Project 4: Graph Algorithms

Large Scale Social and Complex Networks: Design and Algorithms

Ananya Deepak Deoghare, Kimaya Milind Kulkarni, Shruti Mohanty

005627628, 805528337, 705494615

ECE 232E Spring 2022

Stock Market

In this part of the project, we study data from stock market. The goal of this part is to study correlation structures among fluctuation patterns of stock prices using tools from graph theory. The intuition is that investors will have similar strategies of investment for stocks that are effected by the same economic factors. For example, the stocks belonging to the transportation sector may have different absolute prices, but if for example fuel prices change or are expected to change significantly in the near future, then you would expect the investors to buy or sell all stocks similarly and maximize their returns. Towards that goal, we construct different graphs based on similarities among the time series of returns on different stocks at different time scales (day vs a week). Then, we study properties of such graphs. The data is obtained from Yahoo Finance website for 3 years. We're provided with a number of csv tables, each containing several fields: Date, Open, High, Low, Close, Volume, and Adj Close price. The files are named according to Ticker Symbol of each stock, the market sector for each company is present in *Name_sector.csv*. We use the below notations:

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

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

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

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

Then with the above notation, we define the correlation between the log-normalized stock-return time series data of stocks i and j as

$$\rho_{ij} = \frac{\langle r_i(t) r_j(t) \rangle - \langle r_i(t) \rangle \langle r_j(t) \rangle}{\sqrt{(\langle r_i(t)^2 \rangle - \langle r_i(t) \rangle^2)(\langle r_j(t)^2 \rangle - \langle r_j(t) \rangle^2)}}$$

where $\langle \cdot \rangle$ is a temporal average on the investigated time regime (for our data set it is over 3 years).

Question 1:

Pearson correlation coefficient is represented by the formula given above. The top bound is 1 and the lower bound is -1, as demonstrated by the Cauchy-Schwartz inequality. We employ log-normalized returns to reduce skewness in the data. Taking the log of the returns minimizes the variance in the return, giving us a clearer understanding of the relative changes in stock prices and returns.
When p_{ij} is +1, it indicates that the two stocks i and j are strongly positively correlated, with their prices altering in the same direction (though the scale of growth may be different). When p_{ij} is -1, it means that the two stocks i and j are strongly negatively or inversely correlated, with the stock prices moving in opposite directions. We favor the log-normalized return over the standard return for various reasons. Some of them are given below:

- Reduces skewness present in data
- Absolute figures are less capable of representing relative changes in stock values. Relative adjustments are more practical since the percentage scale of such changes is the

same for equities that are often regarded similarly.

- Reduces the effects of outliers and variation in the data, limiting the values to defined boundaries.
- The scale of smaller stocks is enlarged, while the scale of larger stocks is shrunk, resulting in a homogeneous scale for seeing the relative movements of all stocks regardless of their prices.

Question 2:

In this question, we are requested to plot a histogram of the correlation graph's unnormalized distribution of edge weights. The vertices of the correlation graph are the stocks, and the edge weights are as follows:

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

This shows that, when $p_{ij} = +1$, $w_{ij} = 0$; and when $p_{ij} = -1$, $w_{ij} = 2$. This helps us to find the lower and upper bound of the data. We take the below steps to contruct the graph:

- Omit the stocks which have NaN values (NaN values represent the missing data).
- We calculate the $r_i(t)$ which contains the log-normalized return for each stock i.
- Create a file to store the w_{ij} for each stock i(source node) and j(sink node).
- Using *igraph* library of R, we generate the histogram.

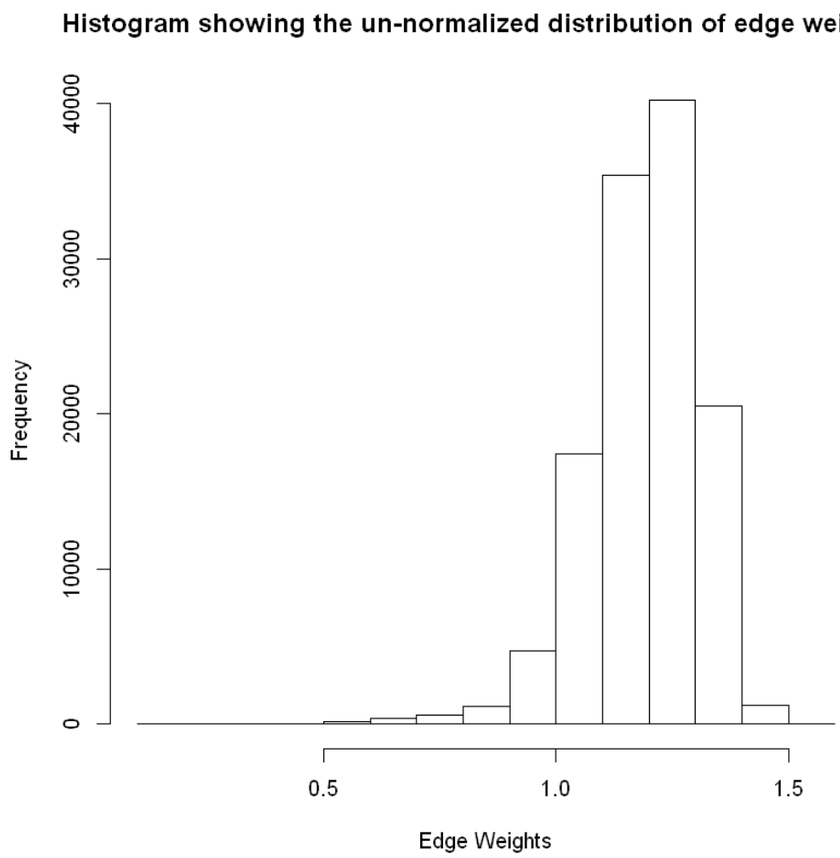


Figure 1: Histogram showing the unnormalized distribution of edge weights for daily data

The above figure shows that the majority of the edge weights are between 1.0 and 1.4. All of the weights are between 0 and 2, as predicted; however, the lowest bound for edge weights is 0.5, not 0. This indicates that there are no stocks with 100% positive correlation since some of the edge weights would have to be 0. The majority of equities are poorly connected with one another, and the majority of stocks have a negative correlation with one another.

Question 3:

In this question, we are required to display the minimum spanning tree (MST) for the correlation graph, color coded to represent different stock sectors (categories). The MST joins all nodes in a connected and edge-weighted undirected graph without any cycles and with the fewest cumulative edge weights achievable. Prim's approach is used to generate the MST from the correlation graph. The figure below depicts the MST for the correlation graph, with stocks from the same industry color colored the same.

Yes, the MST does follow a pattern. In the MST, nodes in the same sector are often linked, whereas nodes in separate sectors are typically detached. The figure above shows this because nodes of the same hue are grouped together, and these clusters are rather homogeneous. This is because nodes in the same sector are strongly connected, but nodes in separate sectors are minimally correlated.

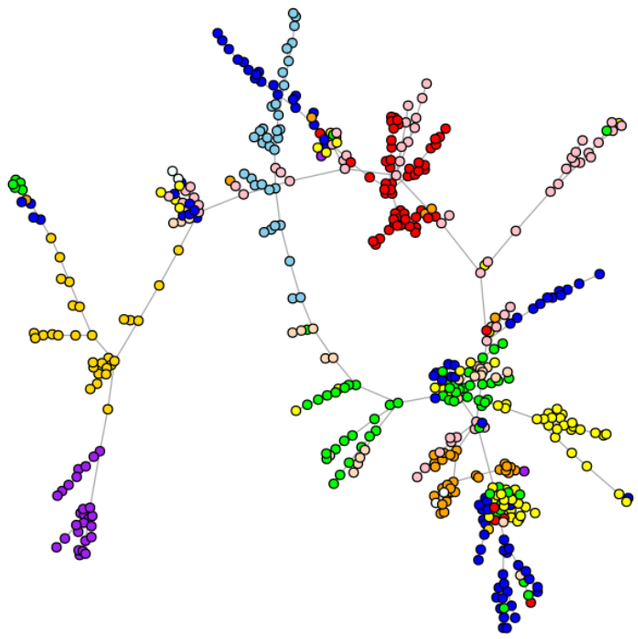


Figure 2: The Minimum Spanning Tree derived from the correlation for daily data

Most stocks in the MST that have a same hue (belong to the same sector) tend to flock together. Nodes of different colors (belonging to different sectors) are not linked to each other. Because MST's purpose is to link all nodes with the lowest cumulative edge weights, it tends to group strongly correlated stocks together. These formations are known as vine clusters because they resemble grapes dangling from a primary branch. The clusters represent various industries. Stocks in the same cluster tend to move in the same direction, necessitating comparable investing methods. Vine clusters depict the stock market over a longer time horizon, such as daily data.

Question 4:

In this question, we are asked to extract the community structures (using walktrap community detection) and homogeneity, completeness scores for the same.
Walktrap, developed by Pascal Pons, is an algorithm in graph theory, used to identify communities in large networks via random walks. These random walks are then used to compute distances between nodes. Nodes are then assigned into groups with small intra and larger inter-community distances via bottom-up hierarchical clustering. It should be noted, of course, that this algorithm considers only one community per node, which in some cases can be an incorrect hypothesis. The walktrap algorithm helps us to cluster and find the community structures in the data. We notice the various clusters that are present in the data and are identified by the algorithm.

A clustering result is homogeneous if all of its clusters include only data points from a single class or circle. The amount of variance in each expected community is measured by homogeneity. A greater homogeneity score is obtained if every anticipated community has people who all share the same circle. The lowest homogeneity score is obtained when members of a forecast community are evenly distributed throughout all feasible circles.
A clustering result is complete if all of the data points in a particular circle are members of the same community. The completeness measure examines each circle and determines the degree of variance in each circle member's community assignment. When every member of a circle is anticipated to belong to the same community, the circle has a high completeness score. If a circle's distribution of community assignments is consistent, the circle has a low community score.

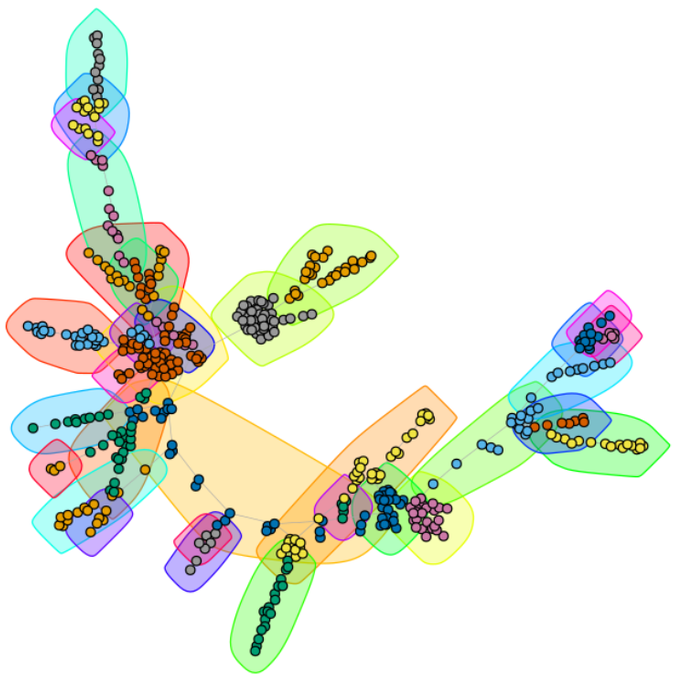


Figure 3: The Community Structure derived from the correlation for daily data

The homogeneity for Daily data: **0.682644648161366**
The completeness for Daily data: **0.479284479244588**

Homogeneity metric of 68% indicates that 32% of the samples on an average are not from the assigned corresponding ground truth label. Completeness of 48% shows that 52% of the samples for a given class label are part of the same cluster.

