EE232E PROJECT 5

GRAPH ALGORITHMS

Devanshi Patel	504945601	devanshipatel@cs.ucla.edu
Ekta Malkan	504945210	emalkan@cs.ucla.edu
Pratiksha Kap	704944610	pratikshakap@cs.ucla.edu
Sneha Shankar	404946026	snehashankar@cs.ucla.edu

Introduction

The applications of Graph analysis are unlimited, spanning from social networks to DNA sequencing. Some of the well-known algorithms include Shortest Path, Minimum Spanning Tree, Euler's Path, The Chinese Postman Problem, Hamiltonian Circuit, Network Flows, Optimal-Graph Coloring, etc.

In this project , we work on two different datasets, namely Stock Market data and Uber data, and perform analysis using some interesting graph algorithms and theorems. In the first part, we model correlation graph between stocks with time series prices. In the second part, we analyse traffic data on real data provided by Uber.

Stock Market

In Stock Markets, stock prices are affected by a number of economic factors like the country's GDP, emerging markets, economic growth, prices of raw materials and dependent commodities, inflation, population and employment ratios, etc. But investor-sentiments play the most vital role in stock market. If the investors feel a particular stock or a sector that stock belongs to is booming, a higher investments in that stock or all stocks in that sector is expected. Alternately, if the investor feels that a stock/sector is not doing that well, a lower interest in that stock is seen. Thus, investors have similar strategies for stocks affected by same economic factors.

Correlation, in the finance and investment industries, is a statistic that measures the degree to which two securities move in relation to each other. **Correlations** are used in advanced portfolio management. Correlation amongst stocks and also amongst the sectors that these stocks belong to is an important metric in stock market analysis.

In this part, we understand time-series of stock returns using graphs. The dataset is a collection of stock returns over 3 years from Yahoo finance website. We perform our analysis for both the day-scale and the week-scale and compare them.

As a preliminary, we apply log-normalization to the returns. We then construct correlation graphs and apply the Minimum Spanning Tree algorithm. We then perform sector prediction of an unknown stock and repeat all these processes for weekly data.

Question 1: Provide an upper and lower bound on ρ_{ij} . Also, provide a justification for using log-normalized return (ri(t)) instead of regular return (qi(t)).

Answer: The correlation coefficient, ρ_{ij} is bounded between **-1 and 1**. The value of ρ_{ij} gives us an intuition about a pair of stocks that are affected by the same economic factors.

The lower bound value is -1 which indicates that this particular pair of stocks is inversely correlated. Inverse correlation means that when price of one stock goes down, the price of other stock rises and vice-versa.

When we say that the upper bound is 1, it means that the pair of stocks is highly correlated. This would imply that when prices rise, it rises for both the stocks and vice-versa. Although the growth /increase would be different for both these stocks, but the direction of rise/fall is the same. And it is easy to apply the same strategy to buy and sell these stocks and maximize the returns.

We can use two types of returns for finding correlation as follows:

Regular Returns :

$$q_i(t) = \frac{p_i(t) - p_i(t-1)}{p_i(t-1)}$$

Here, $p_i(t)$ is the closing price of stock i at the t^{th} day and $q_i(t)$ is the return of stock i over a period of [t - 1, t].

Log Normalized Returns :

$$r_i(t) = \log(1 + q_i(t))$$

where $r_i(t)$ is the log-normalized return of stock i over a period of [t - 1, t].

In this project, we use log-normalized return instead of the regular return. The reason for that is that the log function helps in reducing the variation in the values of return and at the same time, keeps it bounded. Using the log scale helps in getting an idea about relative changes in stock prices and returns. A regular scale would have provided absolute values which is not really useful for practical analysis.

Log-Normalization helps to reduce the skewness in the data, as it helps us to calculate the percentage change between data points instead of absolute change. For example, consider we have two different stocks A and B.

For stock A, the value at time t was 10, and at time t+1 was 14. This means there was a 40% increase. But the absolute increase was only (14-10) =4

Natural Log (1+0.4)= ln(1.4)= 0.3365

On the other hand, let's say that for stock B, at time t the price was 250, and at time t+1, the price was 350. In this case the absolute returns would be (350-250)=100, however , the percentage change was the same, i.e. 40%.

Natural Log(1+100/250)= ln(1.4) = 0.3365

For certain stocks (as shown above) , the absolute value of return might be different but the log returns is same. In such cases, we want to treat both stocks similarly (either buy or sell) since they have same percentage change.

Use of log scale also ensures that for small returns, it gives a value close to absolute return. Additionally, it also facilitates easier calculation of compounding return over a sequence of trades.

Question 2: Plot the degree distribution of the correlation graph and a histogram showing the un-normalized distribution of edge weights.

Answer: In this question, we construct a correlation graph using the computed coefficients. But before generating the graph, we clean up the data so that the length of time series prices for each stock is same(764). There are few stocks that have less number of entries for prices We ignore such stocks while generating the graph. The nodes of the graph are nothing but different stocks. The edges indicate correlation between a pair of stocks and the weights of the edges are calculated using the below equation:

$$w_{ij} = \sqrt{2(1 - \rho_{ij})}$$

The above equation suggests that for higher values of correlation, the weights will be low and for low correlation, the weights in the graph will be high. To get an idea about the properties of this graph, we plot its degree distribution as shown below:

