Graph Algorithms

William Nafack UID : 405725778 Lea Alcantara UID : 005872120
Gaurav Singh UID : 305353434

June 11, 2022

1 Stock Market

1.1 Question 1

What are the upper and lower bounds?

The upper and lower bounds of pij are [-1,1]. Log-normalized return should be used instead of regular return for several reasons. Log-normalized return will ensure that there is log-normality, which will help to justify and convert this exponential problem to one of a linear. The stock prices will be ensured to be normally distributed. This also will help justify the property of time adding. Because of this, the time complexity will be reduced. Additionally, the outliers will be taken care of.

1.2 Question 2

Constructing Correlation Graphs We are able to compute the edge weights for the correlation graphs where we can then plot the un-normalized distribution of such.

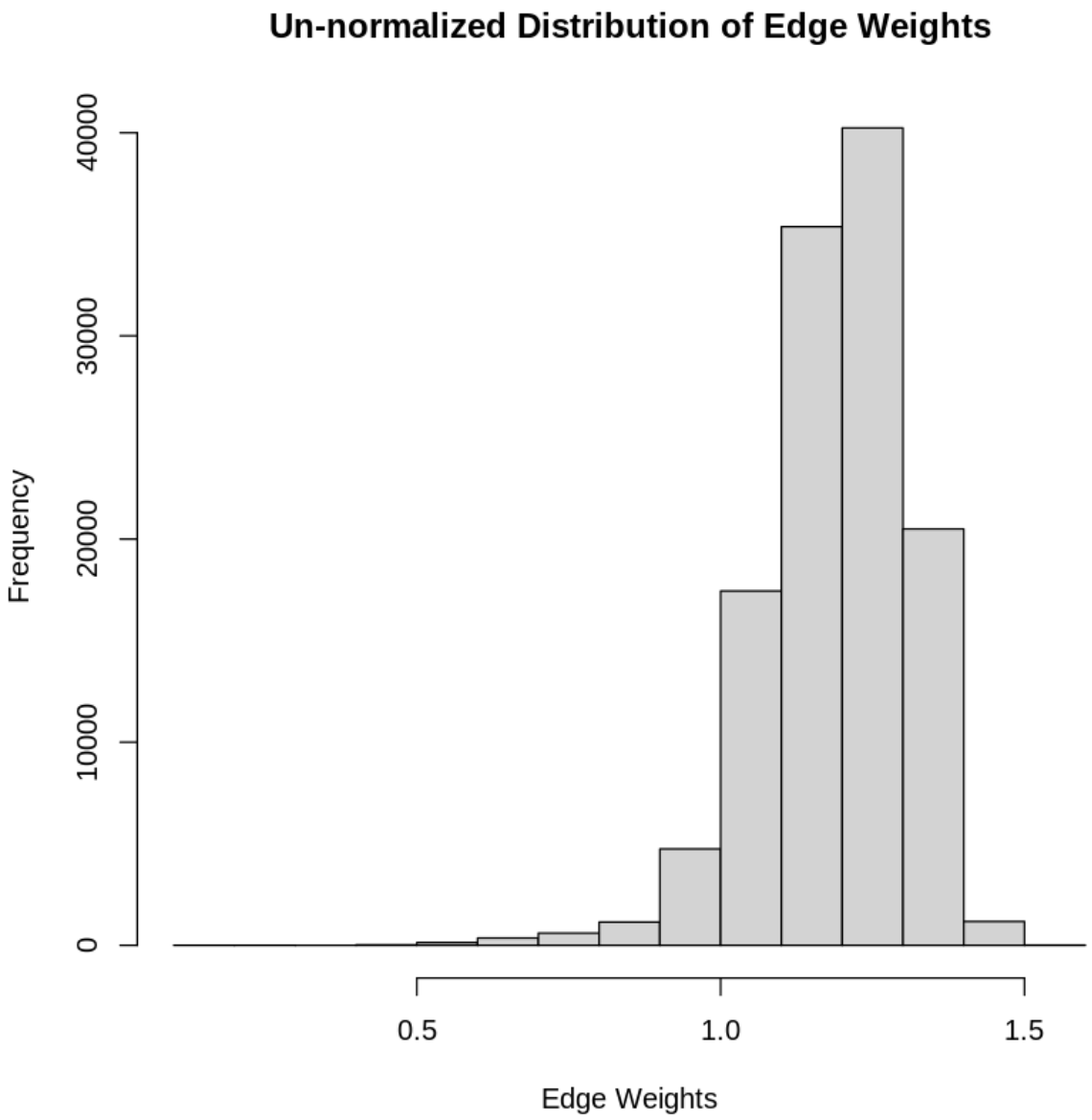
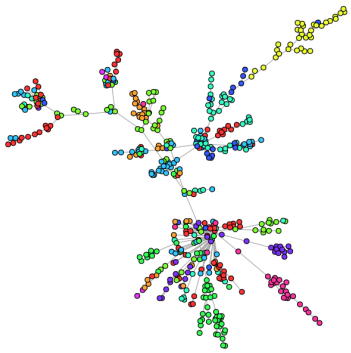