Question 5:

Let us first compute the value of for the first scenario. We should loop over all of the nodes in the created MST, compute P(vi∈ Si), total it up, and average over the nodes. The α is 0.8289. The second example of α really depends on the distribution of the original data rather than the algorithm outcome. This yields a result of $\alpha = 0.1141$. If a randomly picked stock loses its sector and we try to retrieve its label by randomly selecting a neighbor and assigning the sector to the neighbor, then the equation for may be viewed as the correctness of our model. It is worth noting that the accuracy of the clustering result based on MST may be viewed as the accuracy of the original fully connected graph in the first scenario. The difference between these two α demonstrates the great performance of our clustering result based on MST.

- Technique 1: $\alpha = 0.8289$
- Technique 2: $\alpha = 0.1141$

We see that technique 1 has a larger value than technique 2. This is to be expected since approach 1 makes advantage of the MST vine cluster structures of the correlation graph, taking into consideration which nodes are highly connected and flock together rather than all nodes. In other words, approach 1 makes judgments based on local connectedness among surrounding nodes rather than the global correlation network. Technique 2, on the other hand, evaluates all nodes that belong to a sector and extract local spatial linkages or cluster formations by considering the complete network.

Question 6:

Inspecting the csv files provided, we can see that the stock price data has been automatically removed from weekends and national holidays. That is, we just need to determine if the relevant date is Monday or not. We can successfully retrieve all of Monday's data from three years of temporal arrays using R's weekday and same intrinsic functions. Using the same procedure described in Questions 2,3,4 and 5, we can obtain the clustering result based on MST for the Monday data, as shown in the Figure below.

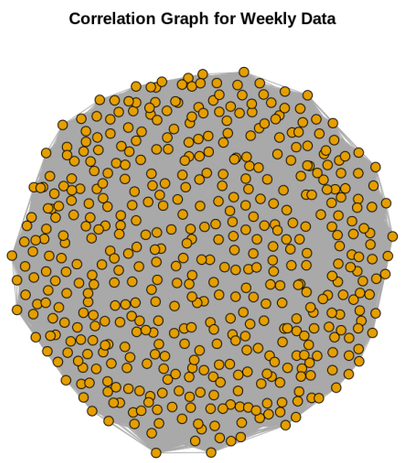


Figure 4: Correlation for weekly data

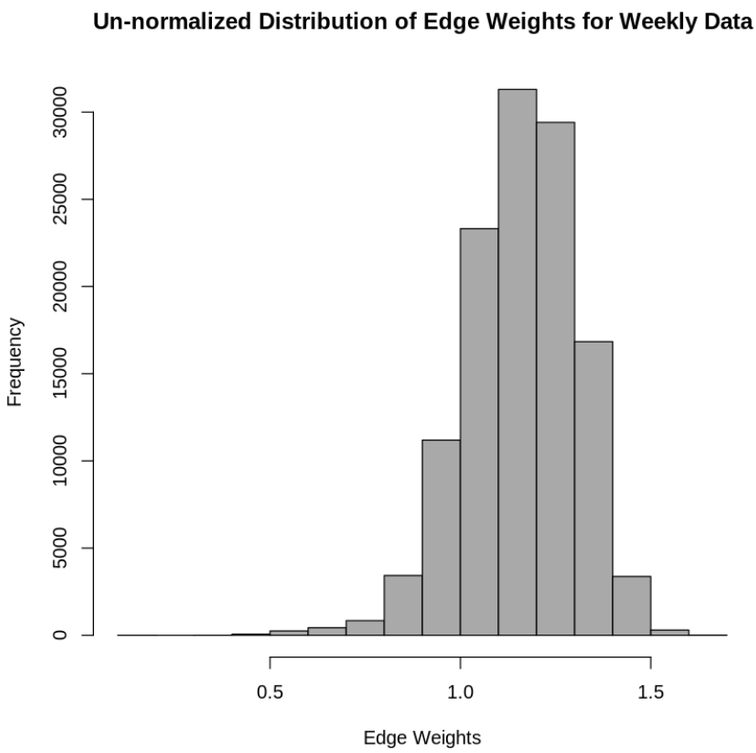


Figure 5: Histogram showing the unnormalized distribution of edge weights for Weekly data

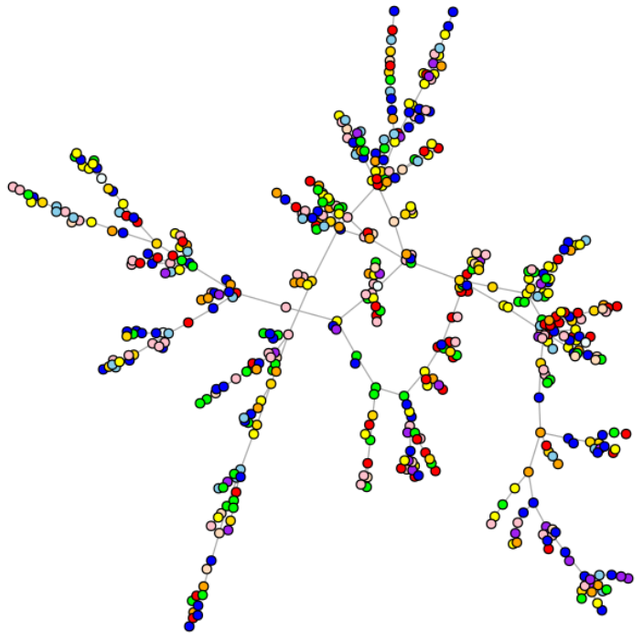


Figure 6: The Minimum Spanning Tree derived from the correlation for weekly data

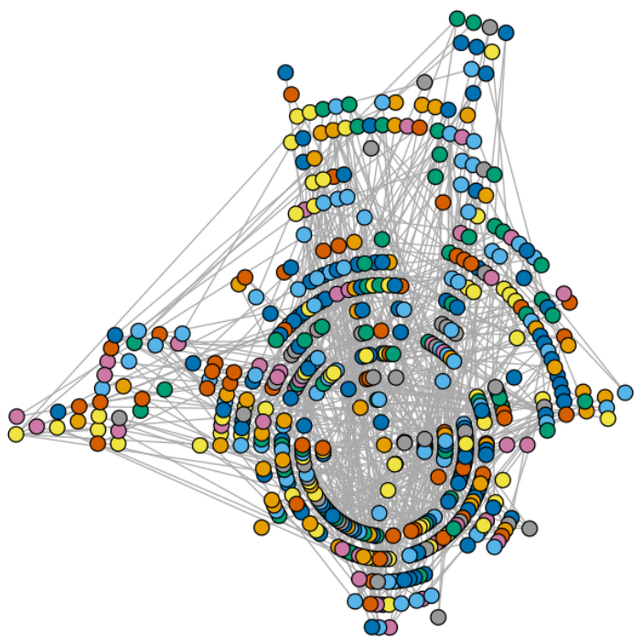


Figure 7: Circular MST / Another MST Representation for weekly data

- Technique 1: $\alpha = 0.73936556329849$

- Technique 2: $\alpha = 0.114610185736004$

The homogeneity for Weekly data: **0.608218622424519**
The completeness for Weekly data: **0.408419725767687**

Homogeneity metric of 60% indicates that 40% of the samples on an average are not from the assigned corresponding ground truth label. Completeness of 41% shows that 59% of the samples for a given class label are part of the same cluster.

At first look, the color distribution of the nodes in Figure 4 appears to be more dispersed than what we calculated in Question 3. Figure 2 appears to display grape-like structures, however this is not the case in Figure 4. To officially assess the performance of clustering on MST for Monday data, we repeat the processes in Question 4, and the result is 0.75095, which is consistent with the behavior shown in the figure. The omission of data can be used to understand the degradation.

As we can see, the omission of data has an effect on the Alpha values, Homogeneity and Completeness values too. This shows that the clusters have not been formed that efficiently and consist of data points from other clusters i.e. the clusters are non homogeneous. When all samples of kind c have been assigned to the same cluster k, the completeness equals 1. But we see that the value of completeness is less than half, that means that same samples of kind c have been assigned to different clusters. That means that the clustering has not occurred efficiently.

Question 7:

Inspecting the csv files provided, we can see that the stock price data has been automatically removed from weekends and national holidays. That is, we just need to determine if the relevant date is above 15 or not. We can successfully retrieve half of the month's data from three years of temporal arrays using R's weekday and same intrinsic functions. Using the same procedure described in Questions 2,3,4 and 5, we can obtain the clustering result based on MST for half the month's data, as shown in the Figure below.

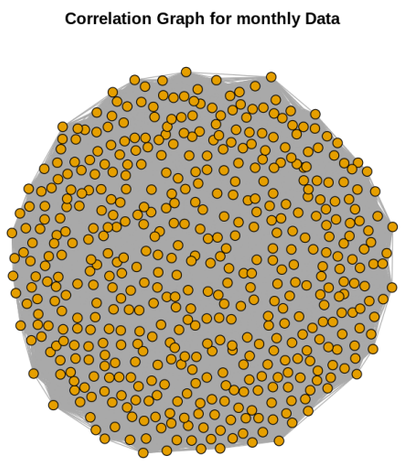


Figure 8: Correlation for Monthly data

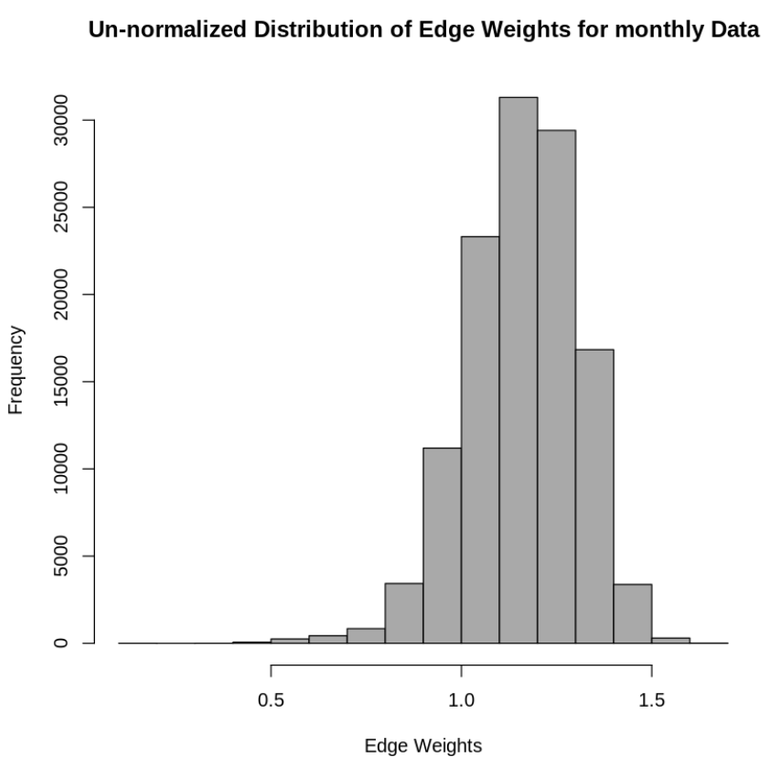


Figure 9: Histogram showing the unnormalized distribution of edge weights for Monthly data