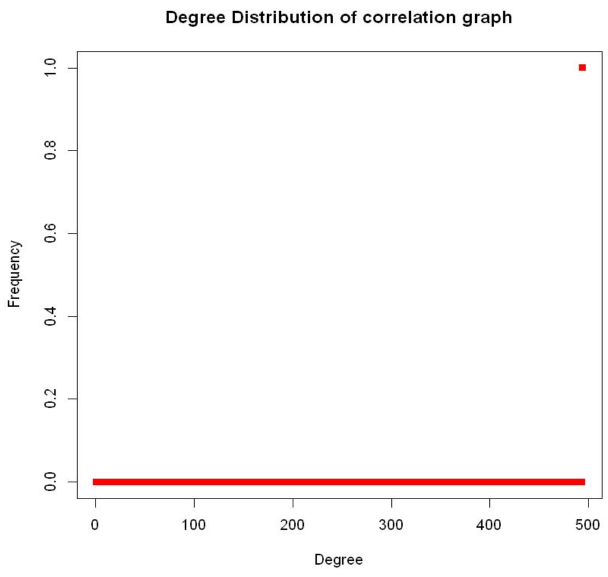


Fig.1

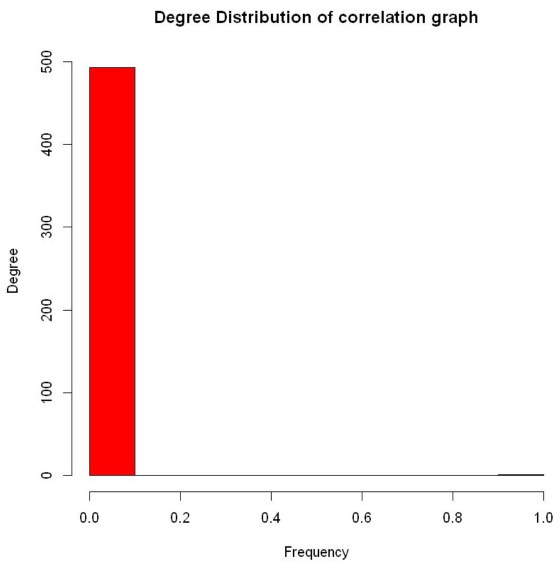
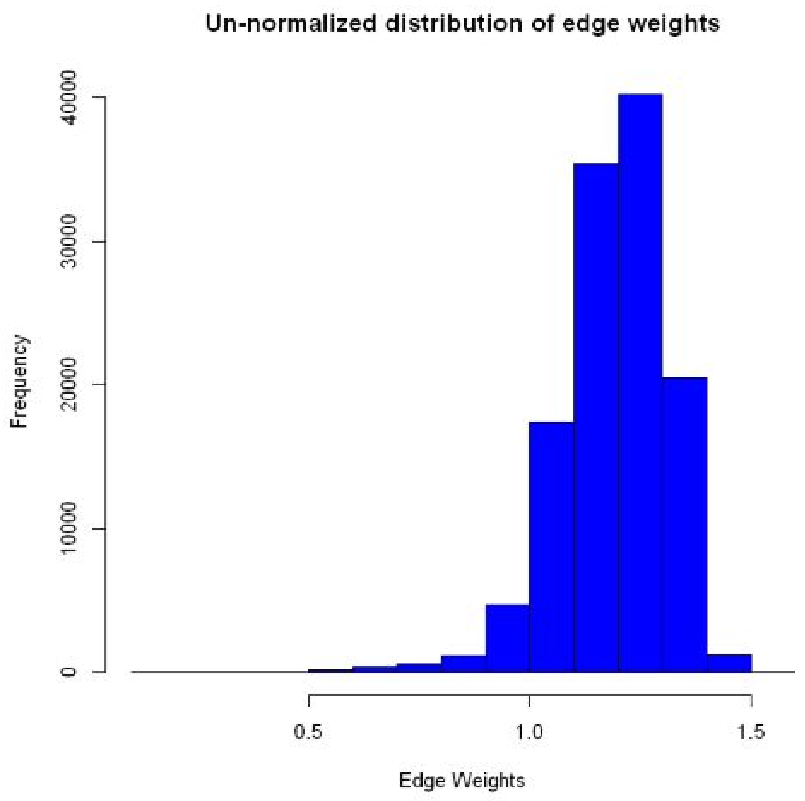


Fig.2

We can clearly tell from the distribution that all nodes have same degree (a square in the fig.1 at co-ordinate 493,1). This distribution is equal to the total number of vertices (494) minus 1. This is quite expected because we calculate correlation between each pair of stocks. So, there will be an edge between each pair of vertices; thus leading to equal degree distribution.



We also plot a histogram that shows un-normalized distribution of the edge weights. We observe that the weights range from 0.5 to 1.5 only. It can be seen that majority of the edges have weights more than 1 and just few of them have lower weights. This indicates that there are only few pairs of stocks with high correlation and rest of them are not really correlated. This is expected because we have drawn edges between each of vertices so it is obvious that only some of them will be correlated.

For perfect correlation, the edge weights will be 0. As in our plot above we see no data at edge-weight 0, it means no pair of stocks in our data have perfect correlation.

Minimum spanning tree (MST)

A **minimum spanning tree** (MST) or **minimum** weight **spanning tree** is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the **minimum** possible total edge weight.

Question 3: Extract the MST of the correlation graph. Each stock can be categorized into a sector, which can be found in Name_sector.csv file. Plot the MST and color-code the nodes based on sectors. Do you see any pattern in the MST? The structures that you find in MST are called Vine clusters. Provide a detailed explanation about the pattern you observe.

Answer: In this part, we extract the minimum spanning tree(MST) from the correlation graph generated earlier. The MST picks out a set of (n-1) edges with large correlation coefficients and minimum edge weights. We use the built-in function mst() available in igraph library of R. We use Prim's algorithm to generate MST for our weighted undirected graph.

Each stock can be categorized into a sector so that all stocks belonging to the same sector can be treated in a similar manner. We collect the sector information for stocks and then plot the MST by color-coding the nodes such that all nodes belonging to the same sector are assigned the same color. The plots shown below represent the MST for our graph.

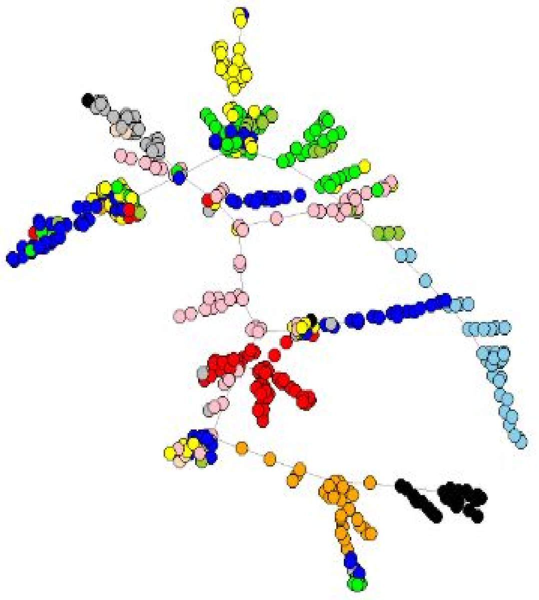


Fig.3

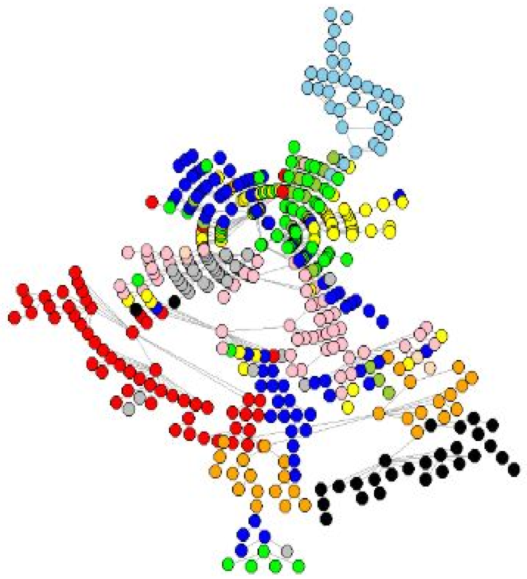


Fig.4

In the Fig.4, we can see each node clearly in the circular layout. It can also be observed that the stocks coded by same color tend to cluster together. These clusters can be used to identify different economic groups/sectors in the stock market. These clusters are also evident from Fig.3 where we observe a distinctive pattern in the MST. These structures are called Vine clusters and they also represent different economic sectors. In other words, the Fig.3 shows a tree-like structure where nodes belonging to the same sector are arranged as branches. From [1], we found that for lower time scale, we get a star-like structure. This is because on smaller time scales, the effect of non-market mode correlations is covered by the correlations caused due to market part. Since our time scale is of one day, we get a Vine-cluster structure.

Additionally, these structures can be really useful for stock investors as they tend to have same investment strategy for stocks in the same sector. This is because in the longer run, the cross correlation between the stocks belonging to same cluster is high.

Question 4: Report the value of α for the above two cases and provide an interpretation for the difference.

Answer: In this part, we try to predict the market sector of an unknown stock in the market. We perform this using two different methods. The first metric used to evaluate the performance of the methods is listed below:

$$\alpha = \frac{1}{|V|} \sum_{v_i \in V} P(v_i \in S_i)$$

Here, S_i denoted the sector of a node i, |V| is the total number of vertices and the probability P has been defined below:

$$P(v_i \in S_i) = \frac{|Q_i|}{|N_i|}$$

Here, Q_i is the set of neighbors of node i that share the same sector and N_i, is the set of all neighbors of node i. Thus, we define probability as the fraction of neighbors of node i that belong to the same economic sector. We compare the performance of this method with another method where the probability is defined differently as shown below:

$$P(v_i \in S_i) = \frac{|S_i|}{|V|}$$

Here, the probability is simply the number of nodes belonging to the same sector as node i as a fraction of total number of vertices. We report the value of α using both methods:

- Value of α using first method: **0.8289**
- Value of α using second method: **0.1142**

From the values of α, we see that there is a huge difference in the two values. The value of 0.8289 indicates that this method is highly efficient in predicting the sector of an unknown stock. This is because this method utilizes the cluster structure of MST and looks at sectors of only neighboring nodes to calculate the probability. We have already seen that stocks belonging to same sector tend to form Vine clusters. Thus, the method 1 leads to a better evaluation.

On the other hand, method 2 gives a very low value of 0.1142. This is quite expected as the method does not take the structure of MST into consideration while calculating the probabilities. It simply calculates the basic probability of a node belonging to a sector. Thus, it is no surprise that this method fails to provide a good evaluation on performance.