Figure 1: Histogram of Un-Normalized Distribution of Edge Weights

1.3 Question 3

MST of Correlation Graph

In question 3, we looked at the MST (minimum spanning tree) 2a of the correlation graph–utilizing calculations from the previous question. We found this by splitting up into sectors which are defined within the Name sector file given. The associated colors are defined in the legend 2b. From this plot, we can interpret that there is a pattern seen in the MST. We can identify that the sectors are grouped within similar nodes. This means that the nodes with the same colors, that are representative of the same sector, are more likely to be connected than those in different sectors and have higher correlation and lower weights than the latter. The vine clusters typically have the same sector node colors grouped together. It is true also that all of the nodes from the same sector are not necessarily in the same location, but along the same vine cluster it is more likely to see only nodes from the same sector.



(a) MST of Correlation Graph

(b) MST of Correlation Graph

1.4 Question 4

Community Detection Algorithm The community detection algorithm from the MST correlation graph is plotted below 3. Homogenity: **0.682644648161366**; Completeness: **0.479284479244588**

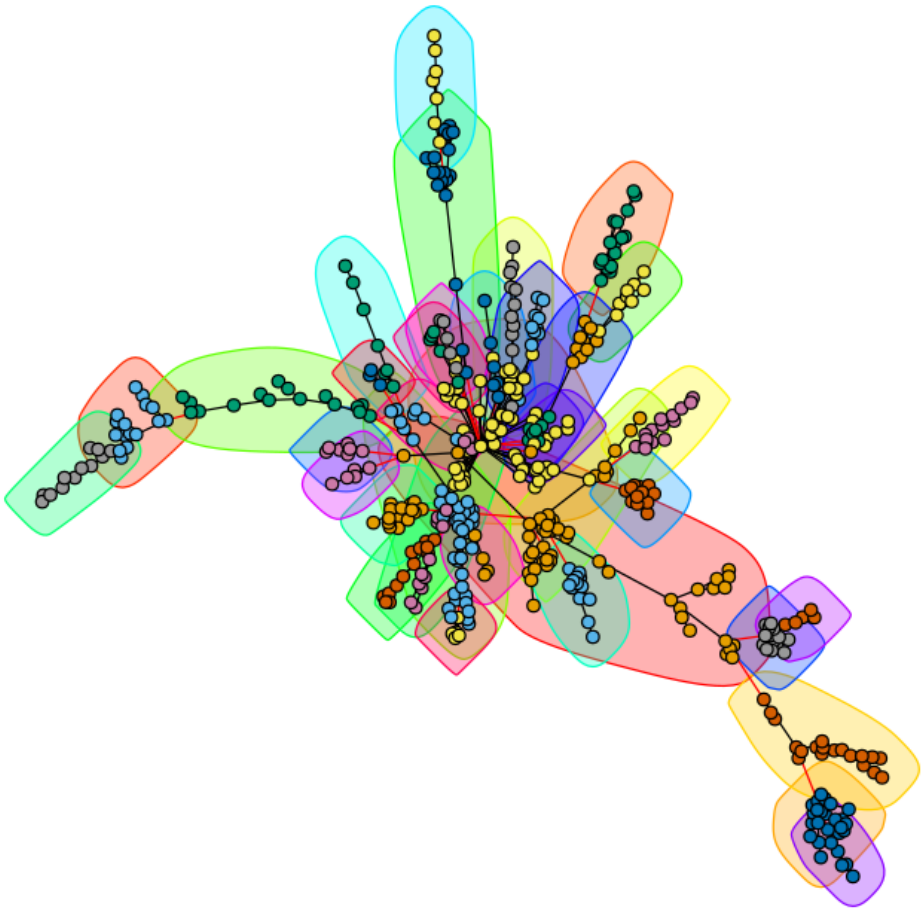


Figure 3: Community Detection Algorithm on MST of Correlation Graph

1.5 Question 5

Alpha 1, Weekly Data:0.112957284810943 Alpha 2, Weekly Data: 0.11470933306894