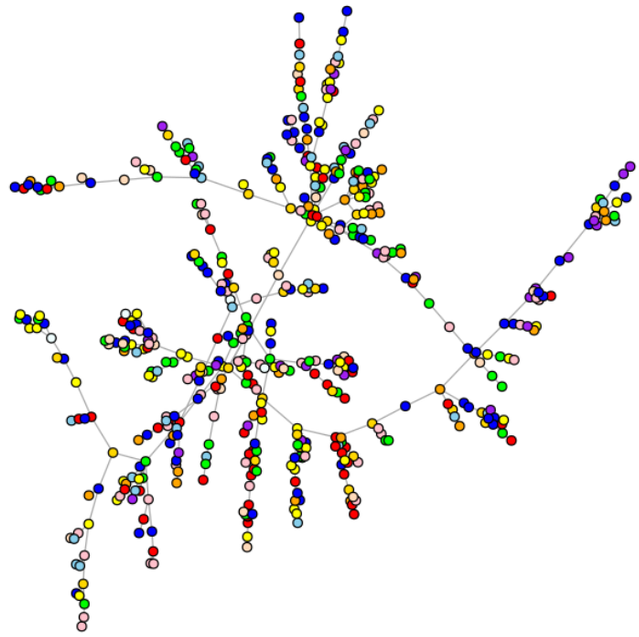


Figure 10: The Minimum Spanning Tree derived from the correlation for Monthly data

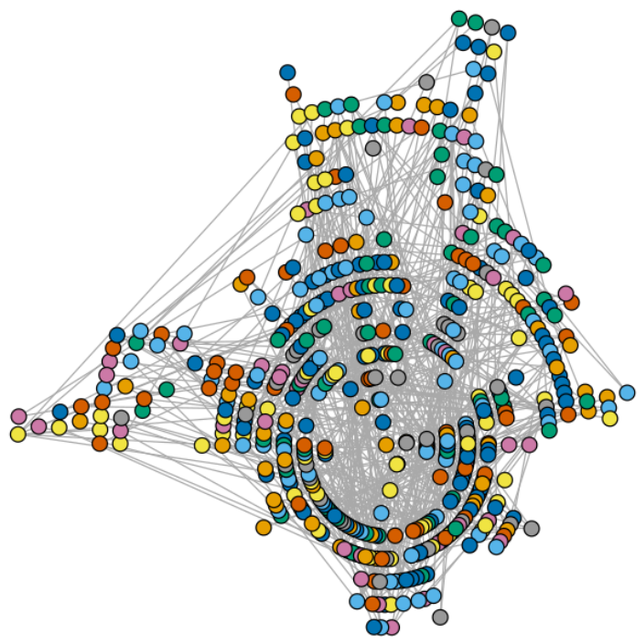


Figure 11: Circular MST / Another MST Representation for Monthly data

- Technique 1: $\alpha = 0.511868466898955$

- Technique 2: $\alpha = 0.114610185736004$

The homogeneity for Weekly data: **0.466300943858798**
The completeness for Weekly data: **0.313121789755227**

Homogeneity metric of 46% indicates that 54% of the samples on an average are not from the assigned corresponding ground truth label. Completeness of 31% shows that 69% of the samples for a given class label are part of the same cluster.

At first look, the color distribution of the nodes in Figure 4 appears to be more dispersed than what we calculated in Question 3. Figure 2 appears to display grape-like structures, however this is not the case in Figure 4. To officially assess the performance of clustering on MST for Monday data, we repeat the processes in Question 4, and the result is 0.75095, which is consistent with the behavior shown in the figure. The omission of data can be used to understand the degradation.

As we can see, the omission of data has an effect on the Alpha values, Homogeneity and Completeness values too. This shows that the clusters have not been formed that efficiently and consist of data points from other clusters i.e. the clusters are non homogeneous. When all samples of kind c have been assigned to the same cluster k, the completeness equals 1. But we see that the value of completeness is less than half, that means that same samples of kind c have been assigned to different clusters. That means that the clustering has not occurred efficiently.

Question 8:

A summary of the metric values is provided in the table below:

Table 1: Comparison of various metrics over the daily, weekly and monthly data

Data	**Alpha 1**	**Alpha 2**	**Homogeneity**	**Completeness**
Daily	0.8289	0.1141	0.6826	0.4792
Weekly	0.7393	0.1146	0.60821	0.40841
Monthly	0.51186	0.1146	0.4663	0.31312

- We notice that as the data is omitted the values for Alpha 1 (Technique 1), Homogeneity and Completeness reduces. This makes sense as there is lesser data to form clusters from. Daily data gave us the best results as we maximised the input data. We believe that if we had more data, then the cluster formation would have been better and the Homogeneity, Completeness values could have also been increased to a value > 0.8

- Clustering metrics do not decrease uniformly with decrease in data (i.e. daily $->$ weekly $->$ monthly).

- We notice that the value of Alpha 2 does not improve much nor does it degrade. This shows that the method is still not able to grasp the complete understanding for the fully connected graph. It probably requires more understanding of how the nodes are connected to each other.

- Alpha 1 is like an accuracy measure for the clustering result. We notice that the values for alpha 1 decrease as the data decreases (i.e. daily $->$ weekly $->$ monthly). This trend makes sense. Clustering algorithms depend on the data given to them. They work on the correlation between the data points, hence if more data is given, more correlation can be found and trends can be easily identified.

- A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. Homogeneity metric indicates that a certain number of the samples on an average are not from the assigned corresponding ground truth label. As we decrease the data, the number of samples on an average are wrongly clustered. This shows that we should increase the number of samples and not decrease. This makes sense intuitively as we mentioned above that we need more data so we can form correlations between them. From the data we can see that the values for homogeneity decreases as we move from daily $->$ weekly $->$ monthly.

- A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster. Completeness metric shows that a certain number of the samples for a given class label are part of the same cluster. We can see that the completeness score decreases, which means that the values which have higher correlation are clustered into one group.

- Alpha 2 values remain the same for Weekly and Monthly. Alpha 1, homogeneity and completeness values decrease as the data decreases (i.e. daily $->$ weekly $->$ monthly).

We think that the "Daily" granularity gives the best results when predicting the sector of an unknown stock. The Alpha 1 value is around 82.89%, which means that the accuracy is around 82.89%. Alpha 2 value helps us understand the fully connected graph, this value remains the same and does not contribute to the choosing of the granularity. As we know homogeneity and completeness should be as close to 1 as possible. As the data decreases, the values go away from 1. We know that the values of Homogeneity and Completeness are the highest for daily. This means that the cluster formation is right most of the times, and most of the data points are classified to the right cluster group.

Daily and Weekly both are much better measures for predicting than Monthly. We can probably add a different granularity measure. We could collect data thrice a day or hourly. This measure could drastically improve the results or it may also lead to overfitting the data.

Let's Help Santa!

Question 9:

Companies like Google and Uber have a vast amount of statistics about transportation dynamics. Santa has decided to use network theory to facilitate his gift delivery for the next Christmas. We went to "Uber Movement" website and downloaded data of Travel Times by Month (All Days), 2019 Quarter 4, for Los Angeles area. The file that we obtain is 'los-angeles-censustracts-2019-4-All-MonthlyAggregate.csv'. The dataset contains pairwise traveling time statistics between most pairs of points in the Los Angeles area. Points on the map are represented by unique IDs. To understand the correspondence between map IDs and areas, we downloaded Geo Boundaries file from the same website. This file contained latitudes and longitudes of the corners of the polygons circumscribing each area.
Below are the steps we followed:

- Extract December month data from the csv file.
- Merge the duplicate edges by taking an average of their weights.
- Maintain the largest connected component i.e., GCC of the graph.
- As an attribute to the appropriate vertex, add the mean of the coordinates of each region's polygon corners.

The nodes in the generated graph correspond to the locations, and the undirected weighted edges correspond to the mean traveling times between each pair of places. The number of nodes and edges in the graph G must be reported for this question.
The number of nodes: **2649**
The number of edges: **1003858**

Question 10:

We will construct a minimal spanning tree (MST) of the graph G in this question. Creating the MST is simple thanks to Python's *igraph* module. We just utilize the library's spinning *tree()* method. And we discover that there are **2649 nodes** and **2648 edges**. We can see from this conclusion that $E = V - 1$, which conforms to the feature of MST.
We examine the street addresses of the two ends of various edges. Unfortunately, this information does not provide exact street addresses; rather, it only includes Census Tract IDs for the Los Angeles region. As a result, locating specific street addresses without precise Latitude and Longitude information is extremely difficult. Our answer to this problem is to look up the Census Tract IDs online and see which ID relates to which area of Los Angeles. We can discover the equivalent location by matching the IDs on this webpage `https://data.lacounty.gov/widgets/rv2f-zsc7`. Also `https://geohub.lacity.org/datasets/enriched-la-county-census-tracts-2015/` maps the Census Tract into the actual Los Angeles map, allowing us to view the location of each Census Tract.
The table below represents the results obtained:

Census Track 1(Node 1)	Census Track 2(Node 2)	Area of Node 1	Area of Node 2
554001	554002	9421, Rosecrans Avenue, Bellflower	9020, Somerset Boulevard, Bellflower
461700	460800	Pasadena	Brookside Golf Club, Arrovo Woods, Pasadena
302201	302202	First United Methodist Church of Glendale, 134 North Kenwood Street, Glendale	East Colorado Street, Glendale
407101	407002	619 North Sunset Avenue, La Puente	652 Puente Avenue, El Monte
433401	433402	10468 Fern Street, South El Monte	2433 Continental Avenue, El Monte
543603	294410	Harbor Freeway, West Carson	828 Lomita Boulevard, West Carson
482001	530700	I-10 Metro ExpressLanes, Los Angeles	4225 Drucker Street, East Los Angeles
460800	460800	Brookside Golf Club, Arrovo Woods, Pasadena	Lower Arroyo Park, 415 Arroyo Boulevard, Pasadena
269100	217001	475 Smithwood Drive, Beverly Hills & Robertson	Airdrome, Robertson Boulevard, Los Angeles
011000	001901	1382 West Valley View Drive, Fullerton	2069 West Walnut Avenue, Fullerton

Table 2: Census Track IDs, their corresponding areas and Distance between them in Los Angeles

On average, the distance between nodes is around 0.7 mile. This makes logical sense since the least spanning tree is made up of edges that combine to provide the lowest weight/cost achievable. This indicates that the edges with the lowest weights will be selected. Because the weights represent the average journey time, a low weight indicates that the commute between two nodes is short, and thus the distance between the nodes is short. As a result, the MST is made up of nodes that are physically near to one another.

