ECE 232E Project 4

Graph Algorithms

Chenchen Kuai	206074833
Hao Wang	405629183
Yuning Yang	705930008

QUESTION 1: What are upper and lower bounds on ρij? Provide a justification for using log-normalized return (ri(t)) instead of regular return (qi(t)).

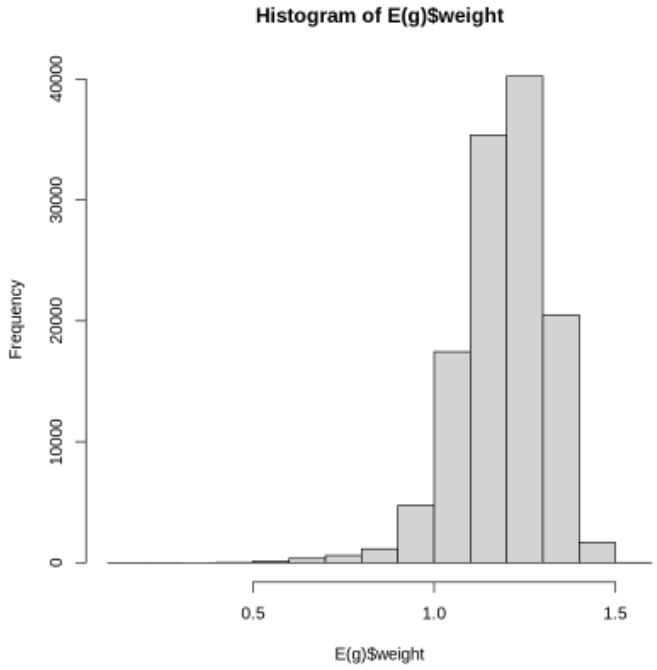
Answer:

For the bounds of the correlation coefficient, its values range from -1 to 1. A negative value indicates negative correlation, a positive value indicates positive correlation, and 0 indicates no correlation.

Regular return refers to the percentage change in stock price between two time points. However, regular return does not necessarily follow a normal distribution. In financial markets, price changes often exhibit a log-normal distribution, which means that taking the logarithm of the returns can make the data more normally distributed. By using log-normalized returns, we can better handle and model financial data. Additionally, outliers are taken care of in the log normalized return.

QUESTION 2: Plot a histogram showing the un-normalized distribution of edge weights.

Answer: We complete the function of calculate the correlation and use the helper code to plot the histogram of graph weight, the histogram is as following.



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:

We can see that there the nodes with same color is more likely to be gathered, such as the energy, mostly gathered in the upper right of the graph. This shows that the prize of these stocks have high correlation. The high correlation between these stocks results in the formation of tightly connected subgraphs within the MST. While, we can see some outliers, such as the upper left Consumer Discretionary stock in the cluster of the Information technology. These stocks exhibited lower correlations with other stocks of its sector. These outliers may represent specific market conditions or independent investment opportunities.

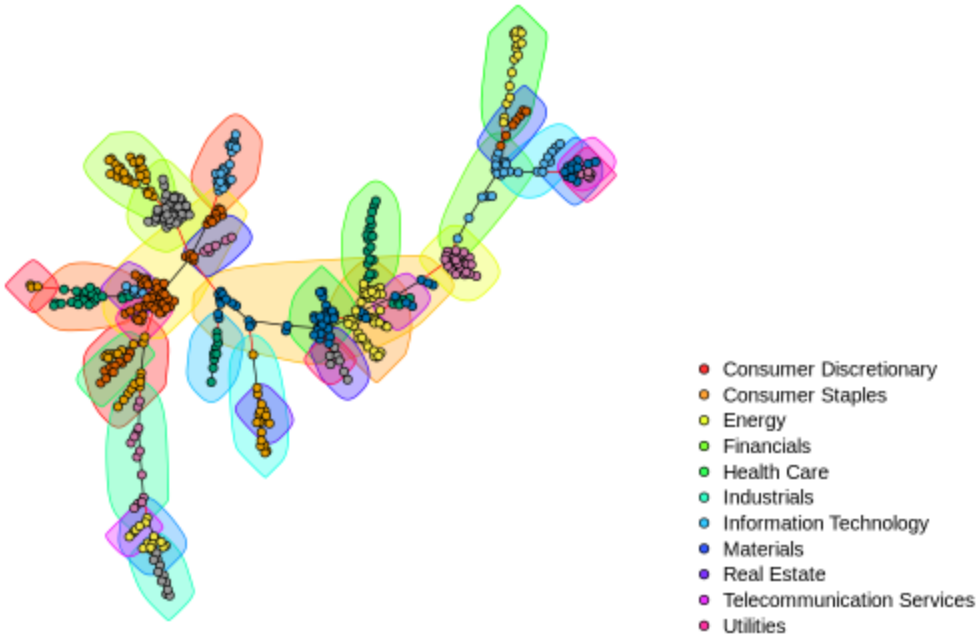
QUESTION 4: Run a community detection algorithm (for example walktrap) on the MST obtained above. Plot the communities formed. Compute the homogeneity and completeness of the clustering. (you can use the 'clevr' library in r to compute homogeneity and completeness).

Answer:

The homogeneity is 0.6826

The completeness is 0.4793

The detected communities is as follows: There are a total of 33 communities



QUESTION 5: Report the value of α for the above two cases and provide an interpretation for the difference.

Answer:

Alpha value for method 1 is: 0.8289

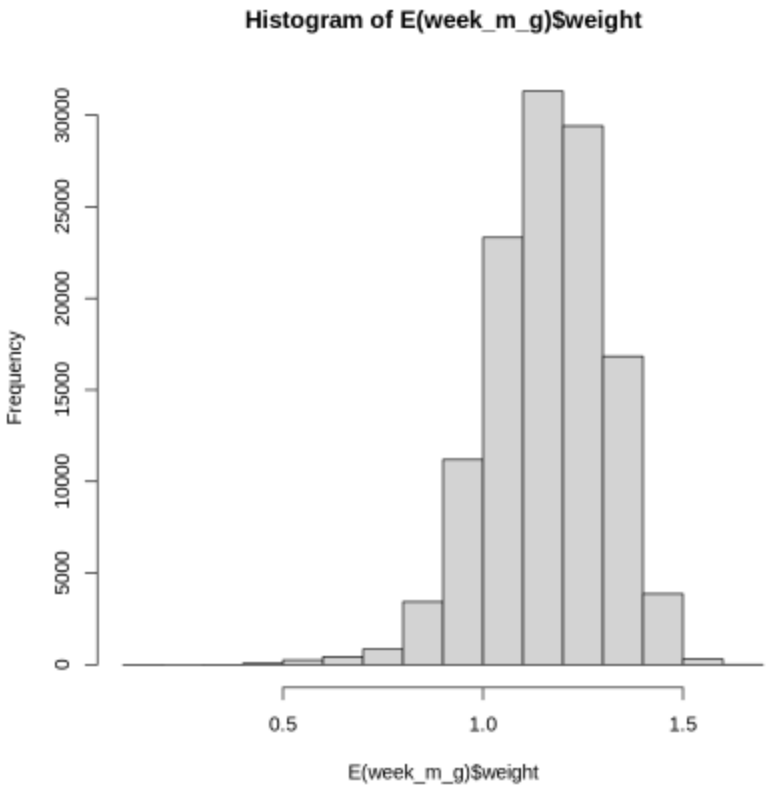
Alpha value for method 2 is: 0.1142

In Method 1, the alpha value is significantly higher compared to Method 2. This indicates that Method 1, which considers the local neighborhood information, performs significantly better in predicting the market sector of the unknown stock compared to Method 2, which relies on the overall sector distribution in the graph. The notable difference between the alpha values suggests that incorporating local neighborhood information (Method 1) is more effective in predicting the sector of the unknown stock than relying solely on the overall sector distribution (Method 2).

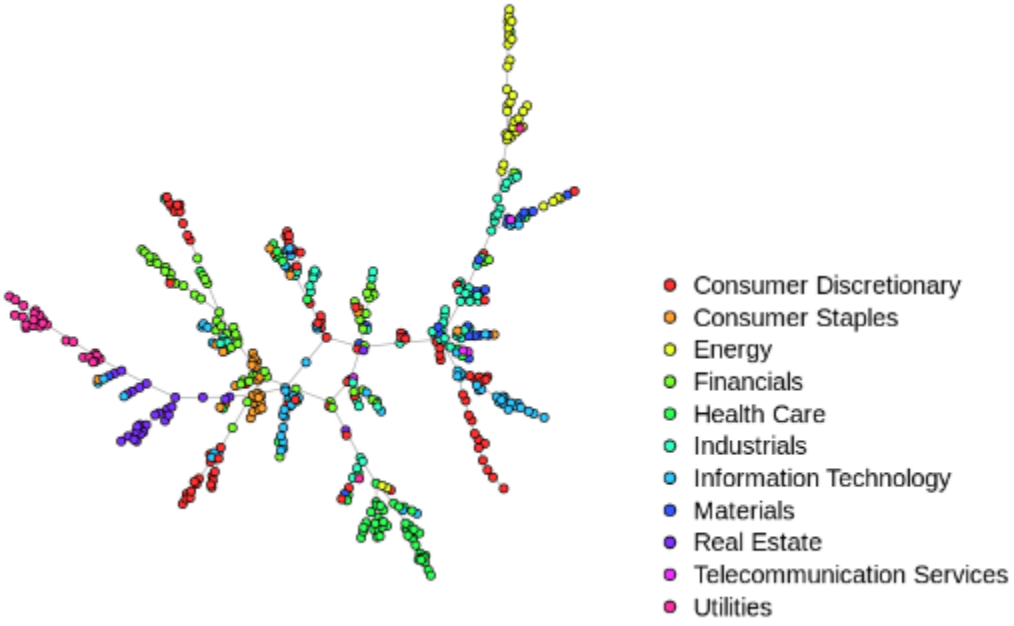
QUESTION 6: Repeat questions 2,3,4,5 on the WEEKLY data.

Answer:

Q6-Q2: histogram of edge weights



Q6-Q3: MST and explanation



We can still see that there the nodes with same color is more likely to be gathered, such as the energy, mostly gathered in the upper right of the graph. This shows that the prize of these stocks

have high correlation. The high correlation between these stocks results in the formation of tightly connected subgraphs within the MST. We can also see outliers, representing specific market conditions or independent investment opportunities.

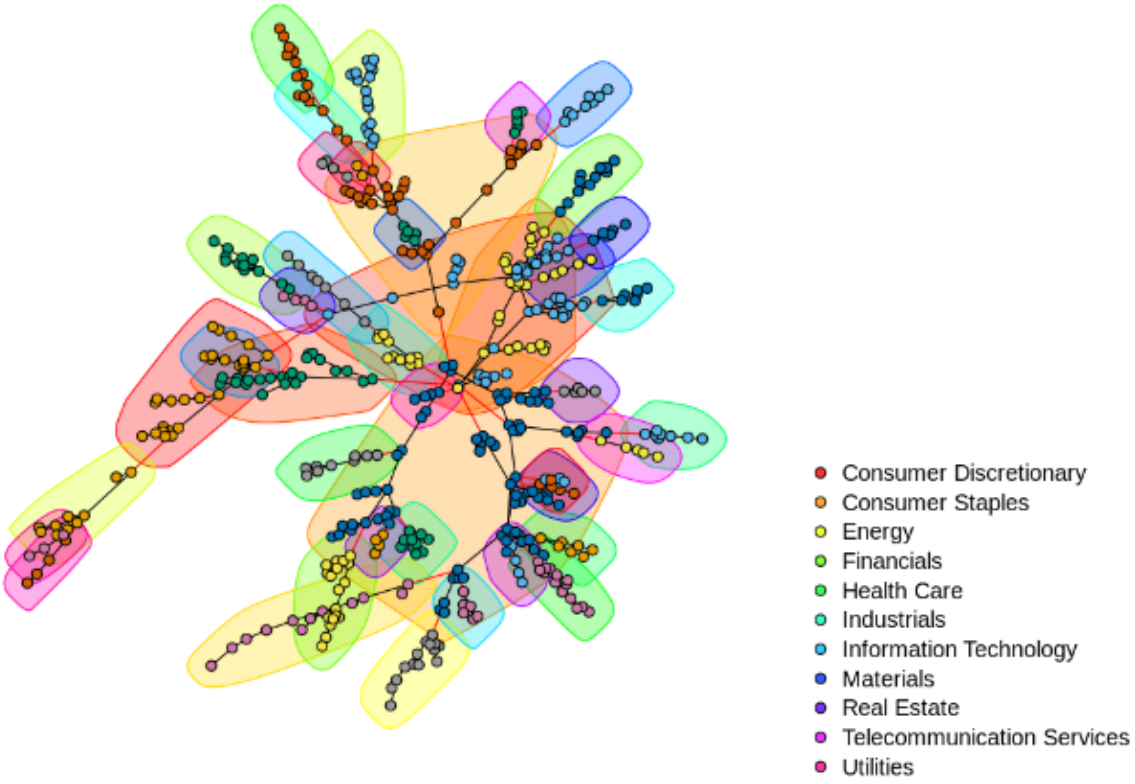
We can also observe some difference with the daily data, that is because the time granularity becomes coarser, the number of data points decreases, leading to fewer connections and weaker correlations.