The difference between the two calculations for alpha is the difference in the calculation of the probability. For the second alpha, the larger one, we divide the nodes per sector by the nodes. The first alpha, smaller one, is the neighbors in the same sector divided by the total number of neighbors so it is only considering those neighbors within the V node.

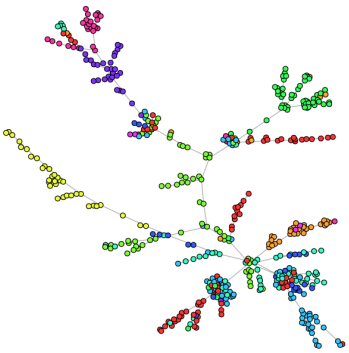
1.6 Question 6

Weekly data

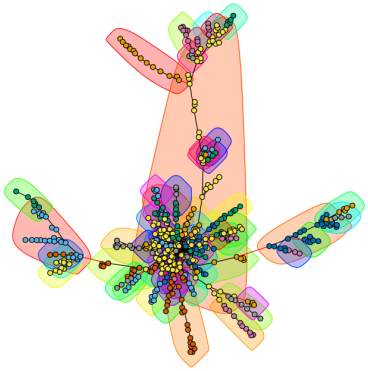
In order to create our weekly graph, we had to get just the Mondays so that we could best isolate the

grouping for the stock values. We can see from the plot in 5a that there are more groupings and the vine clusters are similarly clustered by the node color or the sector of the stock values. However, now as we group by week, we can tell that there exists an even stronger correlation amongst the similar nodes.

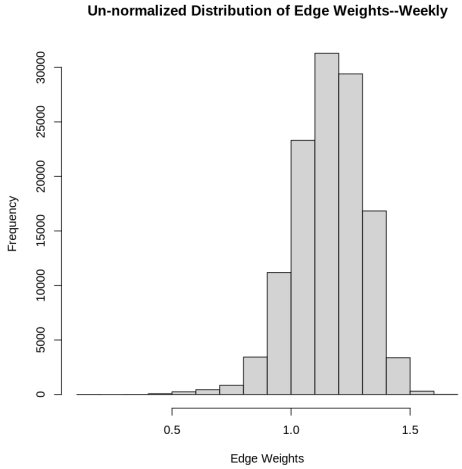
It makes sense then that the clustering are closer together and has nodes that are much more separated than they were with the previous plot for daily.



(a) MST of Correlation Graph on Weekly Data



(b) Community Detection Algorithm of MST of Correlation Graph on Weekly Data
Homogeneity: **0.273974094873204**; Completeness: **0.177481879309562**



(c) Histogram of Un-Normalized Distribution of Weekly Data

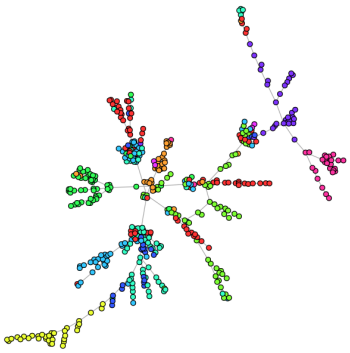
The alpha values here follow the opposite trend as we saw for the alpha values for the daily data. This has the alpha 1, or the alpha which considers only the v neighbors nodes being significantly higher than our calculation for the second alpha. This is logical because of the fact that the clusters for the weekly data are more separated so the calculation for one on the individual node for V would be more pronounced and thus higher.