Question 11:

Assumption: The randomly sampled triangles are not unique.
In this question we are asked to determine what percentage of triangles in the graph (sets of 3 points on the map) satisfy the triangle inequality. We do not need to inspect all triangles, we estimate by random sampling of 1000 triangles.
In Euclidean geometry, **triangle inequality** is the theory that the sum of any two sides of a triangle is larger than or equal to the third side; in symbols, $x + y \geq z$. In essence, the theorem argues that the straight line is the shortest distance between two places. Another representation of the inequality is:

$$x + y > z; \quad y + z > x; \quad z + x > y$$

By using this inequality, it means that the graph has no shortcuts. In our scenario, the vertices will be the locations on the map, and we will ensure that the total of the lengths of any two sides of the triangle is more than the length of the third side, where the length is the edge's weight.
The percentage of triangles in the graph that satisfy the triangle inequality: **91.7%**

Question 12:

Using the graph G obtained above, we aim to discover an approximate solution to the Traveling Salesman Problem (TSP). The TSP is an undirected weighted network, with nodes representing cities and edges representing highways or streets. The edge weights reflect the distance between cities. The purpose is to find the quickest route from a city so that all other cities are only visited once and the tourist returns to the original location. The path is also known as a Hamiltonian cycle that satisfies the triangle inequality. Because the approach is NP-Hard, approximation methods are beneficial for reducing compute time for reaching the answer. To solve the TSP, we utilize the 1-approximation algorithm, which is as follows:

- Find the MST
- Create a multigraph from the MST (each edge is replaced by two directed edges).
- We need to build the embedded tour sequence, and to do this we need to locate the Euler cycle in the multigraph. (An Euler cycle is a cycle that only goes around each edge once)
- In the MST, look for each edge in the tour sequence. If the edge does not exist, use Djkstra's approach to find the shortest path between the nodes that include it.
- Calculate the approximate TSP cost as the sum of the weights of the tour sequence's edges (or shortest path cost if edge is absent in the MST). The MST cost may be used to approximate the ideal TSP cost since the optimal TSP cost (best feasible route) is always larger than the MST cost.

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

We know that MST Cost $\leq$ Optimal TSP Cost $\leq$ Approximate TSP Cost $\leq$ 2*MST Cost. Since the MST cost is a lower bound on the optimal TSP Cost, we use that in the calculation for ρ. The MST cost: **269084.6100000002**
The Approximate cost: **465542.44000000047**
The upper bound on the empirical performance of the approximate algorithm: **1.579951107931527**
Normalizing the above equation we obtain the below results: Thus, 1 $\leq$ Optimal TSP cost $\leq$ 1.58 $\leq$ 2

Question 13:

In this question, we will plot the path that Santa must have taken. We plot the trajectory using the mean coordinate information in particular. The x-axis of the graphic represents Longitude, while the y-axis represents Latitude.

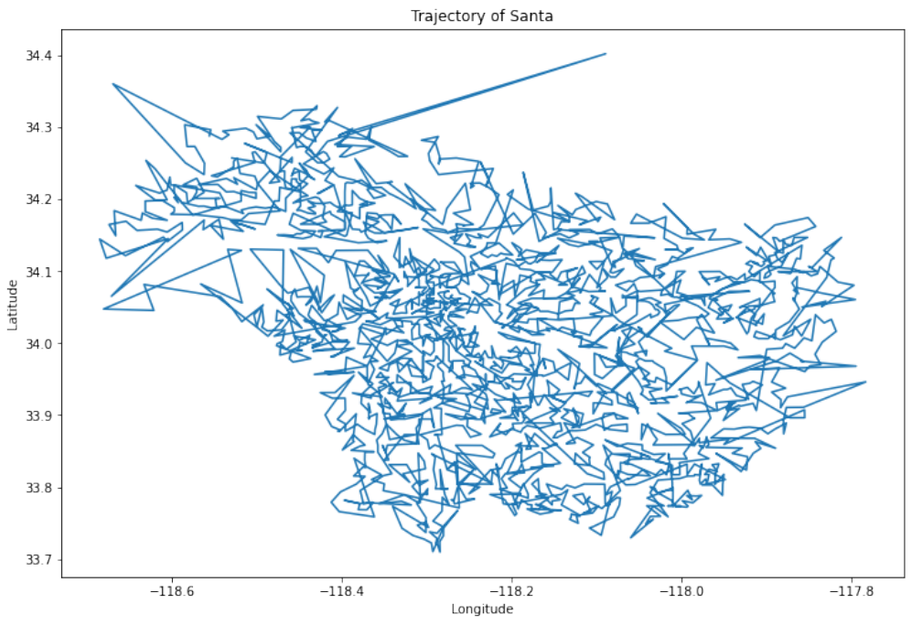


Figure 12: Plot showing the optimal trajectory Santa needs to take for gift delivery

We have plotted the same trajectories and points on the LA map in the jupyter notebook.

Question 14:

Plot the road mesh that you obtain and explain the result. Create a graph $G_{\triangle}$. whose nodes are different locations and its edges are produced by triangulation.

In this question, we estimate the map of roads without using road information, with the help of Delaunay triangulation (DT) algorithm. DT is an algorithm for a collection of discrete points in which no point in the set of points lies within any triangle's circumcircle. In other words, DT(P) is such that no point in P is inside the circumcircle of any triangle in DT(P), given a set P of discrete points. DT aims to maximize the minimum angle of all the angles of the triangles in the triangulation,which in turn avoids the narrow triangles. This is effective for discovering more pragmatic road designs from a set of locations. Actual road structures follow geometric patterns similar to those produced by the triangulation algorithm. The figure in this question shows the road mesh generated using DT for $G_{\triangle}$.

Our observations from this road mesh are -

- The DT algorithm has mostly extracted the road structures of LA in the central and south west regions. We also see roads going over oceans - which is not a real life scenario.
- That can be explained by the DT algorithm that tries to avoid narrow triangles by ensuring no point in the set of points lies within the circumcircle of any triangle. As a result most triangles don't have acute angles.

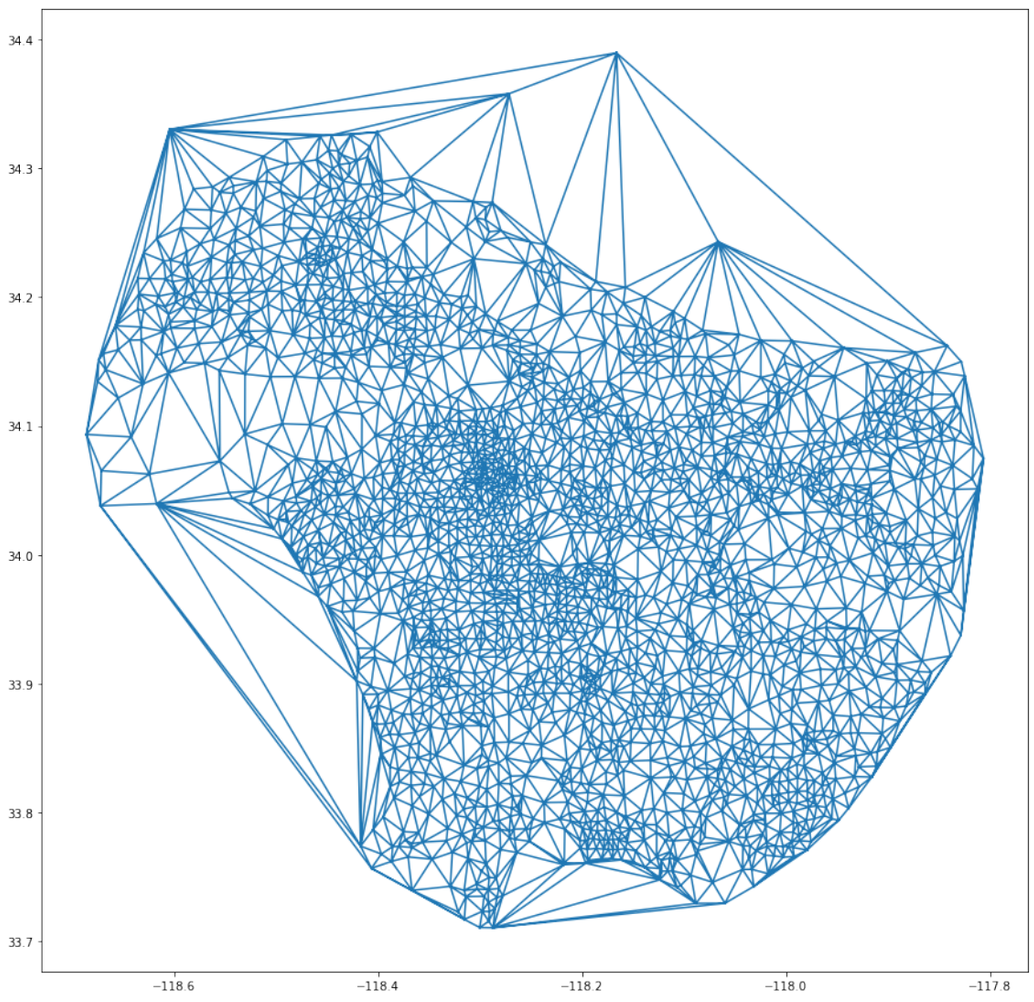


Figure 13: Road mesh of Los Angeles using Delaunay triangulation

Question 15:

Using simple math, calculate the traffic flow for each road in terms of cars/hour. Report your derivation.

These are the assumptions stated in the question. Each degree of latitude and longitude = 69 miles, Car length = 5 m = 0.003 mile , Cars maintain a safety distance of 2 seconds to the next car , and Each road has 2 lanes in each direction. Assuming no traffic jam, consider the calculated traffic flow as the max capacity of each road.

The unit of velocity is in miles per hour. The velocity of car between two coordinates can be derived using Pythagoras theorem:

$$\text{Velocity of car} = \frac{69}{\text{Mean Travel Time}} \times \sqrt{(\text{Lattitude}_{\text{point 1}} - \text{Lattitude}_{\text{point 2}})^2 + (\text{Longitude}_{\text{point 2}} - \text{Longitude}_{\text{point 2}})^2}$$

Traffic flow is given by the below equation for cars/hour - taking into consideration the change from second to hour -

$$\text{Traffic Flow (cars/hour)} = \frac{60 \times 60}{\text{Mean Travel Time}} \times \text{Total number of cars on the road}$$

Total number of cars taking into consideration bidirectional flow for the above equation is -

$$\text{Total number of cars on the road} = \frac{2 \times \text{Total distance}}{\text{Gap}}$$

Gap will be the given gap between the cars and length of each car, and the total distance is -

$$\text{Total distance} = \frac{\text{Velocity of car} \times \text{Mean Travel time}}{60 \times 60}$$

$$\text{Gap} = 0.003 + \frac{2 \times \text{Velocity of car}}{60 \times 60}$$

Putting all these together, the final equation is -

$$\text{Traffic flow (cars/hour)} = \frac{3600 \text{ x velocity of car}}{5.4 + \text{velocity of car}}$$

Question 16:

Calculate the maximum number of cars that can commute per hour from Malibu to Long Beach. 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?

Source coordinates (in Malibu): [34.04, -118.56], Destination coordinates (in Long Beach): [33.77, -118.18].

We determine the maximum number of cars per hour from Malibu to Long Beach using the formulae presented in the previous question and the max-flow technique. The maximum number of cars from Malibu to Long Beach is 109442 cars/hour. Between the two points, there are 5 discontinuous pathways. When we look at the map near the two place maps, our answer makes more sense.

The number of edge-disjoint pathways between two nodes on a graph is equal to the minimum of the number of edges outgoing and arriving to each node. We can observe from the plots in this image that Malibu and Long Beach each have five outgoing and incoming edges, which makes the number of edge-disjoint pathways as 5.