Question 5: Extract the MST from the correlation graph based on weekly data. Compare the pattern of this MST with the pattern of the MST found in question 3.

Answer: Until now, we were working with correlation graph based in daily time series data. In this part, we construct a correlation graph based on weekly data. We sample the stock data weekly to calculate correlation. For this, we select a subset of data that is sampled on Monday and then we clean the data to have equal number of samples for each stock. Additionally, we ignore the weeks that have a holiday on Monday.

We calculate the correlation values between each pair of stocks using the sampled data. We then proceed to generate a correlation graph similar to the previous questions. We extract the MST from the graph and plot it by color-coding the stocks belonging to the same sector. As mentioned in Question 3, all stocks belonging to same economic sector have been assigned same colors.

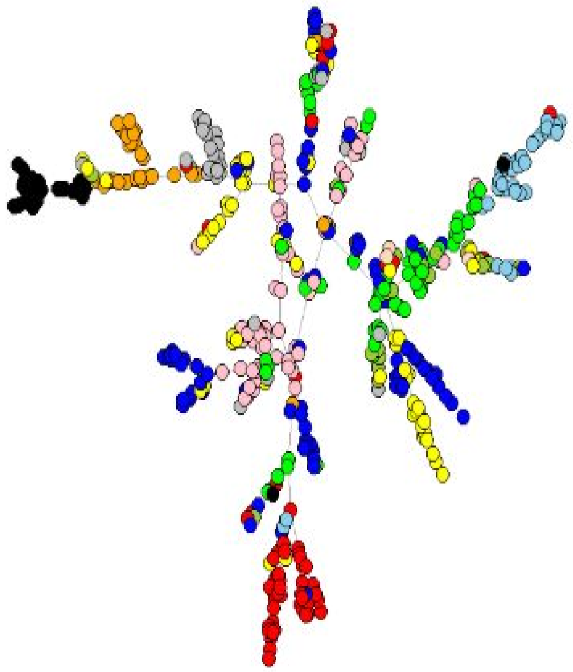


Fig.5

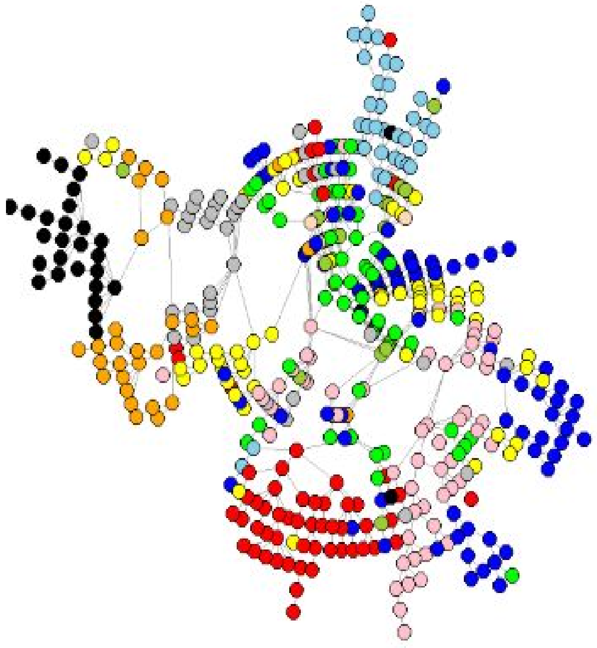


Fig.6

The Fig.6 shows the clusters in a circular layout where we can observe that although the nodes belonging to same cluster are well-separated, it is not as well-separated as in Question 3 with daily data. This is also evident from Fig.6 where we see Vine like clusters.

If we compare the structures of Question 3 with this question, we observe that there is not much difference in the clusters as we are comparing daily data with weekly data. If we would have compared daily data with monthly data, we could have observed more differences in the structures. Additionally, in order to get an idea about the performance of prediction of sector for an unknown stock, we calculate the α metric similar to that in previous question.

- Value of α using first method: **0.7439**
- Value of α using second method: **0.1147**

We can observe that the value of α using second method is not affected and stays almost the same for daily as well as weekly data. On the other hand, the value of α for the first method, reduced from 0.8289 to 0.7439. This indicates that performance for prediction of economic sector for an unknown stock reduces as we increase the time scale of data. This is also evident from the Fig.5 where the Vine clusters are not as well-separated and distinctive as in the case of daily stock data.

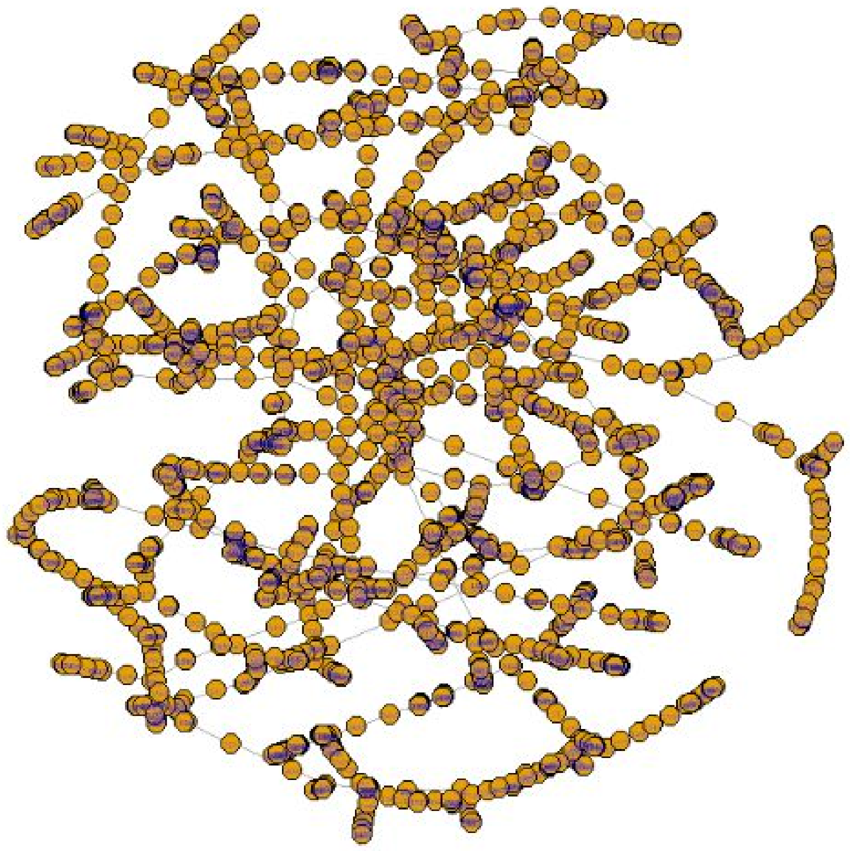
Question 6: Report the number of nodes and edges in G.

Answer: Total number of nodes formed initially is 1887. Total number of nodes after taking gcc and merging duplicate edges is 1880. The number of edges are:311802.

Nodes:1880
Edges: 311802

Question 7: Build a minimum spanning tree (MST) of graph G. Report the street addresses of the two endpoints of a few edges. Are the results intuitive?

MST



Endpoints for a few edges of the MST are:

Endpoint 1	Endpoint 2
400 Northumberland Avenue, Redwood Oaks, Redwood City	1500 Oxford Street, Palm Park, Redwood City
100 Carlsbad Circle, Vacaville	Interstate 505, Vacaville
700 Carlsbad Court, Petaluma	900 Telford Lane, Petaluma
0 Mesa Vista Court, San Ramon	1400 Sheridan Avenue, South Cirby, Roseville
600 9th Street, Pacific Grove	200 Locust Street, Pacific Grove

The results seem pretty intuitive as the endpoints of the edges are pretty close to each other on google maps as well.

Question 8: Determine what percentage of triangles in the graph (sets of 3 points on the map) satisfy the triangle inequality. You do not need to inspect all triangles, you can just estimate by random sampling of 1000 triangles.

Answer: In mathematics, the triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side.