Q6-Q4: community detection, homogeneity and completeness

The homogeneity is 0.5811

The completeness is 0.3900

There are a total of 42 clusters.



Q6-Q5: predict the market sector

Alpha value for method 1 is: 0.7440

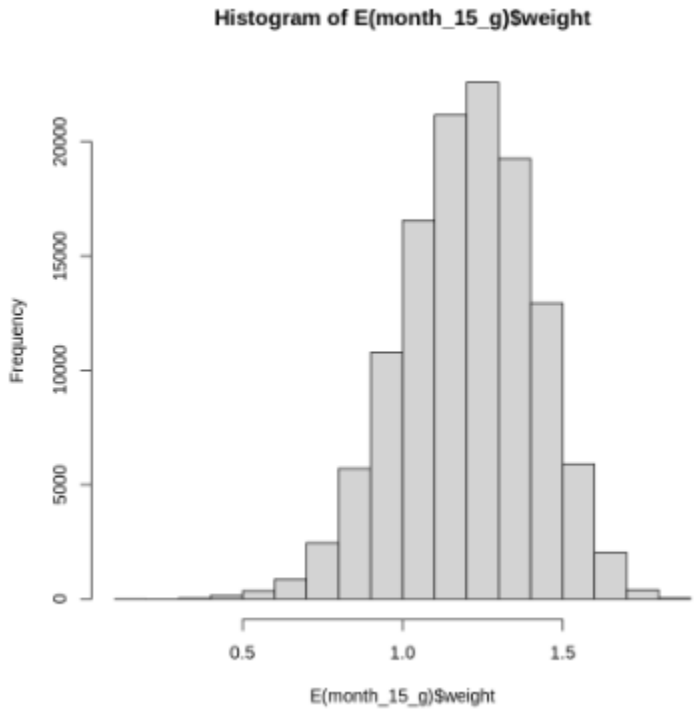
Alpha value for method 2 is: 0.1143

The alpha of method 1 is still higher than method 2, suggesting that incorporating local neighborhood information (Method 1) is more effective in predicting the sector of the unknown stock than relying solely on the overall sector distribution (Method 2).

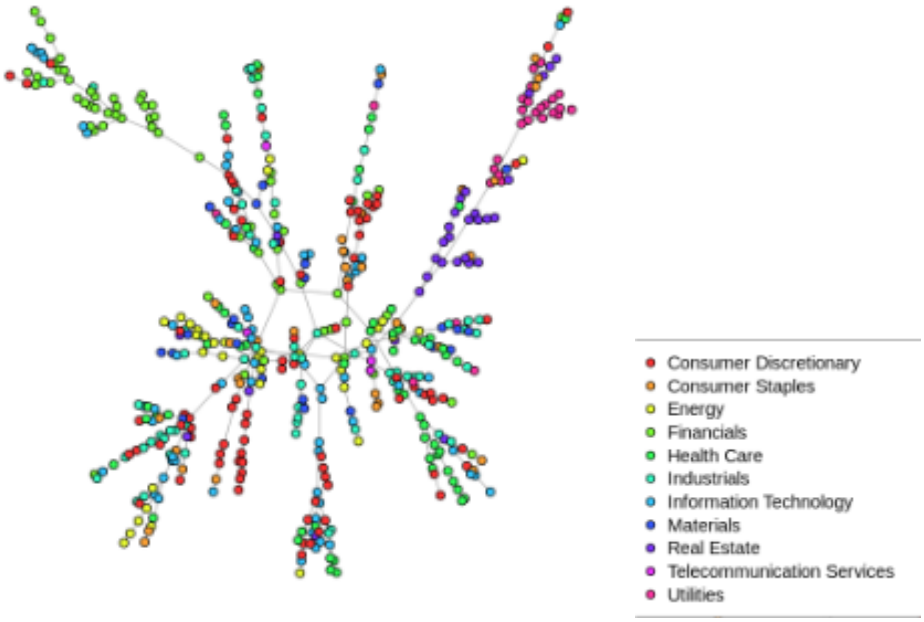
QUESTION 7: Repeat questions 2,3,4,5 on the MONTHLY data.

Answer:

Q7-Q2: histogram of edge weights



Q7-Q3: MST and explanation



We can still see that there the nodes with same color is more likely to be gathered. We can also see outliers, representing specific market conditions or independent investment opportunities.

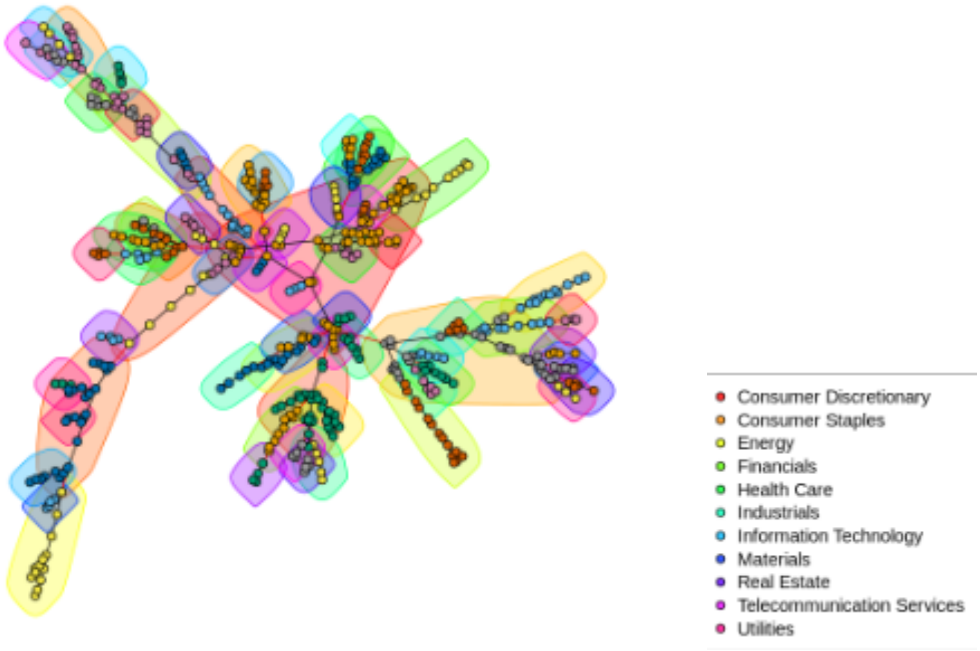
We can see that there are more outliers, that is because the time granularity becomes coarser, the number of data points decreases, leading to more noise, fewer connections, and weaker correlations.

Q7-Q4: community detection, homogeneity and completeness

The homogeneity is 0.4794

The completeness is 0.2775

There are a total of 65 clusters.



Q7-Q5: predict the market sector

Alpha value for method 1 is: 0.4844

Alpha value for method 2 is: 0.1143

The alpha of method 1 is still higher than method 2, suggesting that incorporating local neighborhood information (Method 1) is more effective in predicting the sector of the unknown stock than relying solely on the overall sector distribution (Method 2).

QUESTION 8: Compare and analyze all the results of daily data vs weekly data vs monthly data. What trends do you find? What changes? What remains similar? Give reason for your observations. Which granularity gives the best results when predicting the sector of an unknown stock and why?

Answer:

Trend of MST:

With a transition in time granularity from daily to weekly to monthly, the number of connections and their strength in the MST decrease. As the time granularity becomes coarser, the number of data points decreases, leading to fewer connections and weaker correlations. Additionally, with an increase in time granularity, the MST reveal longer-term trends and structures and result in more stable connections and relatively sparse Vine Clusters in the MST.

Trend of Homogeneity and Completeness:

Homogeneity refers to the purity of clusters formed by stocks within the same sector, while Completeness measures how well stocks of the same sector are assigned to the same cluster. The homogeneity and completeness values decrease as we move from daily to weekly to monthly data. This decrease suggests that finer-grained data (daily) provides more accurate and consistent clustering of stocks within their respective sectors.

Trend of Alpha Value:

Method 1, which considers local neighborhood information, consistently outperforms Method 2, which relies on the overall sector distribution, across all time granularities. The alpha values for Method 1 decreases as the time granularity coarsens. The alpha values for Method 2 remain consistent and very low across all time granularities, indicating that the overall sector distribution is not as effective in predicting the sector of an unknown stock.

Reason:

Finer-grained data (daily) leads to better results in predicting the sector of an unknown stock. The reasons can be attributed to the following factors:

1. Temporal Dynamics:

Financial markets are highly dynamic and finer-grained data captures the temporal dynamics more effectively, allowing for a better understanding of short-term correlations and sector-specific movements.

2. Localized Patterns:

The success of Method 1 in all time granularities suggests the presence of localized patterns within the immediate neighborhood of stocks. Finer-grained data enables the identification of finer-scale clusters and correlations, improving the accuracy of sector prediction based on the sectors of neighboring stocks.

The analysis indicates that daily data granularity provides the best results when predicting the sector of an unknown stock. The finer-grained data captures the temporal dynamics, localized patterns, and short-term correlations, leading to more accurate sector prediction.

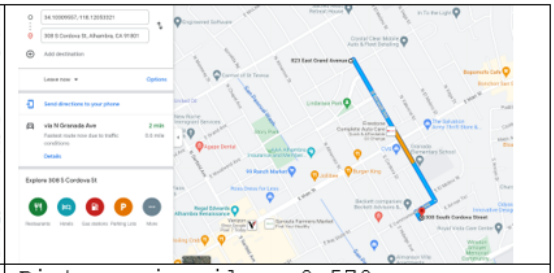
QUESTION 9: Report the number of nodes and edges in G.

Answer:

```
number of vertices: 2649
number of edges: 1003858
```

QUESTION 10: Build a minimum spanning tree (MST) of graph G. Report the street addresses near the two endpoints (the centroid locations) of a few edges. Are the results intuitive?

Answer:

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Distance in miles: 0.885
Time taken: 129.8</td><td>Distance in miles: 0.570
Time taken: 118.3</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Distance in miles: 0.447
Time taken: 90.2</td><td>Distance in miles: 0.621
Time taken: 126.5</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Distance in miles: 0.812
Time taken: 125.7</td><td>Distance in miles: 0.618
Time taken: 119.9</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Distance in miles: 0.936
Time taken: 125.2</td><td>Distance in miles: 0.412
Time taken: 91.8</td></tr>
</table>

The results are intuitive. First, the distance and time between two endpoints of an edge in the minimum spanning tree (MST) are relatively low, less than 1 mile and 2 minutes travel time. Also, for most of the source vertices, their target vertices are picked in their adjacent census tract. Such a choice could help establish the property of the MST: minimum possible total edge weight and explains how an MST picks the target vertices.