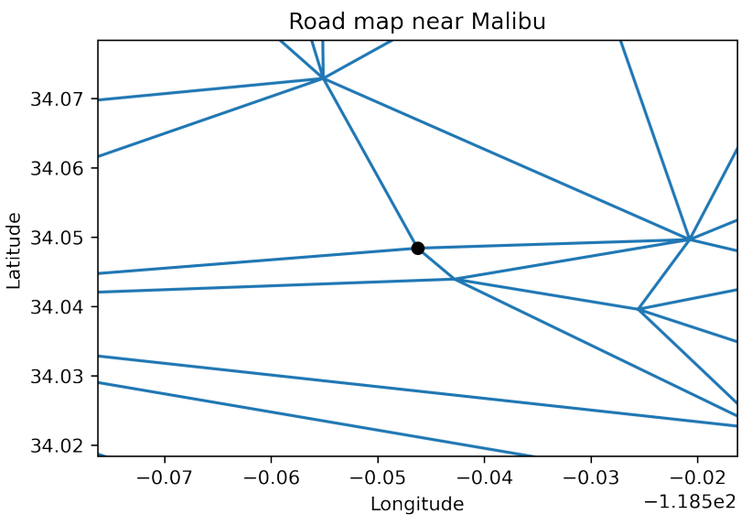


Figure 14: Road map near Malibu

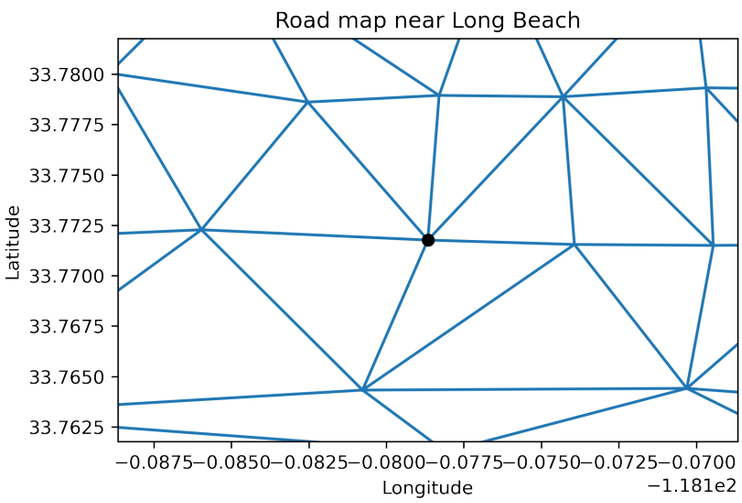


Figure 15: Road map near Long Beach

Question 17:

Plot $G_{\triangle}$ on actual coordinates. Do you think the thresholding method worked?

In this question we apply a pruning method - In $G_{\triangle}$, there are a number of unreal roads that could be removed. For instance, we notice some unreal links along the concavities of the beach, as well as in the hills of Topanga. We apply a threshold on the travel time of the roads in $G_{\triangle}$ to remove the fake edges. The resulting graph is called $G_{\triangle}$.

We prune our graph to remove non-existent and unreal roads, particularly those crossing the oceans and going over mountains. To do so, we apply a threshold on the travel time of the roads on $G_{\triangle}$ .The thresholding approach, in theory, reduces big edges produced by the DT algorithm in order to accommodate triangles into the circumcircle. Because other edges go over numerous nodes of lower distances, the mean travel time for this edge would be longer than for other edges. We increase the threshold by taking distance into consideration, allowing us to prune out places that are close but have a long journey time. This ensures that we do not cut out roads between points that are actually far apart.The figure below shows the resulting road mesh map plot on actual coordinates.

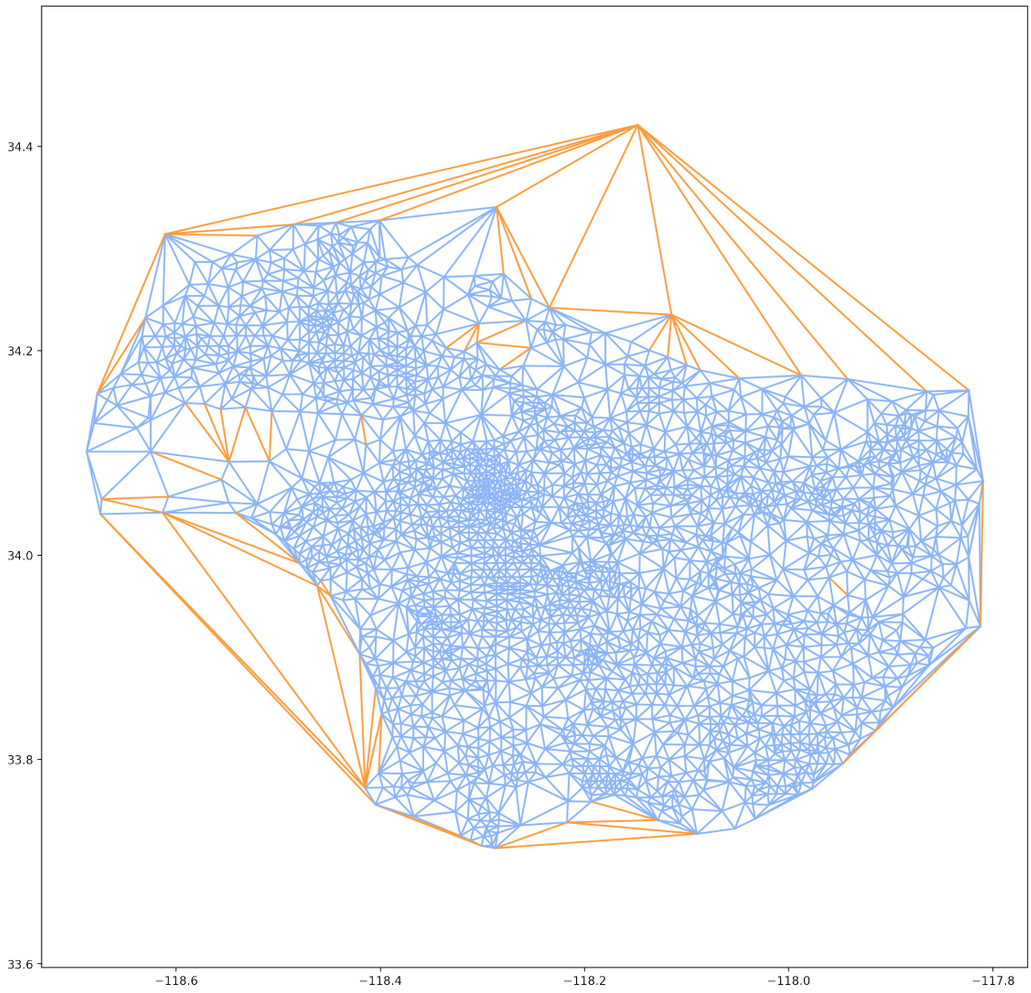


Figure 16: Pruned road mesh map

The threshold used is 800. We compare the weights of the edge attributes to check if they are greater than 800. The ones greater than 800 are removed during graph formulation. The edges removed are highlighted in orange in this figure.

Yes, we believe this thresholding strategy was successful. There are less lengthy edges be-

tween neighboring nodes. Long, non-existent routes over the ocean, as well as routes over the Topanga mountain ranges, have vanished, as may be seen. This proves that the thresholding strategy was effective. The road mesh network now matches the actual Los Angeles road plan, with most of the routes running down the coast rather than via mountains or beaches.

Question 18:

Now, repeat question 16 for $G_{\triangle}$ and report the results. Do you see any changes? Why?

The maximum number of cars from Malibu to Long Beach is 238915 cars/hour. This makes intuitive sense, as we can assume some edge that was previously present in the graph was connecting 2 nodes. After pruning, that edge was removed and the car probably had to take an alternative path. So, the max flow will in this case will change.

The number of edge-disjoint pathways between two nodes on a graph is equal to the minimum of the number of edges outgoing and arriving to each node. We can observe from the plots in this image that Malibu and Long Beach each have five outgoing and incoming edges, which makes the number of edge-disjoint pathways as 5. Also, from the road maps we can observe that the road network is less dense around the two areas, showing the effect of pruning even though the number of edge disjoint pathways remains the same.

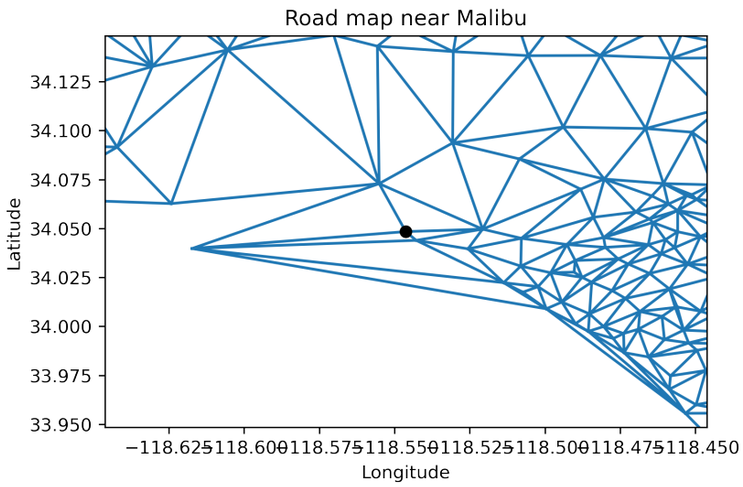


Figure 17: Road map near Malibu

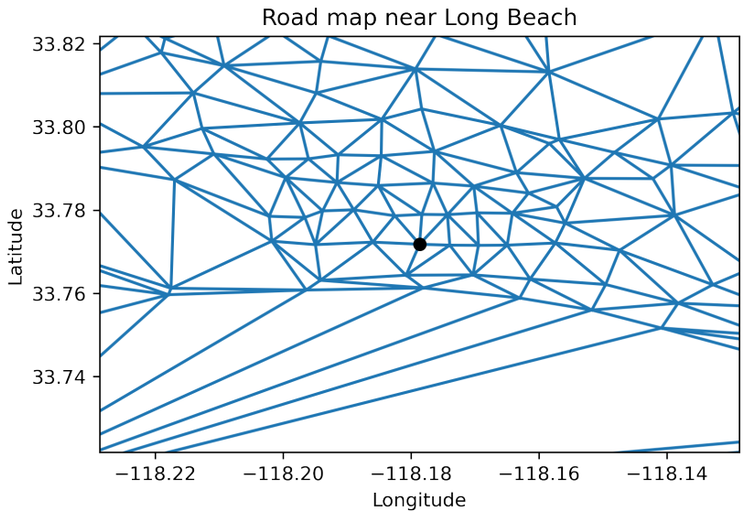


Figure 18: Road map near Long Beach

Question 19:

Strategy 1 (geo - distance, static) -
In this question we first find the euclidean distance between the nodes using the euclidean distance formula given below.
$euclidean\_distance(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$

The shortest path between any two given nodes is found using dijkstra algorithm. The steps for the same are given below:

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest-path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
- While sptSet doesn't include all vertices
  - Pick a vertex u which is not there in sptSet and has a minimum distance value.
  - Include u to sptSet.
  - Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

The extra distance is found by subtracting euclidean distance from the shortest path distance. Now, to find the top 20 paths, we actually took top 40 paths in the matrix as (A->B and B->A are considered as different paths by the code but we consider them as the same paths). The top 20 paths are basically the paths where extra distance is maximum. Now these top 20 paths that we found are added to a variable. The source and destination nodes of all edges are printed and can be observed in figure 19.