For our scenario, it intuitively means the following:

The least distant path to reach a vertex j from i is always to reach j directly from i, rather than through some other vertex k (or vertices), i.e., dis(i, j) is always less than or equal to dis(i, k) + dist(k, j).

Suppose if A,B,C are three vertices of a triangle, then,
meanTravelTime(A,B)+meanTravelTime(B,C) >meanTravelTime(A,C)

We did random sampling and checked the triangle inequality for 1000 triangles. 923 out of 1000 triangles satisfied the triangle inequality.

92.3% of triangles satisfied the inequality.

Travelling Salesman Problem

The travelling salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route (Hamiltonian Cycle) that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization. [2]

TSP can be modelled as an undirected weighted graph. In this case the cities are the graph vertices and the roads/paths are the edges. This is a cost-minimization problem, as each weighted edge represents the cost to travel that path. The problem can be stated as : Given a vertex, you traverse through all the cities(vertices) in the graph exactly once, finding the minimum-cost possible path and returning to the beginning vertex. If an edge does not exist connecting two cities, a random edge connecting their vertices is added to the graph, to complete the graph without affecting optimal tour.

Question 9: Find the empirical performance of the approximate algorithm.
Answer: As TSP is an NP-Hard problem, there are two ways in which the problems of TSP can be solved.

1. Exact algorithms
2. Approximate Algorithms

Exact algorithms take very direct approach, such as finding all possible permutations and calculation of the least cost path, (a O(n!) solution) or using Dynamic Programming and other approaches which reduce the complexity a bit, but still impractical for large-scale graphs.

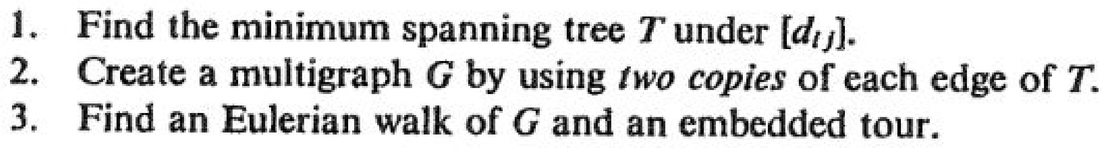
Approximation algorithms are efficient algorithms that find approximate solutions to NP Hard problems. Approximation algorithms are a consequence of P!=NP belief in CS. The approximate algorithms work only if the problem instance satisfies Triangle-Inequality. An Approximate algorithm always gives provable guarantees about the distance of the returned solution to the optimal one.

Without triangle inequality, finding good approximate tours for TSP in polynomial time is impossible unless P=NP [3].

For this project,we have used MST based TSP algorithm which uses Euler's Circuit. In graph theory, a multigraph (in contrast to a simple graph) is a graph which is permitted to have multiple edges (also called parallel edges), that is, edges that have the same end nodes. Thus two vertices may be connected by more than one edge.

A multigraph is called Eulerian, if it has a closed walk in which each node appears at least once, and each edge appears exactly once. Such a walk is called Eulerian walk. A Eulerian cycle is a Eulerian walk that is closed.

A one-approximate solution for TSP uses the following algorithm:



1. **Find the minimum spanning tree T under $[d_{ij}]$.**
2. **Create a multigraph G by using *two copies* of each edge of T.**
3. **Find an Eulerian walk of G and an embedded tour.**

Where d_{ij} is the weight of edge between vertices i and j.

We have implemented a 2- approximate solution for TSP using the following algorithm:

1. Find the minimum spanning Tree of the graph (MST).
2. Derive a preorder traversal of the minimum Spanning Tree.
3. For each pair of vertices from the preorder traversal in order, check if an edge exists between the vertices in the MST.
4. Initialize a Sum_of_Cost variable to zero.
5. If an edge exists in the MST, add its mean travel time (edge weight) to the Sum_of_Cost.
6. Else, check if the edge between this pair of vertices exist in the original graph. If yes, take its mean travel time and add it to the sum.
7. Otherwise, find the shortest path between these two vertices in the original graph. Add the cost of this shortest path to the Sum_of_Cost.
8. The final sum obtained by this algorithm is the 2-approximate TSP cost.

We applied the algorithm exactly as above , and found our 2 approximate TSP cost to be **464758.3.** The MST cost is **279408.18.**

For this question, we calculate the empirical performance of the approximate algorithm as follows:

$$\rho = \frac{\text{Approximate TSP Cost}}{\text{Optimal TSP Cost}}$$

$$\boldsymbol{\rho} \mathbf{= 1.663}$$

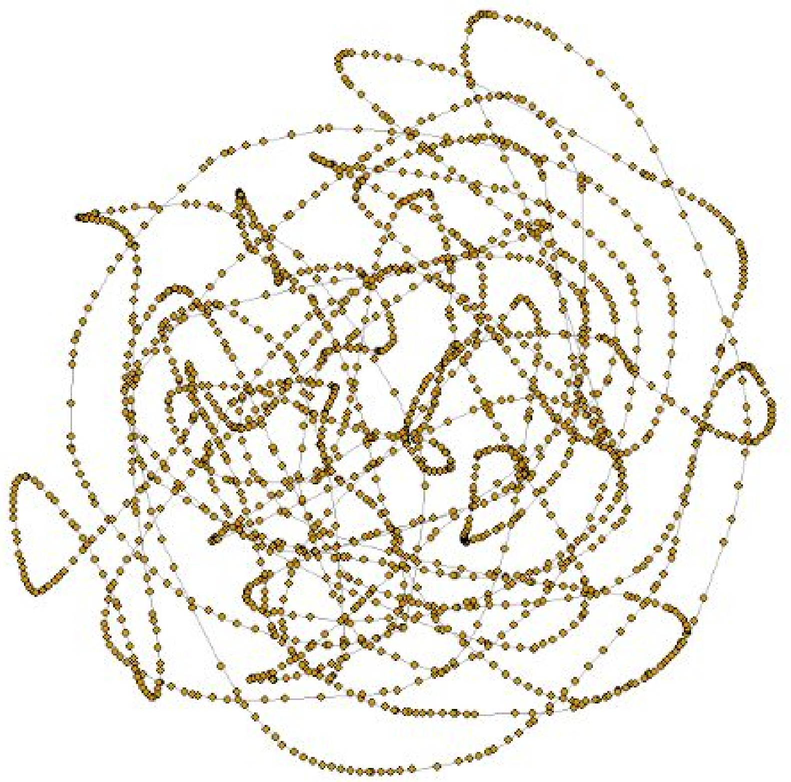
For calculation of ρ, we used the MST cost in the denominator instead of Optimal TSP cost. Finding the optimal TSP is an NP-Hard problem. And using MST cost in the denominator is a good approximation here as the cost of best possible Travelling Salesman tour is never less than the cost of MST.

The relation between the various costs can be summarized as follows:
mst cost < optimal tsp cost < approximate tsp cost < 2 * mst cost

279408.18 < Optimal TSP Cost < 464758.3 < 558816.36

Thus, the upper bound on the empirical performance of the 2-approximate algorithm is **2**. The lower bound would be slightly higher than the MST cost and hence is **1** . Hence the bounds with respect to MST cost lie between **1 and 2**.