QUESTION 11: 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:

```
Triangle inequality holds by 93.7%
```

QUESTION 12: Find an upper bound on the empirical performance of the approximate algorithm

Answer:

Steps to use tree algorithm for TSP:

1. **Calculate TSP**: The Traveling Salesman Problem (TSP) is solved to find the optimal TSP path. This involves finding the shortest path that visits all cities and returns to the starting city, minimizing the total distance traveled.
2. **Duplicate the TSP**: The TSP path is duplicated to create a graph that can be traversed using a Eulerian walk. This duplication process ensures that all vertices have an even degree, allowing for the creation of an Eulerian circuit.
3. **Perform Euler walk on the duplicated graph**: An Eulerian walk is performed on the duplicated graph, starting from any vertex. This walk traces each edge exactly once, creating an embedded tour that visits each vertex once and returns to the starting vertex.
4. **Get the shortest distance of each tour OD**: For each pair of consecutive vertices in the embedded tour, the shortest distance (or weight) between those vertices is computed. This involves finding the shortest path distance between each pair of cities in the embedded tour.
5. **Sum up to get the optimal TSP distance**: Finally, the shortest distances for each pair of consecutive vertices are summed up to obtain the optimal TSP distance. This represents

the total cost of the TSP path, considering the shortest path distances between each pair of cities in the embedded tour.

Results:

```
MST Cost: 269085
TSP Cost: 422811
Ratio: 1.57
```

The relationship between the TSP cost, the MST cost, and the Approximate TSP cost are indicated as follows:

MST cost < TSP cost =< Approximate TSP cost<2 * MST cost

Thus, we have such inequality:

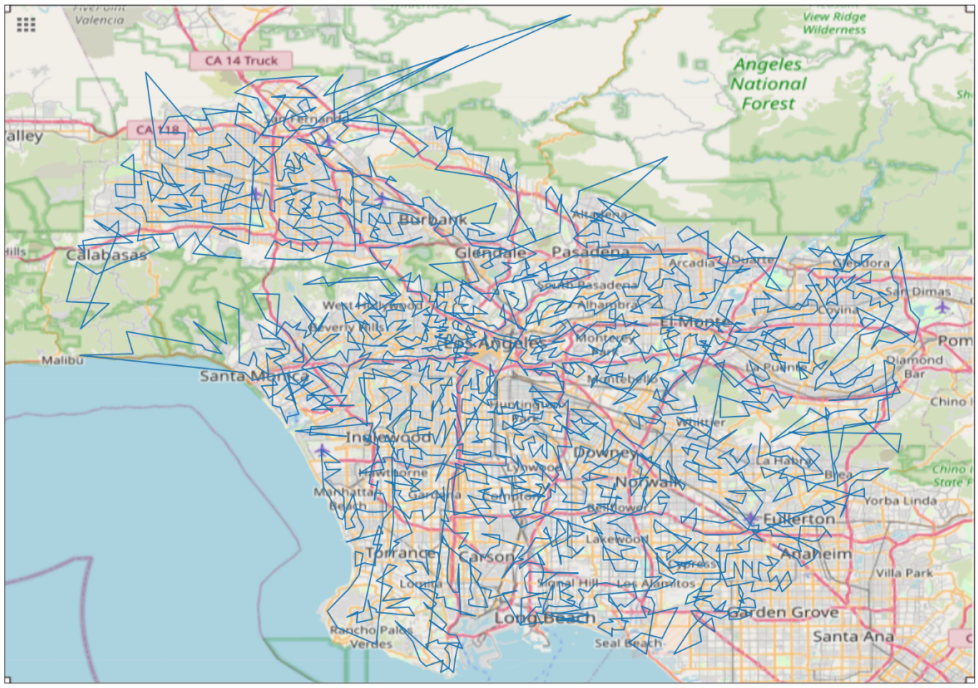
$$1 \leq \rho = \frac{Approximate\ TSP\ Cost}{Optimal\ TSP\ Cost} < \frac{Approximate\ TSP\ Cost}{MST\ Cost} = 1.57 < 2$$

Since 1<1.57<2, the ratio is in the reasonable range of expected value.

QUESTION 13: Plot the trajectory that Santa has to travel!

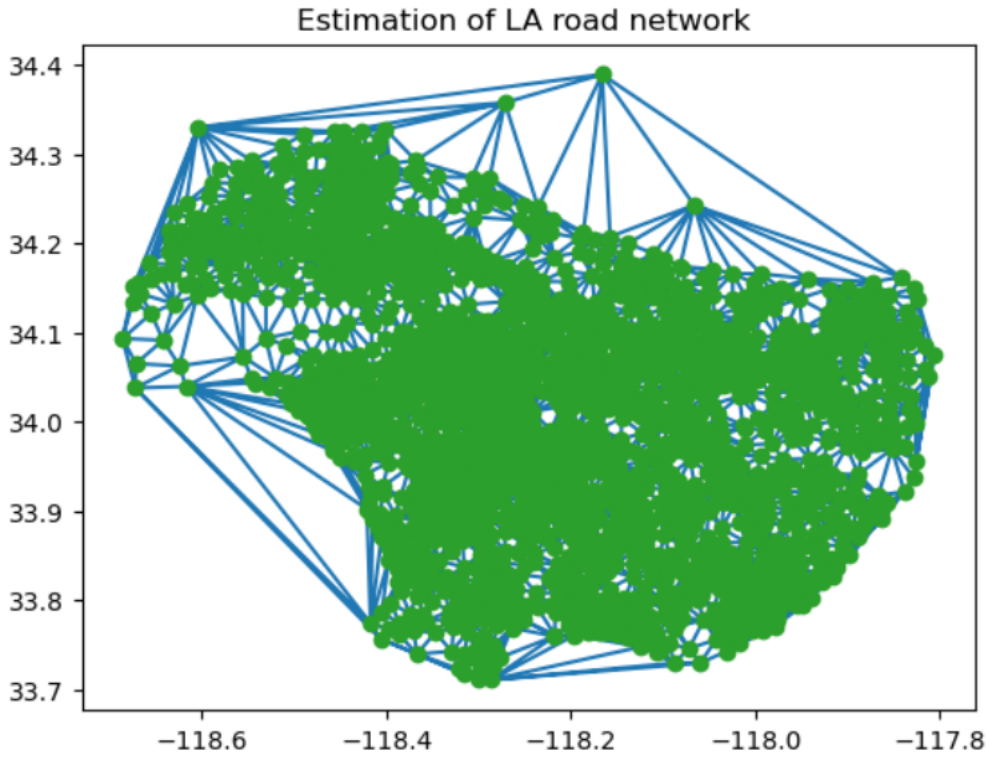
Answer:

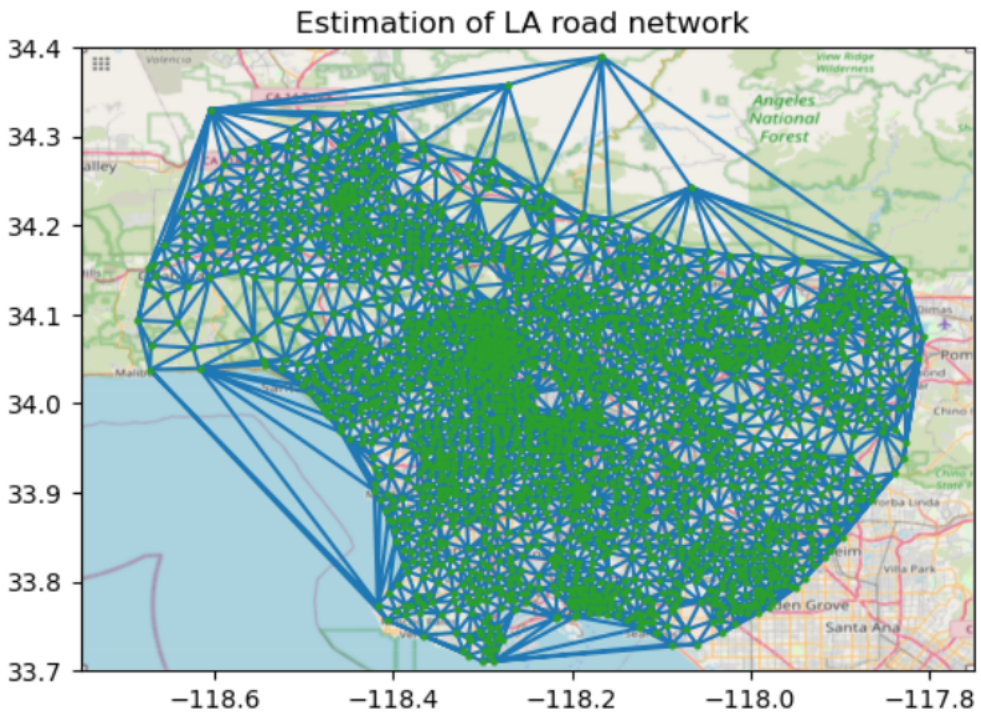
Here's the figure showing the tour of Santa!



QUESTION 14: Plot the road mesh that you obtain and explain the result. Create a graph GΔ whose nodes are different locations and its edges are produced by triangulation.

Answer:

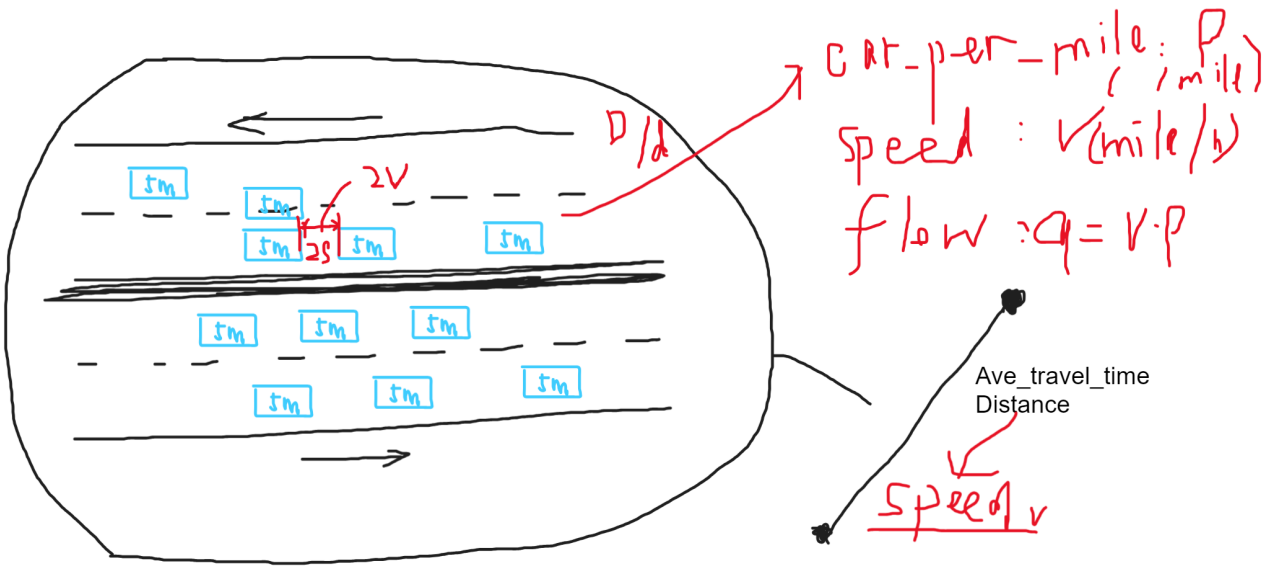




QUESTION 15: Using simple math, calculate the traffic flow for each road in terms of cars/hour. Report your derivation.