The algorithm used for strategy 1 is summarised below:

- Find euclidean distance between all source, destination pairs.
- Find the shortest path between source - destination pairs using dijkstra algorithm.
- Find extra distance by subtracting euclidean distance from shortest paths.
- Find the top 20 edges. Print the source and destination for each edge and also plot the edges in the graph.

```
[41, 2419]
[44, 2419]
[48, 2419]
[146, 1783]
[393, 2419]
[968, 1510]
[988, 1510]
[988, 1678]
[1005, 1510]
[1005, 1678]
[1005, 1511]
[1699, 1783]
[1699, 1860]
[1782, 2416]
[1783, 2413]
[1783, 2416]
[1859, 2416]
[1860, 2413]
[1860, 2416]
[2159, 2419]
```

Figure 19: Source and Destination nodes of new edges

The new edges are also plotted on the graph which are highlighted in red color.

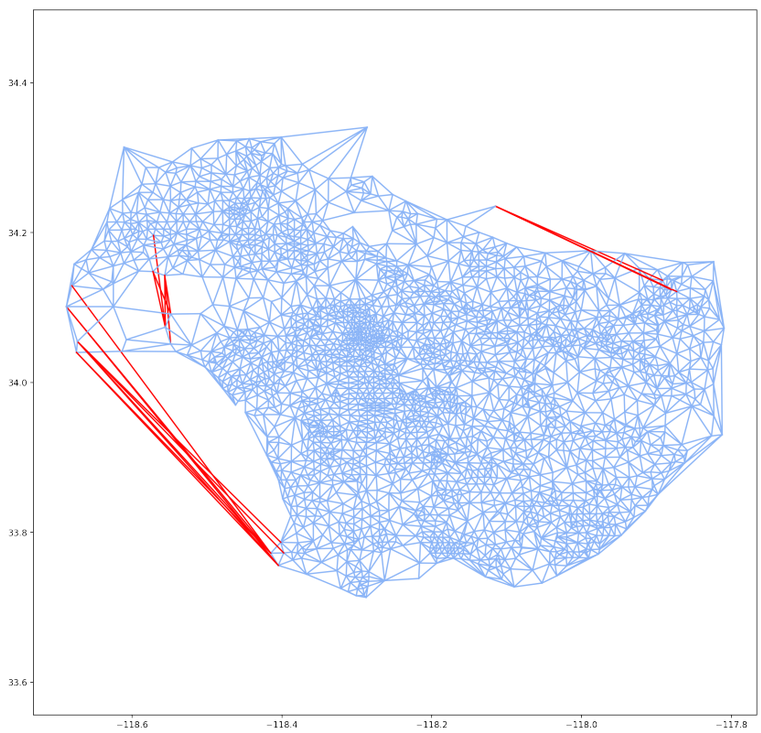


Figure 20: Plot of roads with new edges

- The time complexity of any algorithm is decided by the slowest step in the algorithm. We are using shortest path function in networkx library in python. Dijkstra algorithm itself has a time complexity on $O(n^2)$ due to two nested for loop. In our algorithm the shortest path finding function is inside a for loop which makes the time complexity of our algorithm $\mathbf{O(n^3)}$, where n is the number of nodes.
- As can be seen from the graph most new edges created for strategy 1 are in the ocean regions.
- Most edges start around the points 1005, 1783 and the destination points are also closer which makes most of the new roads along the same regions.

Question 20:

Strategy 2 (geo - distance, static, with frequency) -
In this question we introduce frequency of travel between each source destination pair. Essentially we want to consider a real world situation where some edges are preferred over others i.e. some roads are more in-demand than others.
The algorithm used for this question is as follows:

- Compute extra distance using shortest path and euclidean distance between source distance pairs.
- Multiply this extra distance by a random number between 1 to 1000 which will be considered as frequency of travel. This will be weighted extra distance.
- Using this weighted extra distance find the top 20 pairs of edges.
- Print source and destination of new edges. Plot the new edges in the graph(marked in red).

The source and destination corresponding to each new edge can be found in figure 21, where the plot of the graph with new edges marked in red can be found in figure 22.
The time complexity of this algorithm is $O(n^3)$, where n is the number of nodes. As we are again using Dijkstra algorithm to find the shortest path which is inside a for loop. And this would be the slowest step in the algorithm.

```
[41, 2419]
[223, 2419]
[225, 2420]
[285, 2419]
[381, 2420]
[382, 2420]
[952, 1510]
[954, 1678]
[968, 1678]
[969, 1678]
[971, 1678]
[988, 1510]
[1699, 1860]
[1783, 2413]
[1854, 2419]
[2149, 2419]
[2155, 2419]
[2158, 2419]
[2419, 2599]
[2419, 2420]
```

Figure 21: Source and Destination nodes of new edges

- Frequency of travel is useful information as we can see in the graphs the roads across ocean is significantly reduces and more roads are added in the land regions.

- Some roads are added across mountain regions, while some connect mountain regions to the land regions.
- There are more roads in the northeast part of the map. From the co-ordinates also it can be seen that the roads cover more ground than strategy 1.
- The roads with source 1005 are removed and there is more variation in source and destination nodes.
- We see that a lot of roads have destination as 2419 in Strategy 2, whereas the results are a little more varied in Strategy 1.

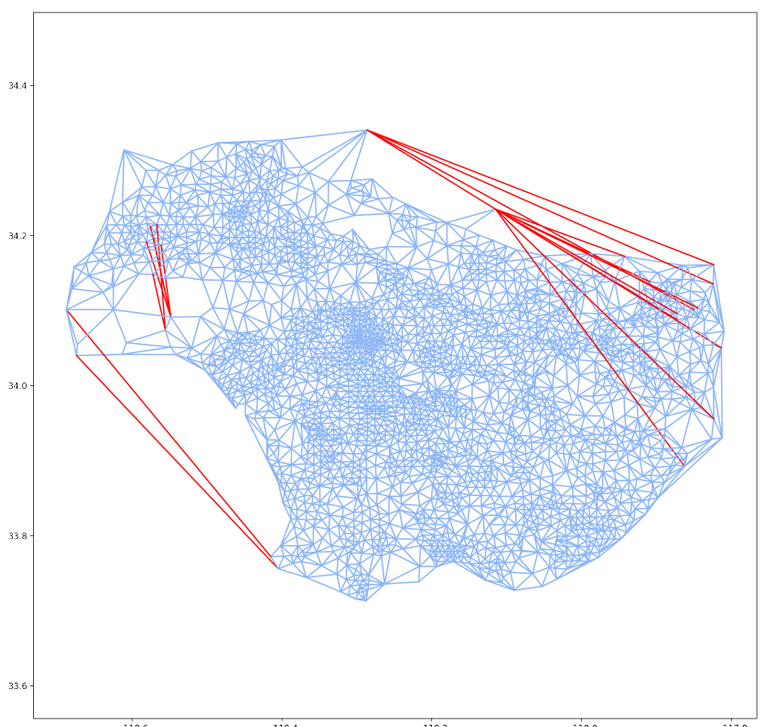


Figure 22: Plot of roads with new edges

Question 21:

Strategy 3 (geo distance, dynamic) - In the above strategies, we are creating all roads at the same time. In this strategy, we will create the roads one by one. This makes more intuitive sense as we build other new roads, based on the recently built roads. In this way we can avoid building roads that are unnecessary.
The algorithm followed in this question is (Repeated 20 times) -

- Compute extra distance between all the pairs in the graph.
- Create a road between the pair with highest extra distance and update the graph.
- Print the source and destination of this new edge.

The formula to calculate each of the distance matrices, and select the indexes with the largest value is mentioned in question 19. The way this algorithm varies from strategy 1 is that, the computation is carried out edge wise. So every new edge created is added to the graph. The creation of the next new edge makes use of this graph (with the previous new edge added as well). This is repeated 20 times in our case for 20 new edges.

For this reason the algorithm also takes more time to run when compared to others. The Time complexity of the algorithm is $O(n^3)$, where n is the number of nodes.

The source and destination of the 20 new edges are shown below -

```
[44, 2419]
[120, 688]
[120, 1679]
[144, 2565]
[285, 2419]
[356, 1679]
[382, 2420]
[657, 1679]
[1005, 1510]
[1672, 2627]
[1672, 2376]
[1700, 1783]
[1717, 2419]
[1783, 2416]
[1875, 2416]
[1956, 2419]
[2239, 2419]
[2247, 2420]
[2419, 2420]
[2420, 2461]
```

Figure 23: Source and Destination nodes of new edges

The below figure shows the new edges (marked in red) added to the pruned graph by the strategy of dynamic geo distance edges between the source and destination coordinates reported above.

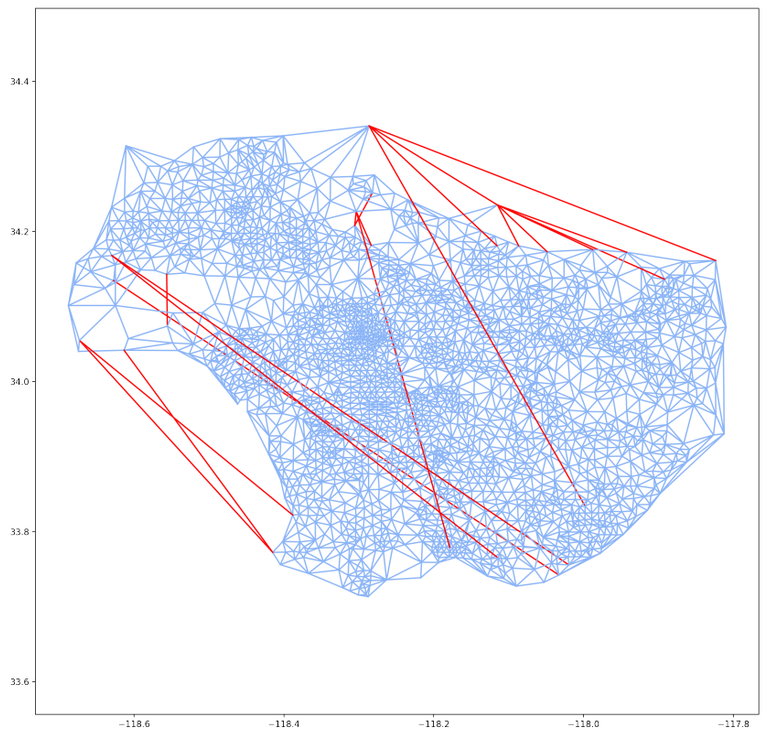


Figure 24: Plot of roads with new edges

The results obtained in this method make intuitive sense -

- We will make use of the new road construction to decide how the other new roads are built, instead of building all the new roads at once. In this way, we can calculate if more different edges are needed to some coordinates.
- This is noticed in the source and destination coordinates. When compared to the previous 2 strategies, more roads are added to the destination with coordinates 2416, and 2420, and less roads to 2419. The road from 2419 to 2599 is also removed in this method.
- This is a dynamic strategy which helps build roads in a more optimised way.
- However, we still have roads built across oceans which we had removed using pruning. So, this is not a very accurate real life strategy in that consideration. Most of the new roads added connect the east and west LA.

Strategy 4 (Travel time, static) - This strategy will take travel time into consideration, unlike the previous strategies which optimised only the travel distance. We want to reduce the maximum extra travelling time. The extra travelling time between 2 locations is the difference of the shortest traveling time and the straight line travel time. The straight line travel time includes the euclidean distance.

extra_time(v,s) = travel_time_of_shortest_path(v,s) - euclidean_distance(v,s)/travel_speed(v,s)

travel_speed(v,s) = distance_of_shortest_path(v,s) / travel_time_of_shortest_path(v,s)

The algorithm followed for this strategy is -

- Use the coordinates of v and s to get the euclidean distance between them.
- Calculate the extra time between all pairs of points based on the above formula
- The top 20 pairs with highest extra time will be the new edges we suggest.
- Print the source and destination of these pairs

The distances are calculated using the same algorithm that was used in strategy 1.