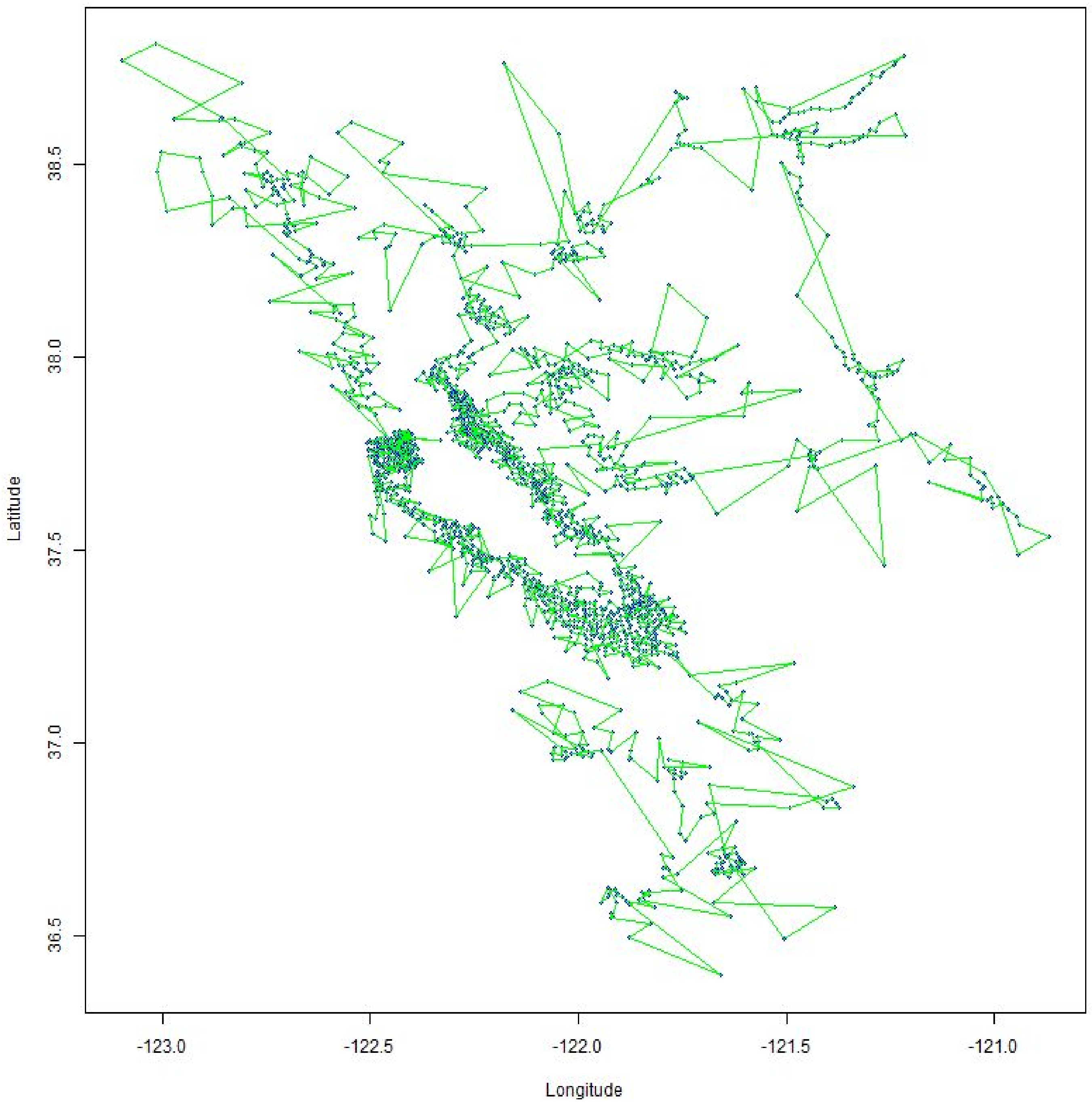
We can visualize the preorder traversal obtained after 2-approximate TSP as follows:



Question 10: Plot the trajectory that Santa has to travel!

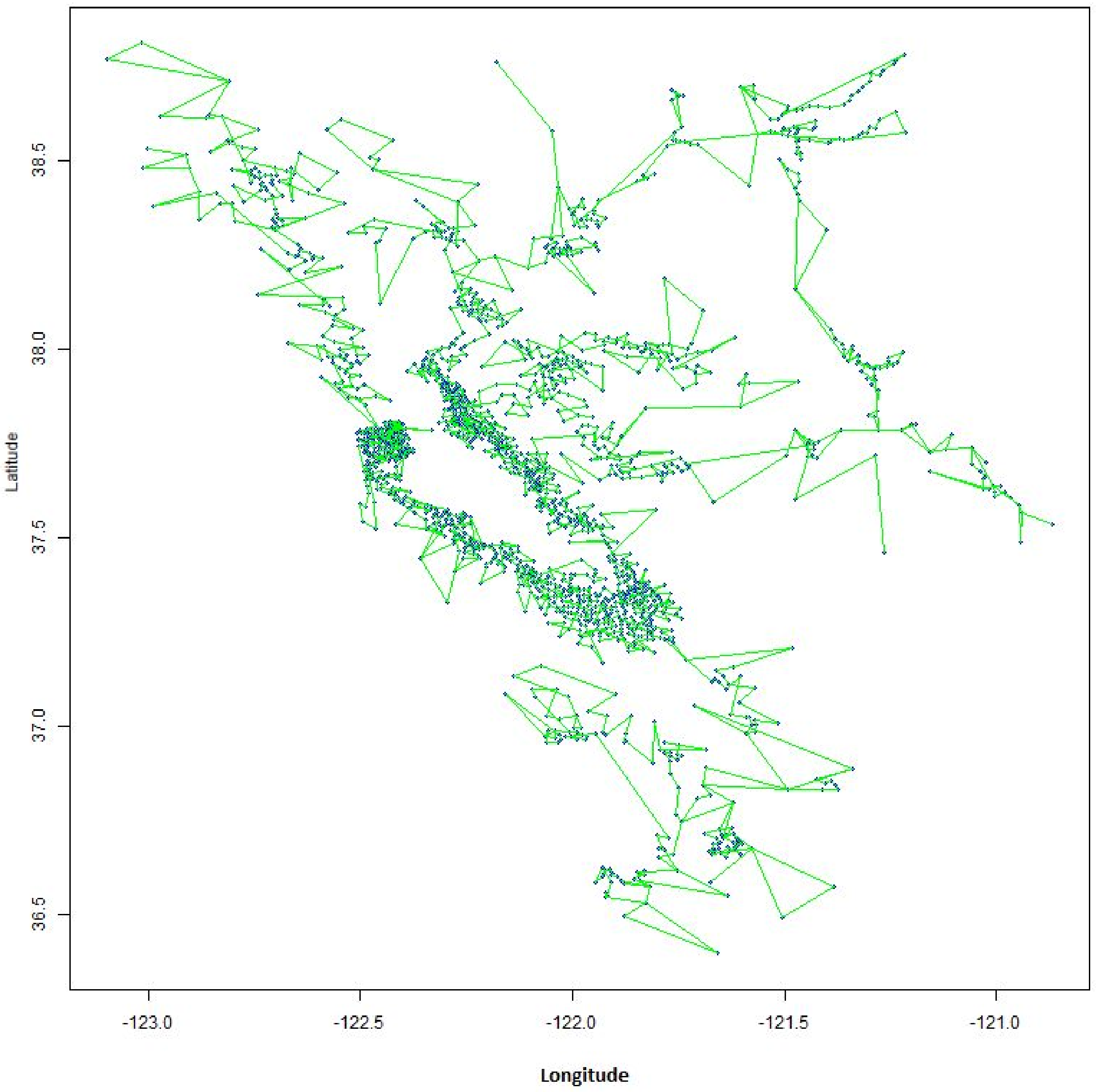
Answer: It’s Christmas time, and Santa has to travel through the homes of many children and deliver them presents. This is a classic Travelling salesperson Problem that we solved above. So we decided to help Santa with our Graphs and Networks Analysis.