Answer:

Here shows the calculation steps of the traffic flow:



```
edge_ends = tri_g.get_edgelist()
edge_ends
array1 = tri_g.vs[sources]['coordinates']
array2 = tri_g.vs[targets]['coordinates']
result = [a1 - a2 for a1, a2 in zip(array1, array2)]
distance = [69 * np.sqrt(np.sum(np.square(d))) for d in result]
travel_time = tri_g.es['weight']
speeds =  np.array(distance) / np.array(travel_time) * 3600
car_length = 0.003
safety_distances = speeds * (2/3600)
n_lanes = 2
cars_per_mile = distance/(safety_distances+car_length)*n_lanes
cars_per_hour = cars_per_mile * np.array(speeds)
```

Thus,

Traffic flow = cars_per_mile (density) * speed

Here is the output of traffic flow:

```
Out[43]: array([2610.30838923, 2320.93506639, 1564.10824046, ..., 1723.36439509,
                2680.91204853, 1450.17172086])
```

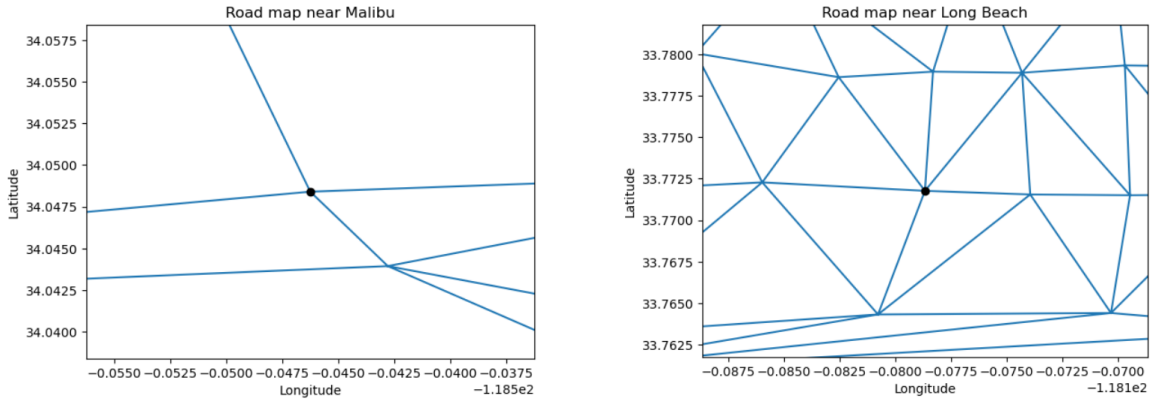
QUESTION 16: Calculate the maximum number of cars that can commute per hour from Malibu to Long Beach. Also calculate the number of edge-disjoint paths between the two spots. Does the number of edge-disjoint paths match what you see on your road map?

Answer:

Here are the results of this problem:

```
Number of independent roads: 4
Number of cars per hour: 7579.860651821042
```

The road network from source node (Malibu) to target node (Long Beach):

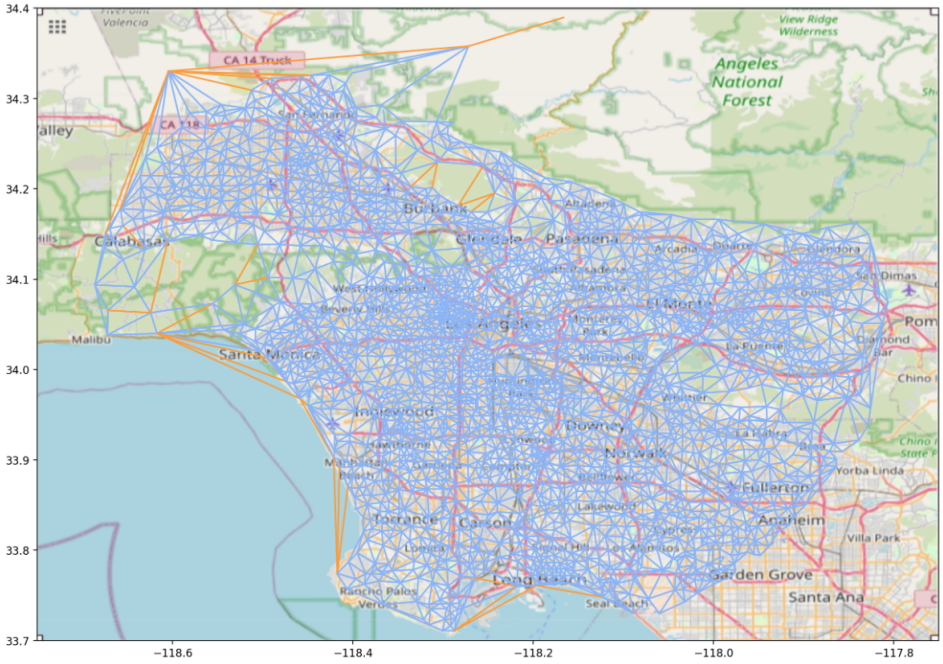


The degree of the Malibu node is 4, while the degree of the Long Beach node is 6. Thus, the max number of independent routes is 4. Since the road network is large, there is a high probability that the routes from Malibu can find a disjoint path to Long Beach, and the number of edge-disjoint paths could be 4 from what I can see in the road map, matching the result.

QUESTION 17: Plot G˜Δ on actual coordinates. Do you think the thresholding method worked?

Answer:

The threshold for the weight is set 800, The trimmed edges are shown in yellow.



QUESTION 18: Now, repeat question 13 for G˜Δ and report the results. Do you see any changes? Why?

Answer:

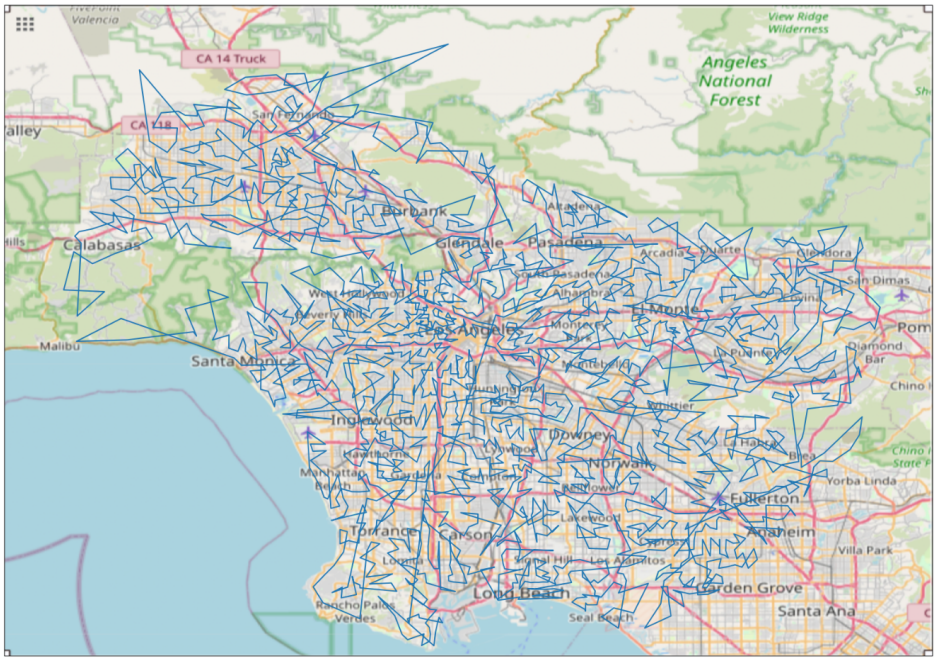


Figure. TSP route for trimmed graph

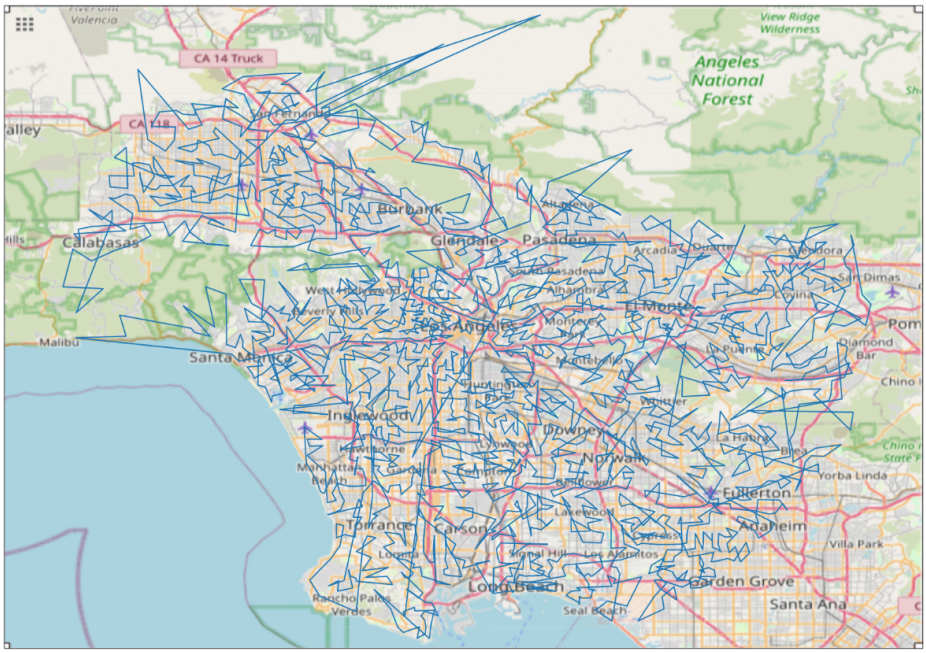


Figure. TSP for origin graph

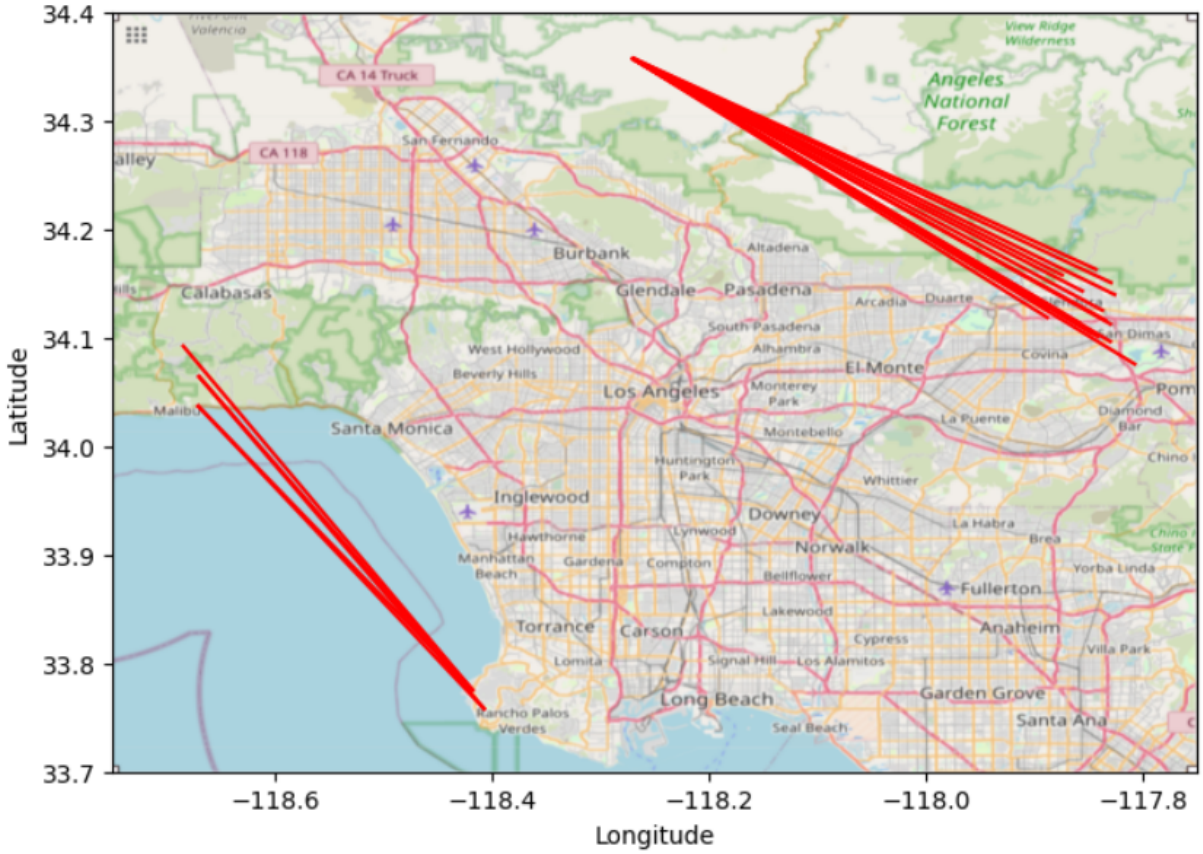
Most of the trip patterns are similar, while there are still changes. The routes now are all in reasonable places, excluding the trips in sea and in mountains. Also, there are fewer large jumps.