The Time complexity of the algorithm is $O(n^3)$, where n is the number of nodes. The multiplying constant in this algorithm is greater than the rest, and hence it takes a lot more time for these 'for loops' to execute.

The source and destination of the 20 new edges are shown below -

```
[47, 2419]
[385, 2419]
[389, 2419]
[393, 2419]
[1005, 1511]
[1699, 1860]
[1699, 1859]
[1699, 1783]
[1782, 2416]
[1783, 2413]
[1783, 2416]
[1857, 2416]
[1859, 2416]
[1860, 2416]
[1882, 2416]
[2049, 2419]
[2144, 2419]
[2159, 2419]
[2162, 2419]
[2163, 2419]
```

Figure 25: Source and Destination nodes of new edges

The below figure shows the new edges (marked in red) added to the pruned graph by the strategy of static travel time edges between the source and destination coordinates reported above.

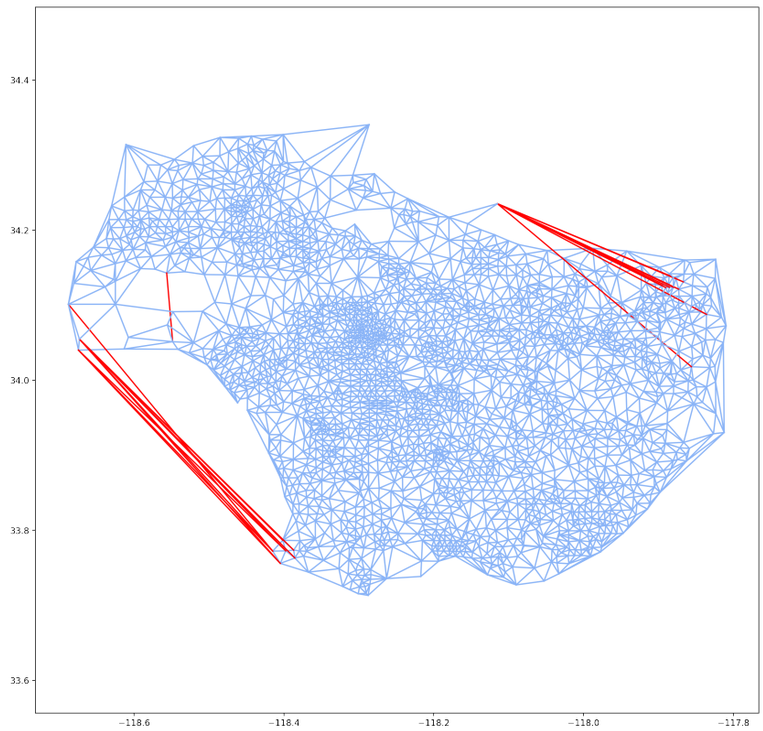


Figure 26: Plot of roads with new edges

Summarising the results of this strategy -

- In this strategy new edges are added based on the highest travel time. All roads are added at once in a static manner. This will make sense in the scenario when our highest travel time road indexes are in the different sub regions of LA. So, in this case adding all roads at once will give more perception.
- The number of roads across the ocean is a lot more in this strategy when compared to the previous methods. Also, a lot of new roads are added in the regions close to the hill. There are very less new roads interconnecting the other regions of LA.
- Adding new roads across the hill near the land regions makes sense because they were removed during pruning.
- Most of the new roads added in this method have the destination coordinate of 2419, and 2416.

Question 22:

Strategy 5 (Travel time, dynamic) - Similar to strategy 3, this time we will create roads one by one while optimizing the travel time (instead of the distance).

The algorithm followed in this question is (Repeated 20 times) -

- Compute extra time between all the pairs in the graph.
- Create a road between the pair with highest extra time and update the graph.
- Print the source and destination of this new edge.

The formula to calculate extra time based off the euclidean and shortest distance is described in Strategy 1 and 4.

The Time complexity of the algorithm is $O(n^3)$, where n is the number of nodes, with each computation being added back to the graph for next edge calculation. The time constant with the complexity in this strategy is high which explains the greater run time when compared to other strategies.

The source and destination of the 20 new edges are shown below -

```
[22, 2419]
[42, 2419]
[47, 2420]
[118, 2419]
[120, 1679]
[285, 2420]
[311, 2419]
[385, 2419]
[386, 2420]
[512, 2420]
[555, 2420]
[1004, 1678]
[1005, 1511]
[1700, 1783]
[1783, 2416]
[1875, 2416]
[2186, 2419]
[2232, 2419]
[2419, 2579]
[2419, 2420]
```

Figure 27: Source and Destination nodes of new edges

The below figure shows the new edges (marked in red) added to the pruned graph by the strategy of dynamic travel time edges between the source and destination coordinates reported above.

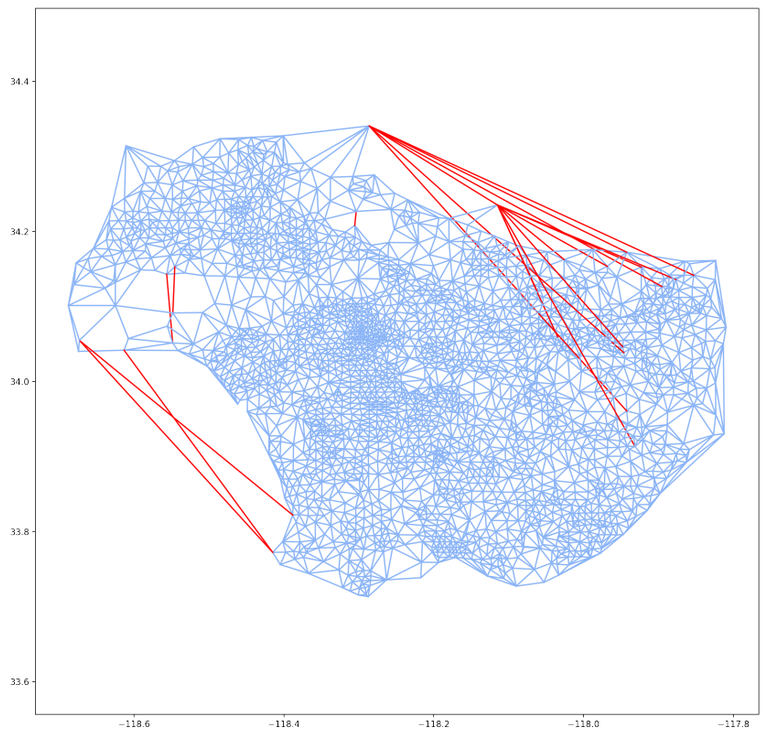


Figure 28: Plot of roads with new edges

Summarising the results of this strategy -

- New edges are added based on the highest travel time in a dynamic fashion. This is different and advantageous to the previous method in the situation when we have to add roads in overlapping sub regions of LA. Adding roads one by one will make intuitive sense.
- The number of roads across the ocean is lesser than Strategy 4 , but a lot of roads are added in the hill regions of LA. Some of them going into the deeper areas on LA. The way these roads vary from Strategy 4 is that, the roads added around the hill regions are more spread out. They connect to a lot of different destination coordinates from a single source.
- Quite a lot of these roads are in the land regions of LA. The method tries to make up for the roads that were removed using the pruning threshold.
- 2419 and 2420 still remains the most accessed coordinate in this method as well.

Question 23:

a) 1 vs 2 - compare and analyze the results. which is better? why?

- In terms of how the Strategies work in the real world, we believe Strategy 2 is better. Strategy 2 takes into account the real world situation were some edges are preferred over others.
- We also notice some of the paths are same in both the Strategies. This shows that some of the new roads are actually the most optimal ones and also the edges for those paths are the most preferred edges. The source 41, 968 are some common points for both strategies. The destination 2419 is common for both the Strategies. We see that a lot of roads have destination as 2419 in Strategy 2, whereas the results are a little more varied in Strategy 1.
- There are more roads across the ocean in Strategy 1 when compared to Strategy 1. It's physically not feasible to build long roads across the ocean. They will take more time, effort and money to build.
- Both strategies have some pros and cons. But if we have to build roads in real life, we believe that Strategy 2 will be a better choice.

b) 1 vs 3 - compare and analyze the results. which is better? why?

- The output of Strategy 3 and Strategy 1 is quite different. We see that the dynamic method for building roads gives us a very different answer when compared to the other static strategies. This makes sense intuitively.
- The dynamic method for road mapping is advantageous in some cases and not all. Dynamic road mapping can help when we have to build roads for the same regions. It would not make sense to have 2-3 roads build for 1 region when the whole region can be covered using 1 road.
- There are more inter region roads in Strategy 3 than Strategy 1. We see there are less roads across the ocean in Strategy 3 than 1. This helps Strategy 3 be more realistic than Strategy 1.
- When compared to the previous 2 strategies, more roads are added to the destination with coordinates 2416, and 2420, and less roads to 2419. The road from 2419 to 2599 is also removed in this method.
- Strategy 3 is more useful than Strategy 1 to build roads, as it helps to build more realistic roads.

c) 1 vs 4 - compare and analyze the results. which is better? why?

- Strategy 4 takes into account the travel time, which Strategy 1,2 & 3 don't. As we know, travel time is a huge factor in deciding which road to travel by. We notice a few similar edges between Strategy 1 and Strategy 4. A similar point was mentioned in the previous subpart.
- Run time for strategy 4 is more when compared to the previous Strategies. The multiplying constant in this algorithm is greater than the rest, and hence it takes a lot more time for these 'for loops' to execute.
- In this strategy new edges are added based on the highest travel time. All roads are added at once in a static manner. This will make sense in the scenario when

our highest travel time road indexes are in the different sub regions of LA. So, in this case adding all roads at once will give more perception.

- The number of roads across the ocean is a lot more in this strategy when compared to the previous methods. Also, a lot of new roads are added in the regions close to the hill. There are very less new roads interconnecting the other regions of LA
- Most of the new roads added in this method have the destination coordinate of 2419, and 2416, similar to that in strategy 2.
- We believe that Strategy 4 is better than Strategy 1 as it takes into account the time of the travel and helps us more in building the roads in a real world scenario even though there are a high number of roads that traverse the ocean.

d) 1 vs 5 - compare and analyze the results. which is better? why?

- We notice the similarities in path between Strategy 2, 3 & 5. Strategy 5 seems like a compilation of the strategies 2 and 3. Which is a very interesting inference!
- Run time for strategy 5 is more when compared to the previous Strategies. The multiplying constant in this algorithm is greater than the rest, and hence it takes a lot more time for these 'for loops' to execute.
- We believe Strategy 5 may be a better option to use in the real world scenario. The roads get optimised depending on the travel time. And as mentioned above, the travel time is a huge factor to take into account while traveling.
- Dynamic strategy helps us understand whether a road is really needed or not. We obviously want to optimize time, but we should also take into account the effort that goes into building the road. We should make sure that the built road is getting used and the effort of the workers does not go to waste.
- Strategy 5 is still better than Strategy 1.

e) statics vs dynamic - Considering we want to improve the overall road network by constructing new roads, is either of them the optimal strategy?