The trajectory that Santa has to travel can be visualized as shown below:



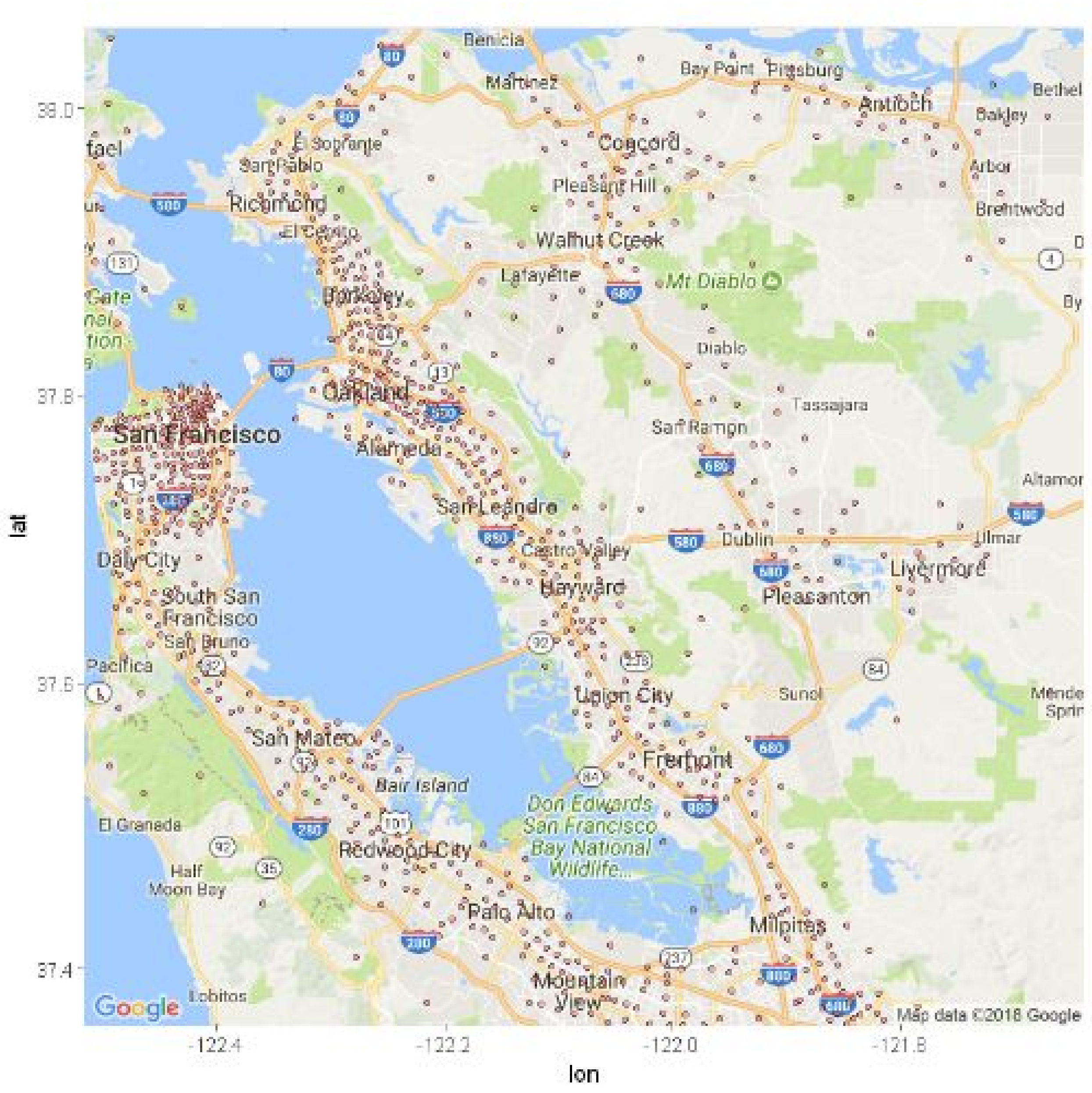
The figure obtained above is the path Santa has to travel. However, there are many edges on this path that do not actually exist. But as Santa travels on reindeers, this should not be a problem!!

But realistically, we decided to help Santa a little further. Hence, we ran our algorithm again, this time taking into consideration the actual shortest paths, and not imaginary edges. If an edge does not exists between two vertices, we calculate the shortest path route and add that to the map. Please note this will not be an actual Euler’s walk (in a Eulerian circuit, each edge gets traversed only once, whereas each vertex gets traversed at least once) , as many edges would get traversed twice, due to absence of a few direct routes. This is for comparison purposes only. The new graph obtained now, after all these changes is as follows :



As we can see from the graph above, a few paths appear hanging. This is because, as those vertices are leaf nodes in the minimum spanning tree, they are connected to only one parent vertex. Hence, when Santa reaches those vertices, without any imaginary paths into consideration, he needs to traverse back to some other vertex, which allows him to simulate the imaginary vertex forming shortest path to the next leaf node.

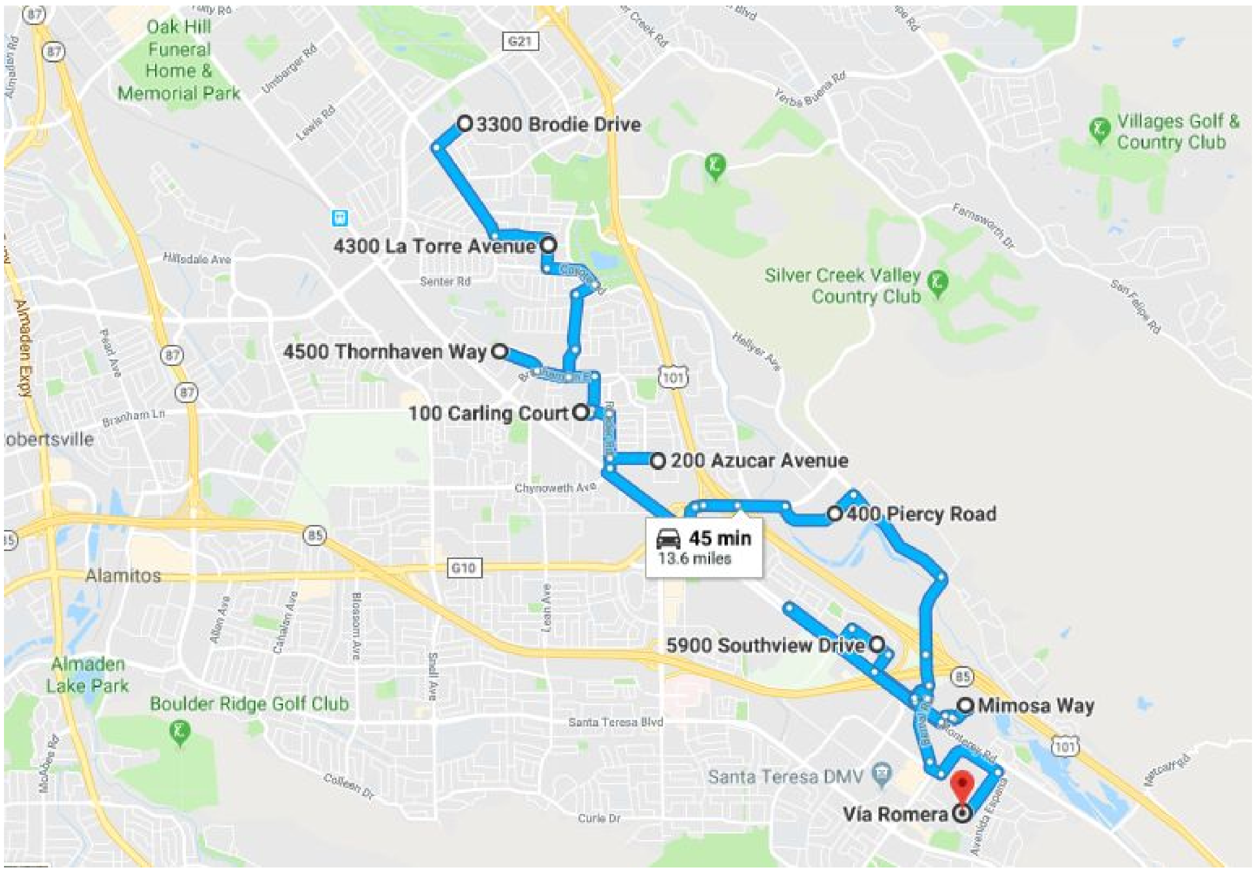
For visualization, we now plot the trajectory on actual map as follows:



Q: Are these results intuitive?