QUESTION 19: Strategy 1 (geo distance, static): Reduce the maximum extra traveling distance. The extra traveling distance between 2 locations is the difference of the shortest traveling distance and the straight line distance. i.e

extra distance(v,s) = distance of shortest path(v,s) - euclidean distance(v,s)

Use the coordinates of v and s to get the euclidean distance between them. Calculate the extra distance between all pairs of points. The top 20 pairs with highest extra distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report.Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?

Answer: The top 20 pairs with the greatest extra distance with the source and destination of these pairs are shown below, along with the graph of new edges, are displayed below. We use the graph that was already trimmed in Question 17, and the plot below is also making sense. Since it is static, The process to identify these pairs involves: calculating path weights using Euclidean distance, finding shortest paths for all pairs, computing extra distances for all pairs, sorting these distances, and selecting the top 20 pairs. The most complex step is finding shortest paths for all pairs, and its time complexity is O(|V|(|V|+|E|)log|V|), and |E| = k|V|(k is constant). Therefore, the overall time complexity simplifies to O(|V|^2 log |V|).



```
Top 20 pairs with highest extra distance: (distance, v, s)
[(0.11661860410860841, 1783, 2416), (0.11617165060871026, 2140, 2419), (0.11414825603644496, 1860, 2416), (0.11337999394573656,
382, 2419), (0.11038819549635265, 383, 2419), (0.10889014837969668, 1699, 1783), (0.10872873756879575, 391, 2419), (0.108115112
86294317, 1901, 2419), (0.10795094696637342, 381, 2419), (0.1066007559055549, 1699, 1860), (0.10657572718707192, 2163, 2419),
(0.10605333016714441, 386, 2419), (0.10600807595529399, 390, 2419), (0.10550259008670515, 1904, 2419), (0.10534695919001047, 4
5, 2419), (0.10533990016218936, 1783, 2413), (0.10533894891149931, 1906, 2419), (0.1053347425776226, 2158, 2419), (0.1052863306
9215532, 1902, 2419), (0.10512508811330096, 392, 2419)]

edge 1: (-118.41714115454548, 33.774103109090916) - (-118.67145511923503, 34.06533878290219)

edge 2: (-117.84197423684212, 34.16224818421053) - (-118.27158981797648, 34.35782482248528)

edge 3: (-118.40648371022725, 33.75689578693178) - (-118.67145511923503, 34.06533878290219)

edge 4: (-117.82810402090584, 34.1499674041812) - (-118.27158981797648, 34.35782482248528)

edge 5: (-117.8732378168498, 34.15698468864466) - (-118.27158981797648, 34.35782482248528)

edge 6: (-118.67225393333337, 34.03809095384618) - (-118.41714115454548, 33.774103109090916)

edge 7: (-117.83331135955059, 34.11532583146067) - (-118.27158981797648, 34.35782482248528)

edge 8: (-117.82866074137928, 34.111707206896554) - (-118.27158981797648, 34.35782482248528)

edge 9: (-117.82467288479266, 34.13885138248847) - (-118.27158981797648, 34.35782482248528)

edge 10: (-118.67225393333337, 34.03809095384618) - (-118.40648371022725, 33.75689578693178)

edge 11: (-117.88665511320752, 34.12443260377358) - (-118.27158981797648, 34.35782482248528)

edge 12: (-117.85454409302326, 34.142120790697675) - (-118.27158981797648, 34.35782482248528)

edge 13: (-117.83602374666668, 34.124341866666676) - (-118.27158981797648, 34.35782482248528)

edge 14: (-117.82879819999992, 34.0955664090909) - (-118.27158981797648, 34.35782482248528)

edge 15: (-117.88700062790699, 34.11772844186047) - (-118.27158981797648, 34.35782482248528)

edge 16: (-118.41714115454548, 33.774103109090916) - (-118.68599468727271, 34.09347549818183)

edge 17: (-117.84047893749997, 34.100165062500004) - (-118.27158981797648, 34.35782482248528)

edge 18: (-117.84598029213484, 34.103187044943816) - (-118.27158981797648, 34.35782482248528)

edge 19: (-117.80629424922105, 34.074753919003115) - (-118.27158981797648, 34.35782482248528)

edge 20: (-117.85748312987019, 34.112945454545475) - (-118.27158981797648, 34.35782482248528)
```

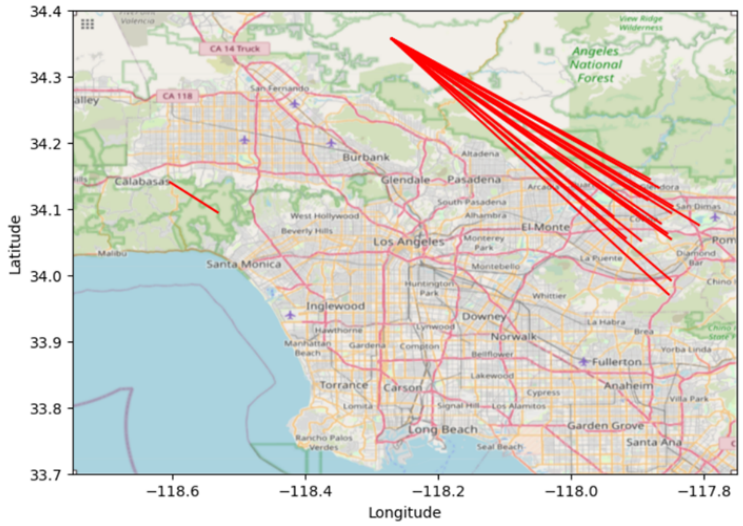
QUESTION 20: Strategy 2 (geo distance, static, with frequency): In strategy1, we are using the geographical distance between points to decide which new road to create. However, in the real world, some pairs are more in demand than other pairs. Assume that you know the frequency of travel between every pair. Use the frequency as the weight to multiply the difference. i.e

*weighted extra distance(v,s) = extra distance(v,s) * frequency(v,s)*

where frequency(v,s) is a random integer between [1,1000]. Use the coordinates of v and s to get the euclidean distance between them. Calculate the weighted extra distance between all pairs of points. The top 20 pairs with highest weighted extra distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?

Answer: For Strategy 2,The top 20 pairs with the highest weighted(frequency) extra distance with the source and destination of these pairs are shown below, along with the graph of new edges, are displayed below. For time complexity, it is the same as strategy 1 which is O(|V|^2 log

|V|) because the only change of strategy is to add frequency which does not impact time complexity.




```
Top 20 pairs with highest weighted extra distance: (distance, v, s)
[(103.98124128532093, 2161, 2419), (101.53301909129863, 43, 2419), (98.02366296487271, 2160, 2419), (96.80105885791512, 221, 24
19), (96.46255758380741, 2060, 2419), (96.24526105974026, 2142, 2419), (95.91584726055349, 1902, 2419), (95.69542269740609, 208
1, 2419), (95.62898984857308, 219, 2419), (95.50046121773036, 2052, 2419), (95.16990597076531, 2176, 2419), (94.43330172386798,
385, 2419), (94.18532618546296, 2151, 2419), (93.26020412511413, 234, 2419), (92.98644668882119, 389, 2419), (92.9052429534631
3, 2158, 2419), (91.65441779534508, 259, 2419), (90.86597665044513, 2069, 2419), (90.15682397510147, 253, 2419), (89.5546022569
9632, 1678, 2411)]

edge 1: (-117.89487851190476, 34.117642214285716) - (-118.27158981797648, 34.35782482248528)

edge 2: (-117.90064863057323, 34.15035921019108) - (-118.27158981797648, 34.35782482248528)

edge 3: (-117.88409822784814, 34.11270067088606) - (-118.27158981797648, 34.35782482248528)

edge 4: (-117.84833258558552, 34.09296071171172) - (-118.27158981797648, 34.35782482248528)

edge 5: (-117.93522791666668, 34.08785375) - (-118.27158981797648, 34.35782482248528)

edge 6: (-117.8802836388889, 34.14393644444445) - (-118.27158981797648, 34.35782482248528)

edge 7: (-117.80629424922105, 34.074753919003115) - (-118.27158981797648, 34.35782482248528)

edge 8: (-117.89365872839508, 34.05103139506174) - (-118.27158981797648, 34.35782482248528)

edge 9: (-117.86369810290833, 34.077044838926184) - (-118.27158981797648, 34.35782482248528)

edge 10: (-117.84852660194171, 34.05349977669904) - (-118.27158981797648, 34.35782482248528)

edge 11: (-117.89841259999999, 34.093485199999996) - (-118.27158981797648, 34.35782482248528)

edge 12: (-117.8771218404256, 34.137339489361686) - (-118.27158981797648, 34.35782482248528)

edge 13: (-117.85410418505344, 34.06184217081851) - (-118.27158981797648, 34.35782482248528)

edge 14: (-117.89010998113207, 34.082624283018866) - (-118.27158981797648, 34.35782482248528)

edge 15: (-117.86688824193546, 34.13122395161291) - (-118.27158981797648, 34.35782482248528)

edge 16: (-117.84598029213484, 34.103187044943816) - (-118.27158981797648, 34.35782482248528)

edge 17: (-117.85197046153847, 33.96954255384617) - (-118.27158981797648, 34.35782482248528)

edge 18: (-117.91658202173915, 34.05489234782609) - (-118.27158981797648, 34.35782482248528)

edge 19: (-117.84879976404494, 33.99119615730338) - (-118.27158981797648, 34.35782482248528)

edge 20: (-118.53067983104125, 34.09368131827117) - (-118.60545802183404, 34.14123286462885)
```

QUESTION 21: Strategy 3 (geo distance, dynamic): In the above strategy, we are creating all roads at the same time. This time, we will create the roads one by one.