- if yes, which one and why?
  No, none of the strategies are optimal. We can see that in both the strategies there are roads built across the oceans and the mountain regions. Can we really build roads across the ocean and mountain regions? Is that the best way to use labor? The answer for the 1st question is Maybe, but the answer to question 2 is definitely a NO. If the coordinates for source and destination don't overlap in the subregions then static is better because we can construct them all at once. But this is not always the situation in real life. There will be a lot of source and destination points that fall into the same subregion and this is when the dynamic system of building roads helps.
- if not, what would be a better strategy to construct roads and is it optimal? What is the time complexity?
  A better strategy would be to use both Static and Dynamic system of building roads. If the coordinates for source and destination don't overlap in the subregions then build statically. But if source and destination don't fall into the same subregion then build the road dynamically. Usually both the strategies together and looking at the terrain of the place, we can build the roads.

f) Open ended question : Come up with new strategy. You are free to make any assumptions. You don't have to necessarily optimize traveling time or distance, you can come up with any constraints. justify your strategy.
A few inferences:

- We notice that for all strategies roads are built across oceans. But Strategy 3 covers more ground than the other strategies.
- Strategy 4 and 1 roads are not built across mountains but there are roads across the ocean.

To build a road we need to take into account the below points / have the below mentioned constraints:

- Take travel time into consideration because its calculated from speed like we did for Strategy 4 and 5.
- the source and destination points across the coast are not present. This will help us avoid the roads that are built across the ocean.
- We need to take into account that roads are not built across mountains and roads.

To build an optimised system of road, we should take into account the best properties from all strategies. We should use distance, time of travel and frequency of the edge points to design the road system. The time complexity of such a system will be $O(n^3)$ similar to the other Strategies.

Question 24:

Define your Own Task

Task 1:

In this section, we investigate the feasibility and accuracy of various approximation techniques for solving the TSP problem using the World TSP dataset. On the World TSP dataset, we compare the performance of these algorithms against the performance of the 1-approximate approach. Several algorithms have been developed in an attempt to approach the optimal solution of TSP problems:

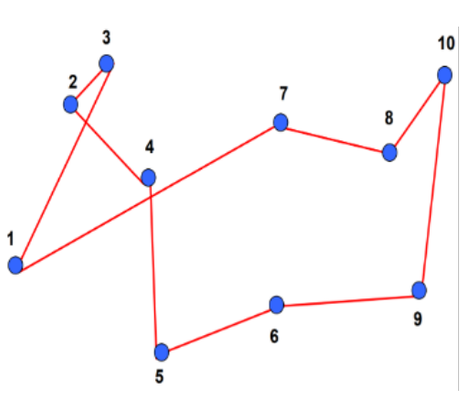
- Lin-Kernighan (LK) heuristics
- Randomized improvement (tabu search, 2-opt, 3-opt, cross entropy, ant colony, simulated annealing)
- Nearest neighbor (greedy algorithm)

There are approximate techniques that discover the best solution in addition to approximation approaches; however, the calculation time restricts the amount of TSP datasets to a significantly small number (about 300-500 cities):

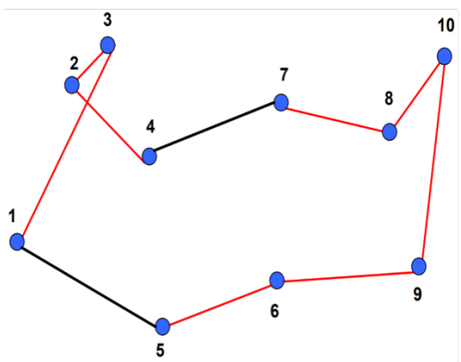
- Progressive improvement techniques (Linear programming)
- Branch-and-bound
- Branch-and-cut

We will investigate approximation techniques, especially simulated annealing and the 2-opt heuristic, in solving TSP datasets supplied by the University of Waterloo `http://www.math.uwaterloo.ca/tsp/world/countries.html`.

The first algorithm we implemented was a 2-opt algorithm. The goal of this local search technique is to continuously improve the cost of a Hamiltonian path by deleting any crossings of the path. This is accomplished by first producing a random tour and then using a node inversion approach on edge crossings:



(a) Input



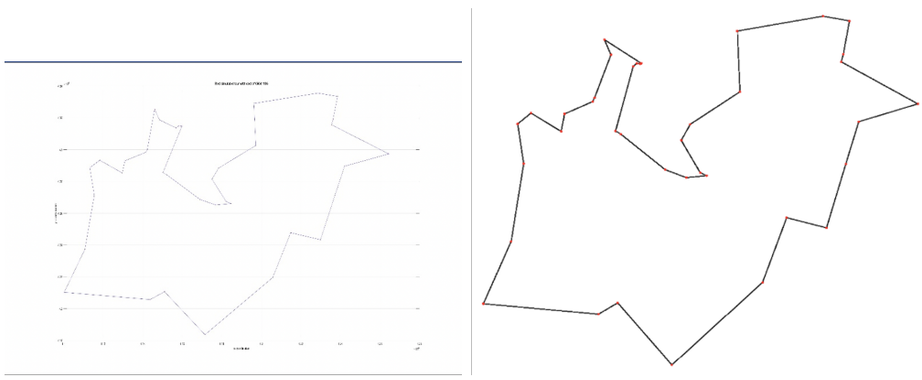
(b) Output we get after running 2 opt algo. multiple times

Figure 29: 2-opt Algorithm Output

As we can see from the figure above, the 2 opt algorithm converges after 15-20 runs and we obtain the output in the above figure. We notice that there were 2 paths that got created by the algorithm which are actually optimal paths. The algorithm is quite quick and fairly simple. We now test the accuracies of the 1-approximate, 2-opt algorithms on a set of 3 countries generated from the World TSP dataset:

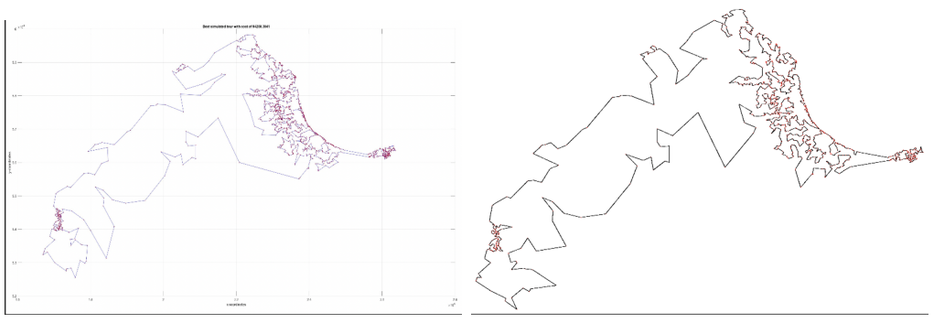
- Djibouti (38 cities)
- Oman (1979 cities)
- Canada (4663 cities)

The choice of these countries is to test the accuracy over a wide range of graph sizes.



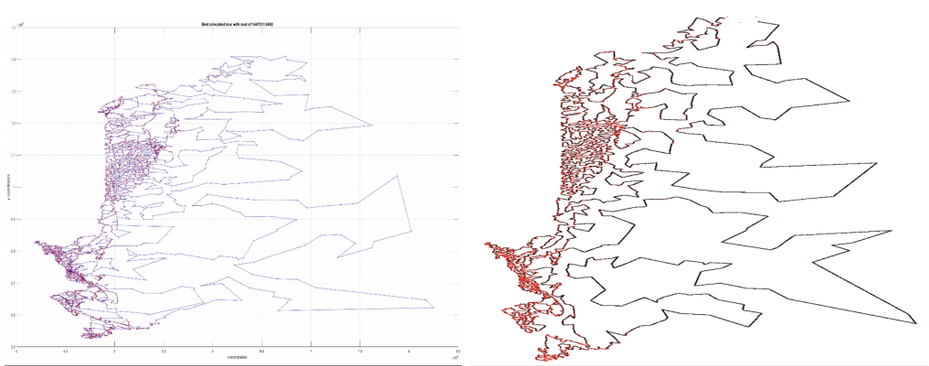
(a) Simulated Route (b) Optimal Route

Figure 30: Djibouti (38 cities) with an Optimal Length of 6656



(a) Simulated Route (b) Optimal Route

Figure 31: Oman (1979 cities) with an Optimal Length of 86891



(a) Simulated Route (b) Optimal Route

Figure 32: Canada (4663 cities) with an Optimal Length of 1290319

We notice that 2-opt algorithm does much better than 1 opt algorithm for all the cities that we did the experiment on.

Task 2:
In this section, we will develop and execute some of our own tasks in addition to the activities already outlined in the project handout. We will design several tasks based on the Uber dataset, and these tasks are built in response to Question 13, which asks us to draw the route that Santa Clause must travel within the Los Angeles area based on our estimated answer to the traveling salesman problem (TSP). We have defined three tasks, which are detailed below:

- Question 13 requires us to map Santa's trajectory, which we do in Python using the built-in 'plot' function. This is uninteresting since we can only observe the trajectory. As a result, the first assignment we presented is to transfer the trajectory we mapped onto a map of Los Angeles. By doing so, we can have a better understanding of Santa's itineraries and the sites that Santa must visit. The figure below depicts our final outcome.

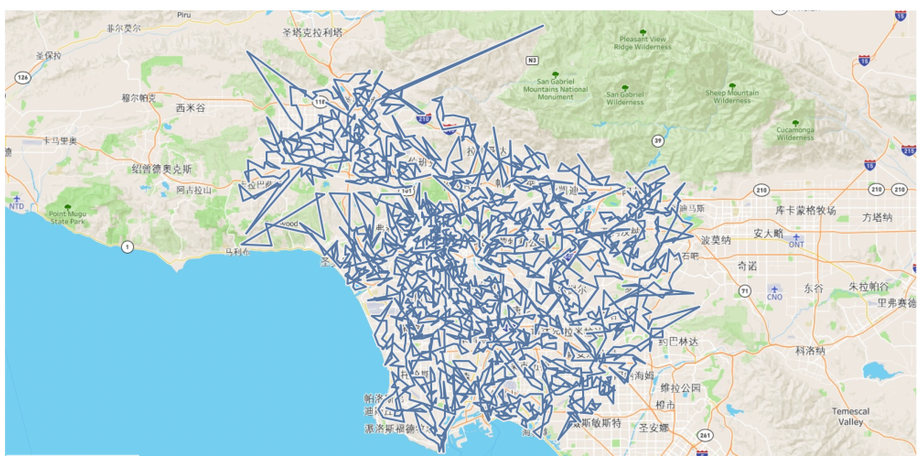


Figure 33: The optimal path which Santa should take for gift delivery on an Actual map of LA

- Given the route of Santa, the second objective we established is to calculate the total distance Santa must go in this TSP. We already have a set of coordinates that indicate Santa's route, so we can utilize it to determine the entire distance. We are specifically utilizing the Haversine formula to compute `https://en.wikipedia.org/wiki/Haversineformula`. And, according to our calculations, Santa must travel a total distance of 4364.616 kilometers just inside the Los Angeles area in this TSP. Thats a lot of distance that Santa has to cover!!

- Given the whole distance that Santa must travel, it is logical to wonder what his pace is. As a result, we designated our third duty as calculating Santa's trip speed. In terms of time, we define Santa's typical trip time as 10 p.m. to 4 a.m., giving us a total of 6 hours. As a consequence, the speed is 727.436 km/hr which is 452.0077746 miles/hr, which is nearly as fast as an airplane. Let's hope the reindeers are that fast or someone gifts Santa a jet this year :)!