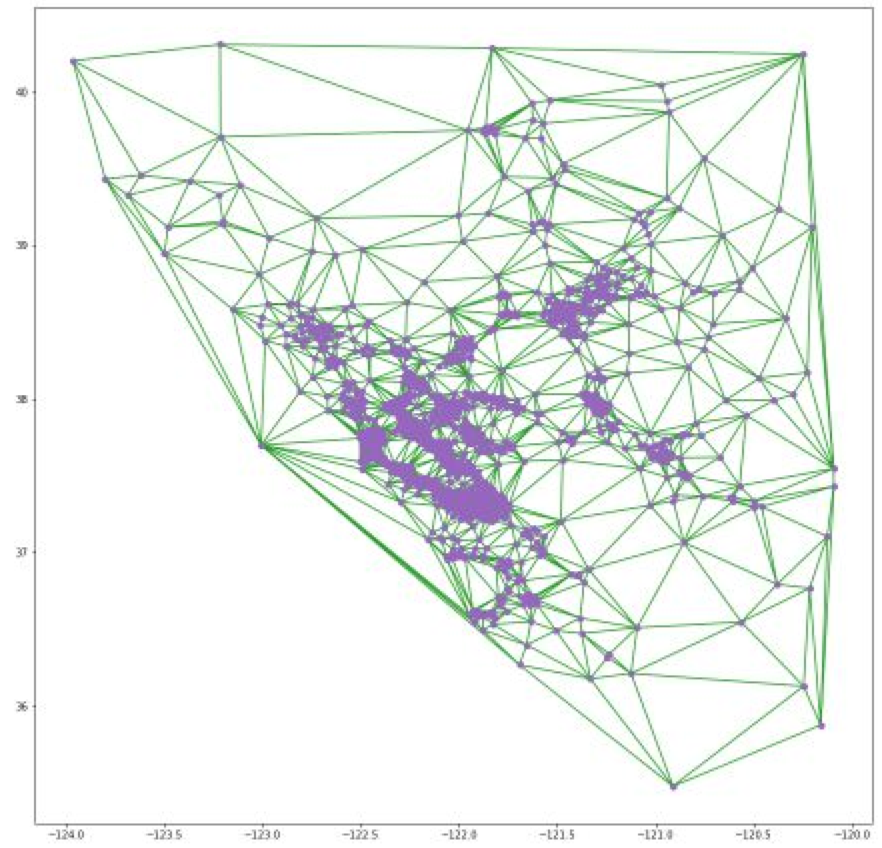
We checked for first 10 street address locations obtained from the 2-approximate TSP algorithm as follows :

Latitude (Y)	Longitude (X)	Street Address
37.29050528	-121.8217357	3300 Brodie Drive, South San Jose, San Jose
37.28097776	-121.8145113	4300 La Torre Avenue, South San Jose, San Jose
37.27304983	-121.8169314	4500 Thornhaven Way, Edenvale, San Jose
37.26705557	-121.8142874	100 Carling Court, South San Jose, San Jose
37.2628994	-121.8049488	200 Azucar Avenue, Edenvale, San Jose
37.25524618	-121.7679505	400 Piercy Road, Edenvale, San Jose
37.25094654	-121.7895472	5900 Southview Drive, Edenvale, San Jose
37.23473767	-121.7626609	Mimosa Way, South San Jose, San Jose
37.22992144	-121.7718317	7100 VÃ-a Romera, South San Jose, San Jose
37.22305783	-121.7646813	100 Cheltenham Way, South San Jose, San Jose



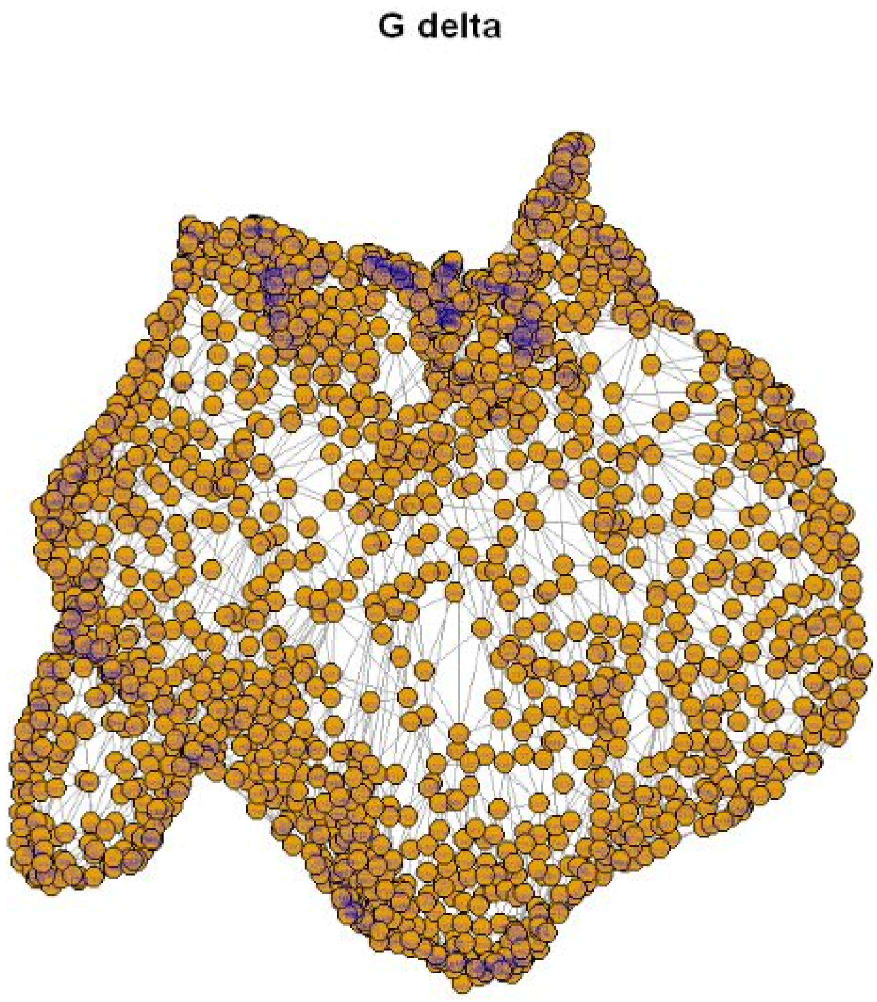
As we can see from the plot above, the mean travel time to visit 10 places is 45 minutes. This means that the average time to reach the next place from current place is about 4.5 minutes. Also , the places follow a logical order, and none of the new locations lead to traversing and passing over the old location again. From the above figure, we can see that each stop follows the path from the previous stop. Thus, the results obtained are quite intuitive.

Question 11: Plot the road mesh that you obtain and explain the result.Create a subgraph G induced by the edges produced by triangulation.



Answer: The above figure is the road mesh obtained after triangulation. After noticing the graph, we can observe that the points represents the road structures of San Francisco. Although, there are many edges in the road mesh which actually do not exist. This is because Delaunay's algorithm adds edges which does not exist such that angle between edges is not very small. It basically tries to fit every triangle inside a circumcircle to achieve this.

The graph induced by the edges of triangulation is:



The number of edges in this graph (G Delta) is 5627. The edge count in the previous graph G was 311802. This proves that we have correctly induced the edges as produced by triangulation.

Question 12: Using simple math, calculate the traffic flow for each road in terms of cars/hour.

Answer: In this part of the question, we calculate the traffic between Stanford University and University of California, Santa Cruz (UCSC). Each road is considered as an edge. We use the following formula to calculate the cars/hour which will basically serve as our traffic flow.

Traffic flow of a single lane = Number of cars in that lane per hour
Flow Rate of a single lane =
(60* 60) / Mean travel time) * ((Distance)/((2*speed of car) + 0.003))
= (60* 60) / Mean travel time) * ((Speed of car * Mean Travel Time) / ((2*speed of car) + 0.003))

By simplifying the above formula, we get
Flow Rate of a single lane = 3600/((0.003/speed of car)+2)
To get the flow rate of the roads from Stanford to UCSC (only one direction and not the reverse), we multiply the flow rate of a single lane by 2 since there are 2 lanes in a road in a particular direction.