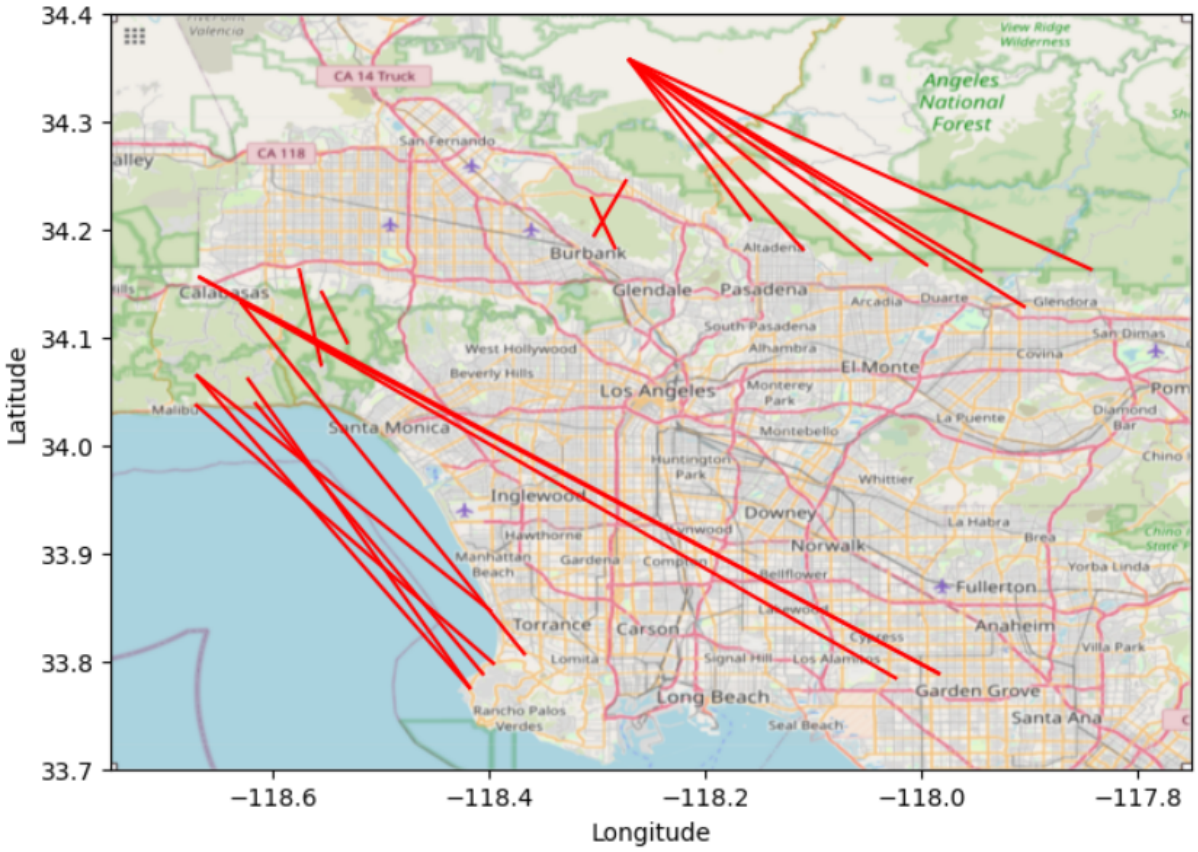
Repeat 20 times: i) Compute extra distance between all the pairs in the graph.

ii) Create a road between the pair with highest extra distance and update the graph.

iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?

Answer: For Strategy 3,The new top 20 pairs with the highest extra distance with the source and destination of these pairs are shown below, along with the graph of new edges, are displayed below. For time complexity, it is the similar as strategy 1 which is O(|V|^2 log| V|) because the only change of strategy 3 is to repeat the strategy 1 20 times.You can also treat it as O(K(|V|^2log|V|)) but in Big-O notation, K can be ignored.



Top 20 pairs with highest extra distance (dynamic): (distance, v, s)
[(0.11661860410860841, 1783, 2416), (0.11617165060871026, 2140, 2419), (0.10463929764174847, 986, 1510), (0.10350492324700983, 49, 2419), (0.1014667965126414, 285, 2419), (0.09603143370735362, 1717, 2419), (0.08975133249591655, 1783, 2417), (0.08808444169976332, 1956, 2419), (0.08073637034270212, 2247, 2419), (0.07875472008634088, 1005, 1678), (0.0775242820048263, 2414, 2619), (0.0740019810227257, 121, 689), (0.07361292518078288, 2242, 2419), (0.07341356533715154, 1699, 1781), (0.07252775265557754, 144, 2619), (0.07039598202487579, 2402, 2416), (0.06839124523119561, 144, 2040), (0.06673515270823566, 1700, 1782), (0.06612849598931886, 356, 1679), (0.06489104989670524, 2414, 2559)]

edge 1: (-118.41714115454548, 33.774103109090916) - (-118.67145511923503, 34.06533878290219)

edge 2: (-117.84197423684212, 34.16224818421053) - (-118.27158981797648, 34.35782482248528)

edge 3: (-118.57609893103452, 34.163914112068966) - (-118.55510068503939, 34.07289098425196)

edge 4: (-117.90384881034481, 34.12744793103448) - (-118.27158981797648, 34.35782482248528)

edge 5: (-117.94347146212125, 34.160677848484866) - (-118.27158981797648, 34.35782482248528)

edge 6: (-117.99399975280905, 34.16600654494379) - (-118.27158981797648, 34.35782482248528)

edge 7: (-118.41714115454548, 33.774103109090916) - (-118.62418590723568, 34.06269522170691)

edge 8: (-118.046042942029, 34.17109230434782) - (-118.27158981797648, 34.35782482248528)

edge 9: (-118.10868277868856, 34.18016231147541) - (-118.27158981797648, 34.35782482248528)

edge 10: (-118.5556897357724, 34.143095296747944) - (-118.53067983104125, 34.09368131827117)

edge 11: (-118.66929280689664, 34.156163006896556) - (-117.98288185714287, 33.787883428571426)

edge 12: (-118.3036932608696, 34.1928858115942) - (-118.2728725826087, 34.24626368695651)

edge 13: (-118.15749682068964, 34.20758387586207) - (-118.27158981797648, 34.35782482248528)

edge 14: (-118.67225393333337, 34.03809095384618) - (-118.39491433093522, 33.796855824940074)

edge 15: (-118.63048660074622, 34.1327155932836) - (-117.98288185714287, 33.787883428571426)

edge 16: (-118.39748945762716, 33.84561491864407) - (-118.67145511923503, 34.06533878290219)

edge 17: (-118.63048660074622, 34.1327155932836) - (-118.36653025668451, 33.80541077540108)

edge 18: (-118.61724170318024, 34.03975774911659) - (-118.40485093013112, 33.78694638427947)

edge 19: (-118.28301819999999, 34.18119773333334) - (-118.3059771739131, 34.22977666086955)

edge 20: (-118.66929280689664, 34.156163006896556) - (-118.02266135185184, 33.78354657407407)

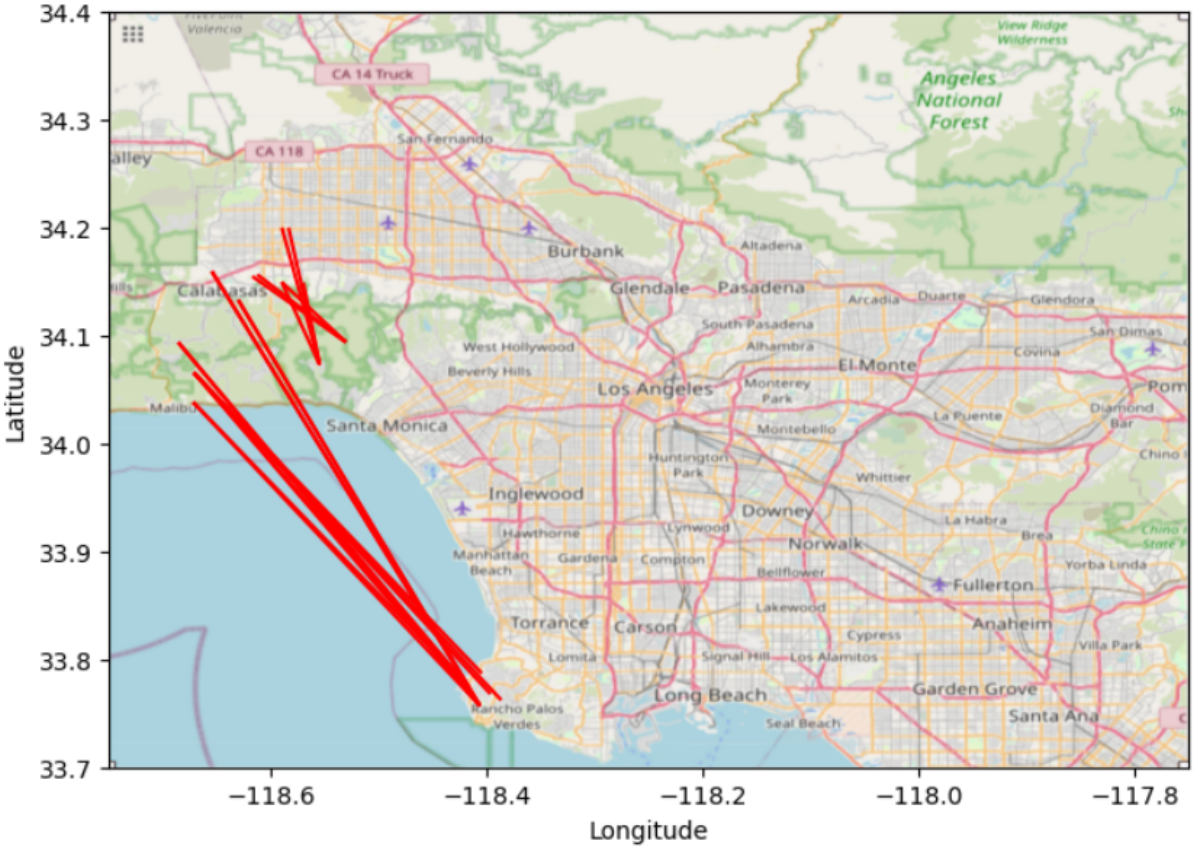
QUESTION 22: Strategy 4 (Travel time, static): We want to optimize to reduce the maximum extra travelling time. The extra travelling distance between 2 locations is the difference of the shortest traveling time and the straight line travel time. i.e

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)

Use the coordinates of v and s to get the euclidean distance between them. Calculate the extra time between all pairs of points. The top 20 pairs with highest extra time will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy? What is the time complexity of the strategy?

Answer: For Strategy 4,The top 20 pairs with the highest extra time with the source and destination of these pairs are shown below, along with the graph of new edges, are displayed below. For time complexity, from our perspective, it is the same as strategy 1 which is O(|V|^2 log |V|)in Big - O. But it might cost two times to compute the shortest distance. For distance of shortest path(v,s) and travel time of shortest path(v,s), when considering them individually, both calculations are of complexity O(|V|^2log|V|). However, in this case, since we need to run Dijkstra's algorithm twice to obtain the shortest distance path and the shortest time path separately, the actual running time will be longer.



Top 20 pairs with highest extra time: (time, v, s)
[(1270.3895410723735, 1783, 2416), (1252.8631429260095, 1860, 2416), (1153.8889769479183, 1783, 2413), (1146.2778771962248, 1859, 2416), (1134.1965992778387, 1860, 2413), (1132.3580394431988, 1699, 1783), (1119.6690284556112, 1699, 1860), (1107.7052218590086, 985, 1678), (1104.5835299882676, 988, 1510), (1094.0169192124886, 1782, 2416), (1075.812196539635, 989, 1510), (1071.6095362416345, 144, 1783), (1053.8289562498721, 144, 1860), (1051.7197552248826, 430, 1783), (1044.4861781848101, 950, 1510), (1043.687810050943, 989, 1678), (1041.616826813589, 1882, 2416), (1034.078921859059, 430, 1860), (1032.7277657916725, 951, 1510), (1030.2237008168743, 984, 1678)]

edge 1: (-118.41714115454548, 33.774103109090916) - (-118.67145511923503, 34.06533878290219)

edge 2: (-118.40648371022725, 33.75689578693178) - (-118.67145511923503, 34.06533878290219)

edge 3: (-118.41714115454548, 33.774103109090916) - (-118.68599468727271, 34.09347549818183)

edge 4: (-118.39683944230771, 33.7679271025641) - (-118.67145511923503, 34.06533878290219)

edge 5: (-118.40648371022725, 33.75689578693178) - (-118.68599468727271, 34.09347549818183)

edge 6: (-118.67225393333337, 34.03809095384618) - (-118.41714115454548, 33.774103109090916)

edge 7: (-118.67225393333337, 34.03809095384618) - (-118.40648371022725, 33.75689578693178)

edge 8: (-118.6174550722892, 34.15459418072292) - (-118.53067983104125, 34.09368131827117)

edge 9: (-118.57017334070797, 34.14872431415929) - (-118.55510068503939, 34.07289098425196)

edge 10: (-118.40485093013112, 33.78694638427947) - (-118.67145511923503, 34.06533878290219)

edge 11: (-118.59018528030296, 34.14922015909091) - (-118.55510068503939, 34.07289098425196)

edge 12: (-118.63048660074622, 34.1327155932836) - (-118.41714115454548, 33.774103109090916)

edge 13: (-118.63048660074622, 34.1327155932836) - (-118.40648371022725, 33.75689578693178)

edge 14: (-118.65489981553401, 34.15918811650484) - (-118.41714115454548, 33.774103109090916)

edge 15: (-118.59007249999999, 34.199754199999994) - (-118.55510068503939, 34.07289098425196)

edge 16: (-118.59018528030296, 34.14922015909091) - (-118.53067983104125, 34.09368131827117)

edge 17: (-118.3868687012987, 33.76222715584416) - (-118.67145511923503, 34.06533878290219)

edge 18: (-118.65489981553401, 34.15918811650484) - (-118.40648371022725, 33.75689578693178)

edge 19: (-118.58392439285714, 34.19973064285714) - (-118.55510068503939, 34.07289098425196)

edge 20: (-118.61281342424239, 34.15615184848486) - (-118.53067983104125, 34.09368131827117)

QUESTION 23: Strategy 5 (Travel time, dynamic): Similar to strategy 3, this time we will create roads one by one while optimizing the travel time.

