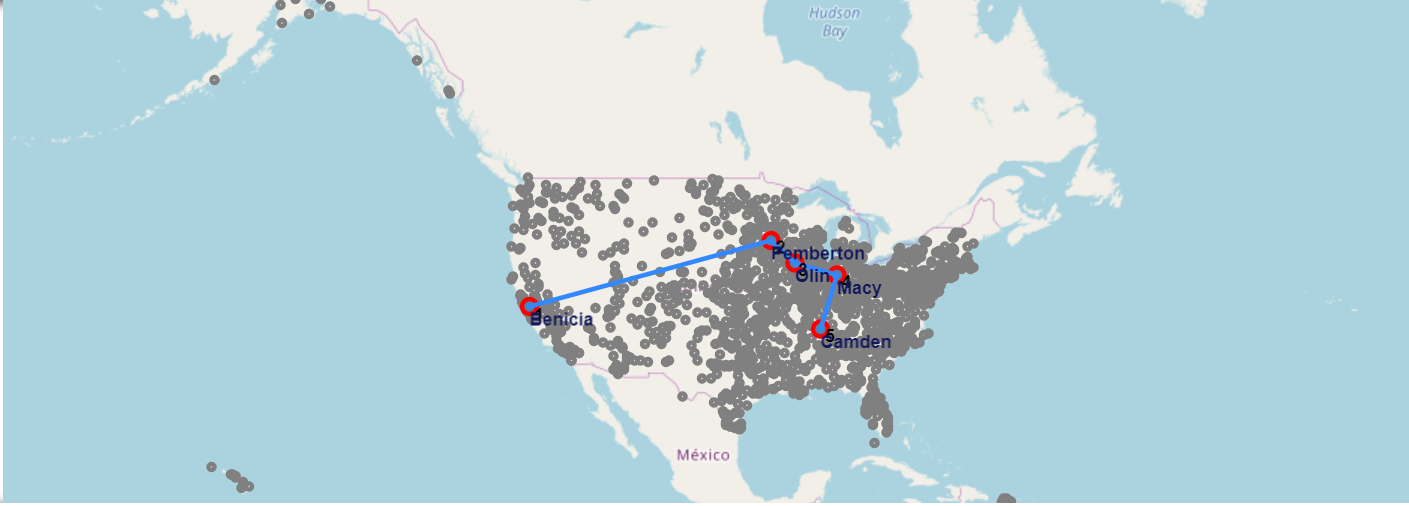
Our output is plot in the geographical map as below –

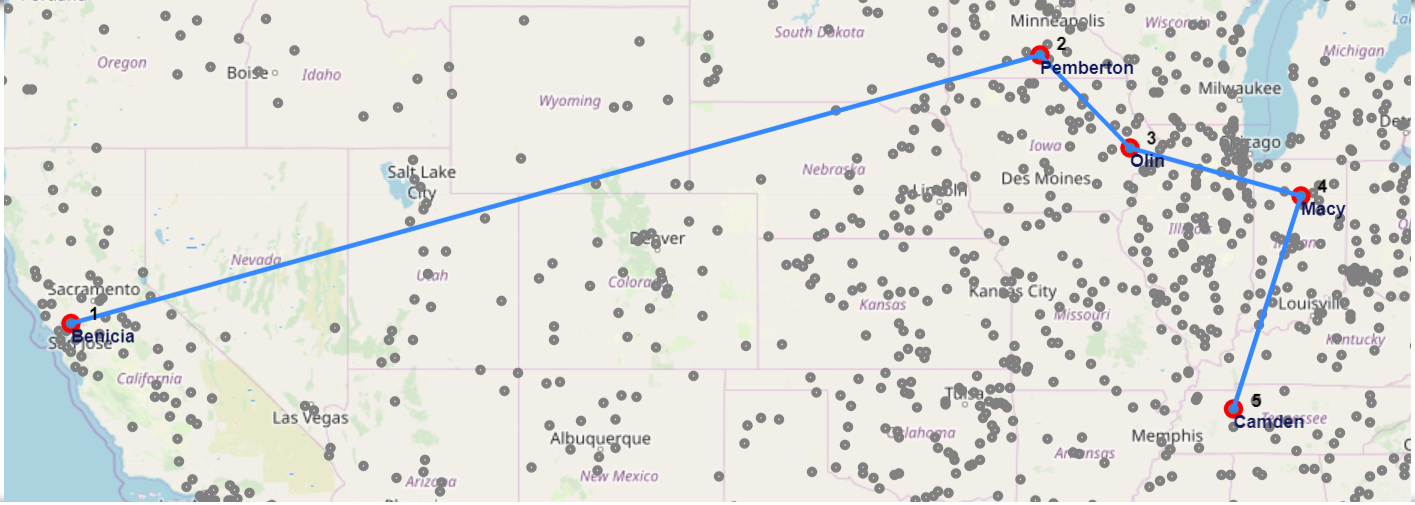
We have two plots in our output one showing the optimal path and the second is for the time complexity through various phases/components of the project.

**The Optimal Path**

The grey points mark all the cities(nodes) under consideration. The red ones are the points involved in the analysis (Staring point 🡪 Intermediate points 🡪 Destination Point). The Blue line illustrates the sequence of the points and hence marking the path to be followed. To make it more precise we have also shown the node number in sequence.

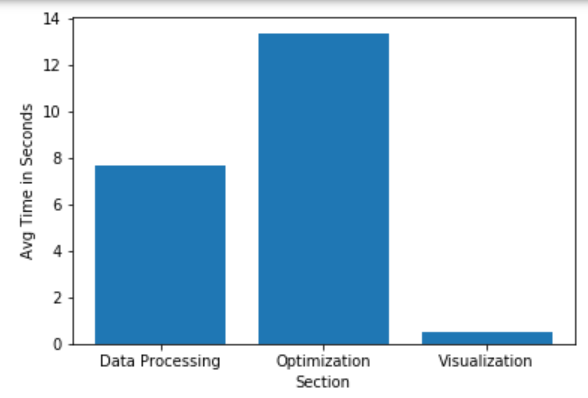
For chart below the origin city is ***Benica*** Intermediate cities (1: ***Pemberton***, 2: ***Olin*** , 3: ***Macy*** ) and destination city ***Camden***.





**Time Complexity through Phases**

The most time complex phase of our project is Optimization Section. Though we have tried to make our algorithm to very efficient still in most of the iterations Optimization section is the most time consuming one.



1. **CONCLUSION –**

Overall Conclusion – We can determine the shortest path given any origin point and destination point.

Future scope:

* External conditions – We can accommodate traffic and weather conditions for weights and constraints.
* Roads – In this project, we are determining the shortest path using air distance. We can accommodate on road routes for road transport.
* Intermediate Stops – We can modify the code to prioritize the route to include user specified additional stops.
* Different mode of transport – We can include use of different mode of transport including Air, Road and Sea to reach from one place to another.

If we scale up the model, for example by considering all the cities in the US and in other countries, this would lead to a huge increase in the number of nodes and routes. This would become computationally challenging.

We can address this problem by renovating the algorithm and adding parallel computing.

1. **REFERENCES -**

1. Dataset is hosted on website 🡪 <https://simplemaps.com/data/us-cities>

1. We referred <http://benalexkeen.com/implementing-djikstras-shortest-path-algorithm-with-python/> for the optimization (Dijkstra’s) algorithm.
2. We have referred the documentation of folium library to plot the charts 🡪 <https://python-visualization.github.io/folium/>