Then, for every edge of the graph formed (the new induced subgraph), we then calculate the FlowRate of the edge. This rate is in terms of cars per hour through both the lanes of the road from Stanford to UCSC.

Analysis: This flow obtained is going to serve as the max capacity of each road. This means that at a time, a particular road cannot in any means have number of cars (in one hour) exceeding the capacity it can hold. This max capacity can be thus considered as the max flow or in this case the max passage of cars through the road. Thus, flow can never exceed the capacity in any edge or road.

Question 13: Calculate the maximum number of cars that can commute per hour from Stanford to UCSC. Also calculate the number of edge-disjoint paths between the two spots. Does the number of edge disjoint paths match what you see on your road map?

Answer: In this part of the project, we apply the max-flow, min-cut algorithm so as to determine the how many cars travel from Stanford to UCSC per hour. Here, Stanford will be considered as the source, UCSC as the sink and the flow is thus between the source and the sink.

Maximum Flow from Stanford to UCSC is: 12245.54 cars/hour i.e. approx 12246 cars/hour
The number of edge-disjoint paths obtained is: 5

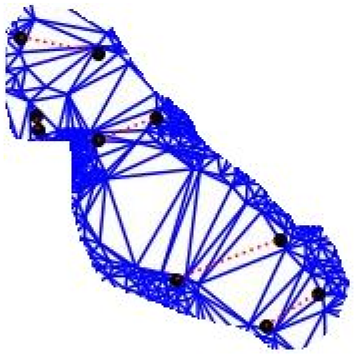
Analysis:
Max flow across the network or the graph will be limited to the summation of the flows across the edges which when removed will disconnect the source from the sink. The collection of such edges is called as the min-cut. That is max-flow = min-cut necessary to separate the source from the sink. If we draw a cut across the graph, the summation of all these edges' (in the direction of source to sink) flows should give the maximum flow in the graph from source to sink. This is the max flow min cut theorem. The max-flow of the graph cannot be more than the summation of thee flows across this cut, which essentially means that the number of edges which form a part of the cut will be the number of edge disjoint paths in the graph. This matches our intuition too, because the flow across any path cannot be more than the minimum capacity of an edge

along that path. This should obviously match with what we see in the road map because it is the road map that we have converted into a graph on which we are running the max flow min cut algorithm. Therefore, it means that the maximum flow from Stanford to UCSC will be achieved along these 5 edge-disjoint paths we have obtained.

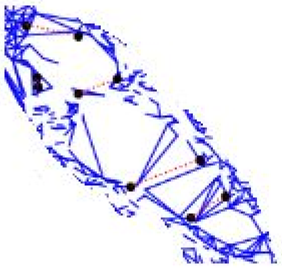
Question 14: Plot ~G Delta on real map coordinates. Are real bridges preserved?

Answer: In this part of the project, we aim to remove all the fake bridges crossing the bay. Why do these fake bridges arise at all? Since we are doing triangulation with Delaunay, in order to create a mesh of triangles, it will add some edges. These triangles are such that each one of it fits inside a circumcircle. In order to create a triangle which fits in a circumcircle, the delaunay algorithm of python might create a larger edge. This edge might be one of the edges/bridges which cross the bay. Such edges are considered as fake bridges in this case. These fake bridges will have larger weights or larger mean travel times. The reason behind this is that these bridges do not actually exist. Thus, it's weight would be the sum of the weights of the other two edges which connect the same points as the fake bridge. Which means the mean standard time along this fake bridge would be more than what is expected.
How can we delete such fake bridges? Here we consider the weights of the edges i.e. the mean standard time of a road as well as the distance to which it connects. This is because, only time might not always give the correct results. It is actually possible in real life that the mean standard time between two points is very large because of the large distance between them. We need to only consider points which are close (by distance) but have a large mean travel time between them. Such edges are fake edges i.e. in our case the fake bridges which cross the bay. Thus, we consider speed which is given by distance/time (so that both these metrics are considered) and remove the fake bridges (or edges) across which the speed is less than 25 miles/hour.

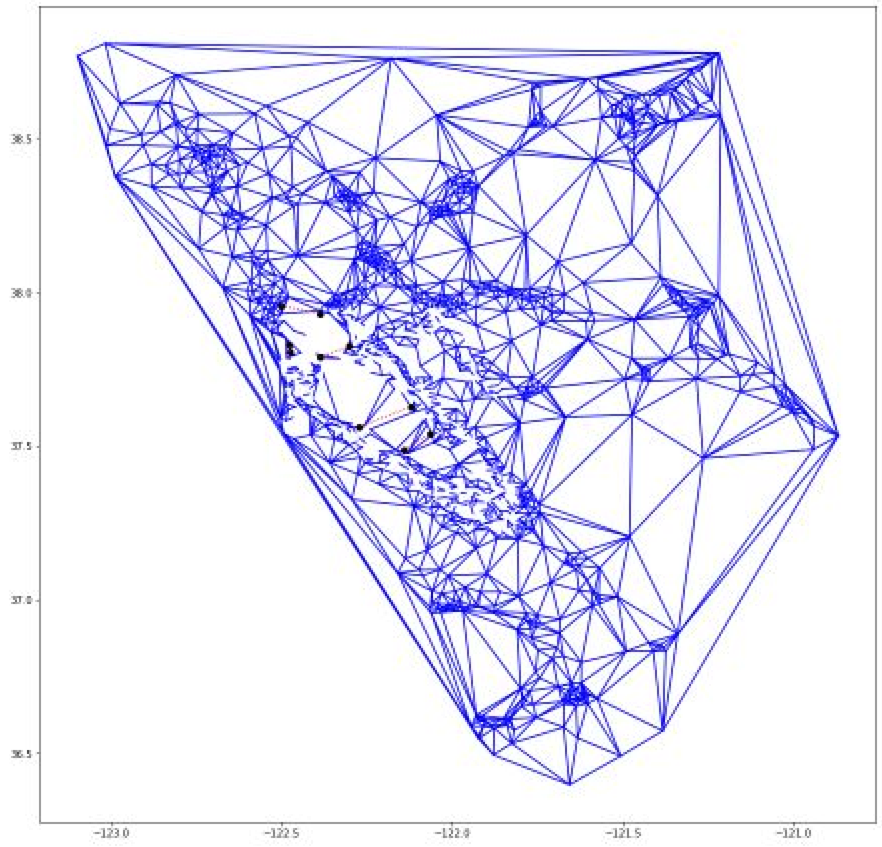
The fake bridges can be seen as follows:



However, after removing the fake bridges, the portion of bay looks like:



The complete graph G’ Delta looks like:



Here we see that the actual bridges are preserved. Which means our algorithm is preserving the edge with shortest mean travel time.

Question 15: Now, repeat question 13 for ~G Delta and report the results. Do you see any significant changes?

Answer: Now that we have removed the fake bridges, we again check the max flow from Stanford and UCSC.
The max flow is : 12243.41 i.e approx 12243 cars/hour
The number of edge disjoint paths is: 5

Analysis:

Here, we see that there is no significant difference in the max flow and the edge disjoint paths between Stanford and UCSC. This is because the removal of fake bridges does not affect the flow of traffic from Stanford to UCSC. The max-flow min-cut algorithm always takes the minimum cut. The summation of flow across this min cut gives the max flow in the network. As mentioned before, the bridges do not have the least weights. Thus, they will not contribute to the min cut and hence will not contribute to the max flow too. As a result, the max flow from Stanford to UCSC remains nearly the same and there is no significant difference observed.

REFERENCES:

1. Qian, Roychowdhury. “The correlations matrix and minimum spanning tree of multiple assets”
2. https://en.wikipedia.org/wiki/Travelling_salesman_problem
3. Generalized Travelling Salesperson Problem, Chapter 35 -Approximation Algorithms,Book-Introduction to Algorithm, 3rd Edition.