Repeat 20 times:

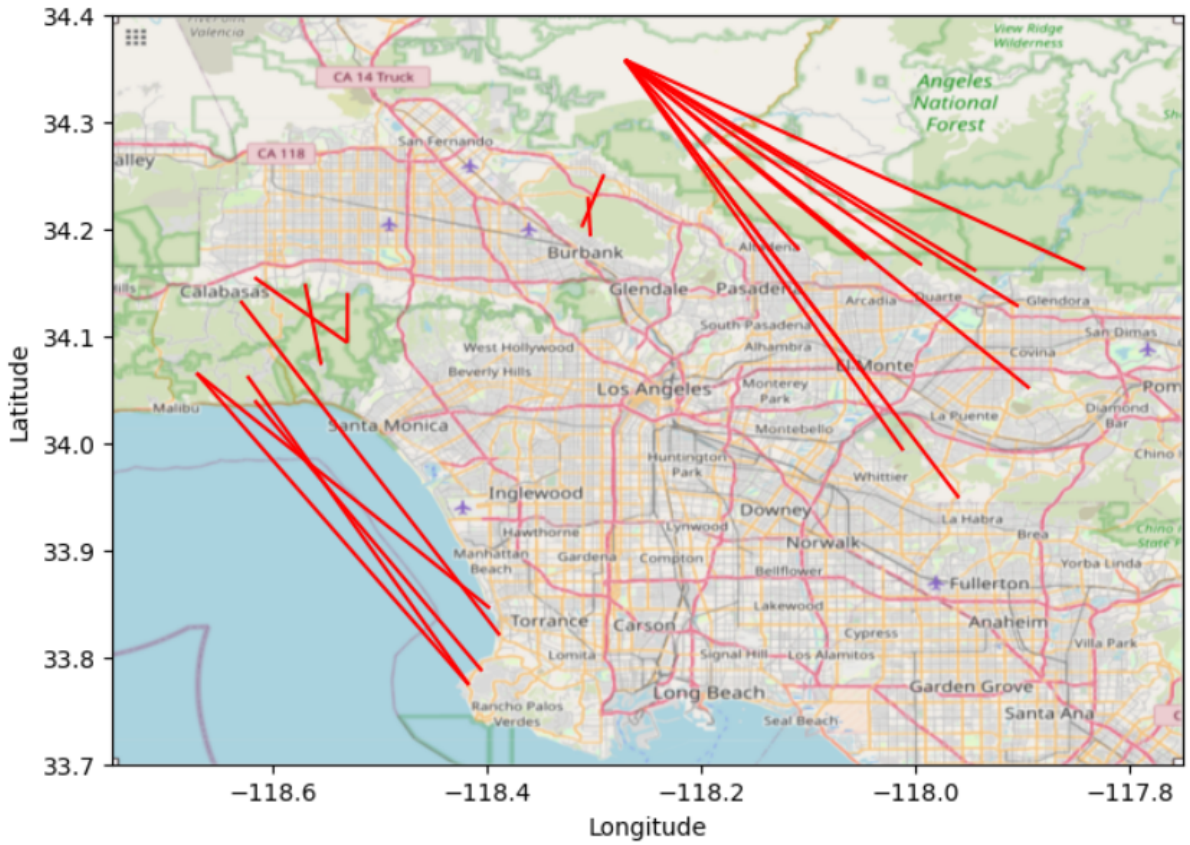
i) Compute extra time between all the pairs in the graph.

ii) Create a road between the pair with the highest extra time and update the graph.

iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?

Answer: For Strategy 5,The new top 20 pairs with the highest extra time with the source and destination of these pairs are shown below, along with the graph of new edges, are displayed below. For time complexity, it is the same as strategy 4 which is O(|V|^2 log |V|) because the only change of strategy 5 is to repeat the strategy 4 20 times.You can also treat it as O(K(|V|^2log|V|)) but in Big-O notation, K can be ignored.



Top 20 pairs with highest extra time (dynamic): (time, v, s)
[(1270.3895410723735, 1783, 2416), (1107.7052218590086, 985, 1678), (1104.5835299882676, 988, 1510), (978.0953184724199, 2140, 2419), (968.0607081219923, 1783, 2417), (906.8975174051993, 2081, 2419), (897.5132840438332, 49, 2419), (854.5711392642315, 285, 2419), (834.3139270021093, 2262, 2419), (793.4149992916323, 2273, 2419), (790.3522519838634, 1717, 2419), (772.4319041228841, 2402, 2416), (739.9104313030123, 1003, 1678), (732.3491204243983, 1964, 2419), (724.8957281211938, 1956, 2419), (707.650688529332, 1700, 1782), (699.1161963754884, 120, 687), (678.5899314331853, 2247, 2419), (668.9713721662154, 121, 1679), (656.8276261852361, 144, 1875)]

edge 1: (-118.41714115454548, 33.774103109090916) - (-118.67145511923503, 34.06533878290219)

edge 2: (-118.6174550722892, 34.15459418072292) - (-118.53067983104125, 34.09368131827117)

edge 3: (-118.57017334070797, 34.14872431415929) - (-118.55510068503939, 34.07289098425196)

edge 4: (-117.84197423684212, 34.16224818421053) - (-118.27158981797648, 34.35782482248528)

edge 5: (-118.41714115454548, 33.774103109090916) - (-118.62418590723568, 34.06269522170691)

edge 6: (-117.89365872839508, 34.05103139506174) - (-118.27158981797648, 34.35782482248528)

edge 7: (-117.90384881034481, 34.12744793103448) - (-118.27158981797648, 34.35782482248528)

edge 8: (-117.94347146212125, 34.160677848484866) - (-118.27158981797648, 34.35782482248528)

edge 9: (-117.95946314285713, 33.94862083333332) - (-118.27158981797648, 34.35782482248528)

edge 10: (-118.01154852195125, 33.99293787317071) - (-118.27158981797648, 34.35782482248528)

edge 11: (-117.99399975280905, 34.16600654494379) - (-118.27158981797648, 34.35782482248528)

edge 12: (-118.39748945762716, 33.84561491864407) - (-118.67145511923503, 34.06533878290219)

edge 13: (-118.53047408659789, 34.14032269896909) - (-118.53067983104125, 34.09368131827117)

edge 14: (-118.04268341379307, 34.0580242413793) - (-118.27158981797648, 34.35782482248528)

edge 15: (-118.046042942029, 34.17109230434782) - (-118.27158981797648, 34.35782482248528)

edge 16: (-118.61724170318024, 34.03975774911659) - (-118.40485093013112, 33.78694638427947)

edge 17: (-118.31185060431655, 34.20150541726619) - (-118.29105503125, 34.250763375)

edge 18: (-118.10868277868856, 34.18016231147541) - (-118.27158981797648, 34.35782482248528)

edge 19: (-118.3036932608696, 34.1928858115942) - (-118.3059771739131, 34.22977666086955)

edge 20: (-118.63048660074622, 34.1327155932836) - (-118.38821546987946, 33.82029325301204)

QUESTION 24: Strategy comparison. Compare the following strategies:

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

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

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

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

i) if yes, which one and why?

ii) if not, what would be a better strategy to construct roads and is it optimal? What is the time complexity? (assume that you want to reduce travelling distances and you know before hand that we have to construct 20 new roads).

e) 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.

Answer:

a) 1 vs 2 - Compare and analyze the results. Which is better? Why?

Strategy 1 focuses on reducing the maximum extra traveling distance by considering the geographical distance between points. Strategy 2 introduces the frequency of travel between pairs as a weight to multiply the extra distance.Upon comparison, Strategy 2 is considered better. This is because Strategy 2 takes into account the frequency of travel, which reflects the demand between pairs. By incorporating this information, the suggested new edges are more likely to serve the routes that are in higher demand, potentially improving overall transportation efficiency.

b) 1 vs 3 - Compare and analyze the results. Which is better? Why?

Strategy 3 involves creating roads one by one while optimizing travel time. On the other hand, Strategy 1 creates all roads at the same time, considering only the geographical distance between points.Upon comparison, Strategy 3 is considered better. When using the static method (Strategy 1) to determine new edges, each pair of points is evaluated independently based on their travel time and potential time savings. As a result, the algorithm may identify neighboring points with significantly reduced travel time between them. In this case, the static method would suggest connecting these points one by one, leading to a clustering of new edges in specific areas.The appearance of this clustering effect is due to the static method's lack of considering the global network dynamics and interactions between different points. It treats each pair of points in isolation, without considering the potential chain reactions in travel time by connecting a pair to other points.On the other hand, the dynamic method (Strategy 3) takes into account the changes in travel time after each new edge is added. By considering the updates in travel time, it can identify cases where connecting additional points may not result in significant time savings

compared to the initial connections. This allows the dynamic method to make more efficient decisions and avoid excessive clustering of new edges in certain areas.In summary, the static method fails to consider the dynamic changes in travel time, leading to clustering of new edges in close proximity. However, the dynamic method can adaptively select edges that provide the maximum time savings, considering the evolution of network conditions and mitigating this clustering effect.

c) 1 vs 4 - Compare and analyze the results. Which is better? Why?

Strategy 4 aims to optimize by reducing the maximum extra traveling time. It calculates the extra time between locations by considering the difference between the shortest traveling time and the straight-line travel time, accounting for travel speed.When comparing Strategy 1 and Strategy 4, they are considered to have similar performance in terms of time complexity, both being O(|V|^2 log |V|) in Big-O notation. However, Strategy 4 requires computing the shortest distance and the travel time for each pair separately, which increases the actual running time.

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

i) If yes, which one and why?

Based on the comparisons made, the dynamic strategy is considered the optimal choice. Dynamic strategies, such as Strategy 3 and Strategy 5, take into account the changes in the network's conditions and interactions between points. They can adaptively select new edges that provide significant time savings, avoiding excessive clustering in specific areas and improving the overall road network's efficiency.

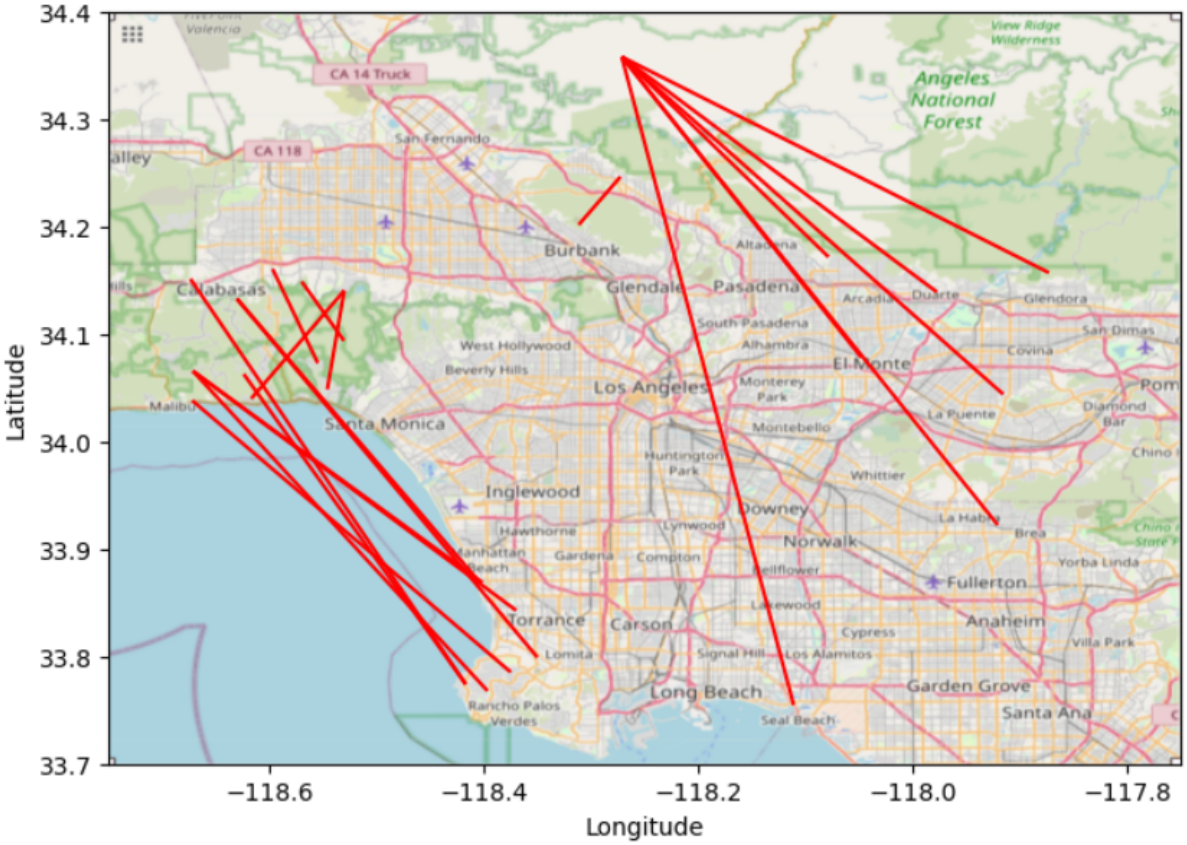
ii) If not, what would be a better strategy to construct roads, and is it optimal? What is the time complexity? (Assuming that you want to reduce traveling distances and you know beforehand that we have to construct 20 new roads).

If neither the static nor the dynamic strategies are considered optimal, a better strategy could be to combine both approaches. This hybrid strategy could involve an initial evaluation of potential new edges using a static method (e.g., Strategy 1 or Strategy 4) to identify pairs with high extra distance or time. Subsequently, a dynamic method (e.g., Strategy 3 or Strategy 5) can be applied to select the most impactful edges while considering the network's evolving conditions. Determining the exact time complexity of this hybrid strategy would depend on the specific implementation details and the algorithms used. However, it would likely involve an additional computational overhead compared to the individual static or dynamic strategies.

e) 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.

Our new strategy, Dynamic Time-Weighted Optimization(Time+Dynamic+weighted), focuses on addressing the real-world situation where travelers prioritize saving travel time over distance. The new top 20 pairs with the highest weighted extra time with the source and destination of

these pairs are shown below, along with the graph of new edges, are displayed below. On the assumption that traffic congestion significantly affects travel time, road capacities influence traffic flow, and certain routes or areas have higher time sensitivity, we aim to optimize the road network for maximum efficiency. By considering real-time traffic conditions, road capacities, and user preferences, our strategy incorporates these factors into the optimization process.The justification for this strategy lies in its ability to prioritize routes that minimize travel time, accounting for the impact of congestion and assigning appropriate weights to factors such as road capacity and time sensitivity. By considering the dynamic nature of traffic conditions and the evolving needs of travelers, the strategy ensures the road network remains adaptive and responsive. This approach acknowledges the importance of time-saving in transportation and aims to provide an enhanced travel experience by optimizing routes based on real-time data and weighted factors. By doing so, we can improve overall transportation efficiency and increase traveler satisfaction.



Top 20 pairs with highest weighted extra time (dynamic): (time, v, s)
[(1014718.1403566865, 987, 1510), (956123.1029802257, 1783, 2415), (955995.7495816515, 1859, 2416), (935908.7285159945, 383, 2419), (852900.5168150081, 2419, 2462), (848314.7167245926, 2079, 2419), (837372.5125255233, 1783, 2417), (810719.9011159596, 988, 1678), (764394.2363749143, 286, 2419), (703964.4187339406, 299, 2419), (675969.4998477531, 1799, 2419), (644080.1587683879, 1699, 1858), (627520.251889558, 2025, 2416), (611045.8936931874, 1003, 1700), (609071.4662387567, 427, 2416), (601809.2941042549, 120, 689), (579310.3656274902, 144, 2043), (575648.9056634746, 1003, 1511), (574048.6241724751, 144, 426), (563329.0003085762, 680, 2419)]

edge 1: (-118.59730984444444, 34.16049380000001) - (-118.55510068503939, 34.07289098425196)

edge 2: (-118.41714115454548, 33.774103109090916) - (-118.67397635416673, 34.15129022916667)

edge 3: (-118.39683944230771, 33.7679271025641) - (-118.67145511923503, 34.06533878290219)

edge 4: (-117.8732378168498, 34.15698468864466) - (-118.27158981797648, 34.35782482248528)

edge 5: (-118.27158981797648, 34.35782482248528) - (-117.92094835714286, 33.92228328571428)

edge 6: (-117.91583063565888, 34.04390227131781) - (-118.27158981797648, 34.35782482248528)

edge 7: (-118.41714115454548, 33.774103109090916) - (-118.62418590723568, 34.06269522170691)

edge 8: (-118.57017334070797, 34.14872431415929) - (-118.53067983104125, 34.09368131827117)

edge 9: (-117.97771498148147, 34.13897174074074) - (-118.27158981797648, 34.35782482248528)

edge 10: (-118.04825531914894, 34.07820174468085) - (-118.27158981797648, 34.35782482248528)

edge 11: (-118.07879766552904, 34.17167453242318) - (-118.27158981797648, 34.35782482248528)

edge 12: (-118.67225393333337, 34.03809095384618) - (-118.3751182100841, 33.785309033613444)

edge 13: (-118.37011925757571, 33.84274427272727) - (-118.67145511923503, 34.06533878290219)

edge 14: (-118.53047408659789, 34.14032269896909) - (-118.61724170318024, 34.03975774911659)

edge 15: (-118.40144004411762, 33.86863104411765) - (-118.67145511923503, 34.06533878290219)

edge 16: (-118.31185060431655, 34.20150541726619) - (-118.2728725826087, 34.24626368695651)

edge 17: (-118.63048660074622, 34.1327155932836) - (-118.3502318076923, 33.79901333846155)

edge 18: (-118.53047408659789, 34.14032269896909) - (-118.54625915050165, 34.048403046822735)

edge 19: (-118.63048660074622, 34.1327155932836) - (-118.40540964161855, 33.870485456647394)

edge 20: (-118.11129299342103, 33.75486076973685) - (-118.27158981797648, 34.35782482248528)

QUESTION 25: Own task

Answer:

1. Introduction

After Brainstorm, in this part, we decided to use other data from the previous work website to conduct an exploration of travel time prediction based on first quarter data of 2020 from the "Uber Movement" website and download data from Travel Times by Month (All Days), 2020 Quarter 1, for the Los Angeles area. We also used Question 17 method to trim the extra/useless edges. The dataset contains pairwise traveling time statistics between most pairs of points in the Los Angeles area. The goal is to devise effective models that can accurately **predict** travel time. We use MSE(mean squared error) as our evaluation metric.

2. Methodology

2.1 Baseline Method

The baseline method for travel time prediction involves the assumption that travel time is directly proportional to the Euclidean distance between two points. The proportionality factor is determined by the sum of all travel times divided by the sum of all Euclidean distances. By applying this baseline method, we obtained a Mean Squared Error (MSE) of 8509.2, which serves as a starting point for comparison with our advanced models.

2.2 Graph Convolutional Network (GCN)

The first advanced model we use is the Graph Convolutional Network (GCN). We set up a graph where nodes represent points, and edges represent travel paths, with edge weights corresponding to travel times. We utilize a 5-fold cross-validation strategy for training, and the model is tasked with learning to predict the weights of edges. The results are obtained by taking the average MSE across the 5 folds.

2.3 Graph Attention Network (GAT)

The second model is the Graph Attention Network (GAT). Similar to GCN, GAT employs an attention mechanism that allows nodes to assign different levels of importance to their neighbors. The same 5-fold cross-validation strategy is applied to this model.

2.4 GraphSAGE

The third model is GraphSAGE, a model that is particularly adept at capturing local neighborhood information by sampling a fixed number of neighbor nodes. This is especially helpful for training on graphs with a large number of neighbor nodes. GraphSAGE is also subject to the same 5-fold cross-validation procedure.

3. Creativity

Beyond the traditional methods of employing Graph Neural Networks, we experimented with additional node features to enhance our models. Specifically, we introduced the concepts of Node Degree and Clustering Coefficient into our Graph Convolutional Network (GCN) model.

3.1 Incorporating Node Degree

The degree of a node is a basic network feature that represents the number of connections a node has with others in the network. Intuitively, nodes with higher degrees could potentially have higher traffic, thus influencing the travel time. Hence, we decided to extend our model by integrating the node degree as an additional node feature.The same 5-fold cross-validation strategy is applied to this model.

3.2 Incorporating Clustering Coefficient

Going a step further, we decided to add the Clustering Coefficient as another feature to our model. The Clustering Coefficient is a measure of the degree to which nodes in a graph tend to cluster together. It captures the local density of the network, i.e., how interconnected a node's neighbors are. We hypothesized that the local clustering of a node could have an impact on the travel time, especially in situations where there are high traffic areas or road networks. After adding the Clustering Coefficient to our model, we again conducted five-fold cross-validation to evaluate the performance.

4. Results

The GCN, GAT, and GraphSAGE models were trained using 5-fold cross-validation, and the results showed an improved performance compared to the baseline method. The GCN is among the best, showing a gain of performance of 5.67%. So we use GCN to add features.

The model's performance with both Node Degree and Clustering Coefficient features was superior to the original GCN and GCN with only node degree. These results demonstrate the benefits of augmenting our model with additional node features, such as Node Degree and Clustering Coefficient, to better capture the complexity of the network. They also highlight the flexibility and versatility of graph neural networks in adapting to different data features for various tasks.

	base line	GCN + coordinates	GAT + coordinates	GraphSAGE + coordinates	GCN + coordinates + node degree	GCN + coordinates + node degree + clustering coefficient
fold 1	-	8196.2	8274.1	8257.7	8207.3	8188.2
fold 2	-	7972.6	7965.6	7957.0	7970.3	7968.8
fold 3	-	7721.2	7639.6	7637.2	7723.7	7711.3
fold 4	-	8319.2	8376.6	8371.3	8309.0	8297.7

fold 5	-	7925.8	7981.5	7991.1	7916.5	7912.9
average mse	8509.2	8027	8047.5	8042.9	8025.4	**8015.8**
diff	-	5.67%	5.43%	5.48%	5.69%	**5.80%**

5. Conclusion

The experiment demonstrates that graph neural networks can be effective in predicting travel time. Although the complexity of these models is higher than the baseline model, their performance improvement indicates the potential for more accurate travel time prediction when additional features are available. However, our prediction results are still not quite satisfactory, only marginally outperforming the baseline. This might be due to the limited information we have - only the graph structure and point coordinates. Intuitively, if more features, such as traffic conditions, road types, or historical travel time data, were included at each node, the predictive performance could be significantly improved. Future work could involve exploring other types of graph neural networks or integrating additional features into the existing models.