Alpha interpretation: **Alpha 1: 0.828930077530676 Alpha 2: 0.114188070612533**

1.7 Question 7

Monthly Data

In order to plot the monthly data, we had to extract from the month day of 15, just as we did for week separation with the Monday isolation. This in turn, allows us to better interpret the overall month trends so that we can get the p ij values. The groupings go in terms of months. We can understand that the stock prediction shows that halfway through the month, the stocks are more likely to fall in their prices, so that it is easier to predict the values here at this point–it serves as a turning point.

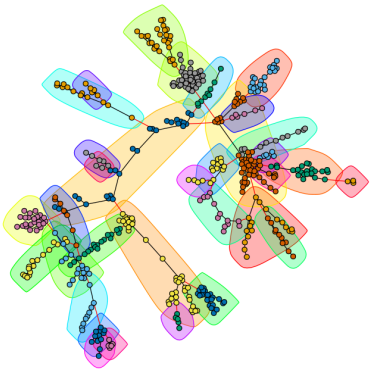


The greatest pattern we can identify from the MST plot of the

(a) MST of Correlation Graph on Monthly Data

monthly data is that there are less category groupings for the month than there are in the weekly or daily because there are less occurrences of the p ij values for 15 than there would be for the p ij occurrences for Monday, for example. This makes the structure of the MST different. There are less groupings for the sectors in the 12 months (except where there are holidays on the 15th), than there are days of the week that do not have holidays on Mondays. This establishes the MST plot nodes to have less defined vine clustering than that of the weekly. The nodes stay associated with the sector, but there exists a stronger correlation with the month and the node it is associated with.

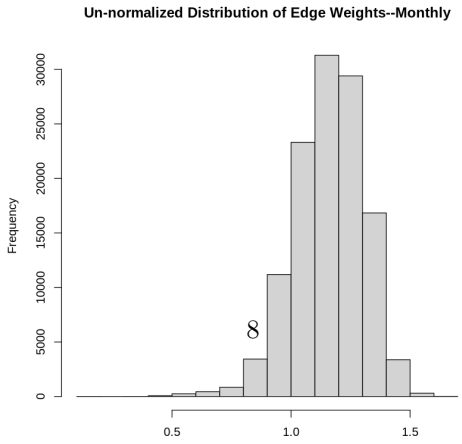
We can see that there exist some of the vine clustering that we saw before, but there exists less correlation within node groupings.



Homogenity: **0.573481879309562**; Completeness:

(b) Community Detection Algorithm of MST of Correlation Graph on Monthly Data

0.162974094873204



The alpha values increased from the weekly data. This is because the predictor of the day of the week, is strong with the Monday prediction, but as we are taking into account the sector it belongs to and the neighboring nodes because of it, we can better predict then within the month. As seen before, the alpha calculation with the first method places more emphasis on the set of neighbors and the second method with more emphasis on the sector. The neighboring then we can expect to be higher than the alpha 2–with an opposite impact as that of the weekly values.

Alpha 1, Monthly Data:0.771270933306894 Alpha 2, Monthly Data: 0.36156284810943

1.8 Question 8

As explained within questions 6 and 7, we know that the trend for weekly data versus monthly data is quite different. The stock predictions for Mondays (daily) have a strong correlation within the neighboring fields as we saw through the first method of alpha calculation. Mondays and the day of the Month being 15 are two of the strongest correlations. On the 15th day, we can understand that there is another high correlation with the stock prices. The monthly correlation as it relates to the sectors, we can understand this from the second alpha calculation which relies more on the sector, is strong within each month. This is because the stock trends per sector tend to go along in seasons as well as with the day of the week. However there is a stronger correlation of sector to the season than there is node to the day of the week. Overall, the **correlation between stocks is decreasing and additionally clustering in the monthly data decreases**. We can understand that in terms of **granularity, there is a decrease in granularity** from weekly to monthly data.

2 Let's Help Santa

2.1 Question 9

The giant connected component of the graph contains **2649** nodes and **1003858** edges.

2.2 Question 10

We create the Minimum Spanning Tree (MST) of the graph as shown below in figure 6.

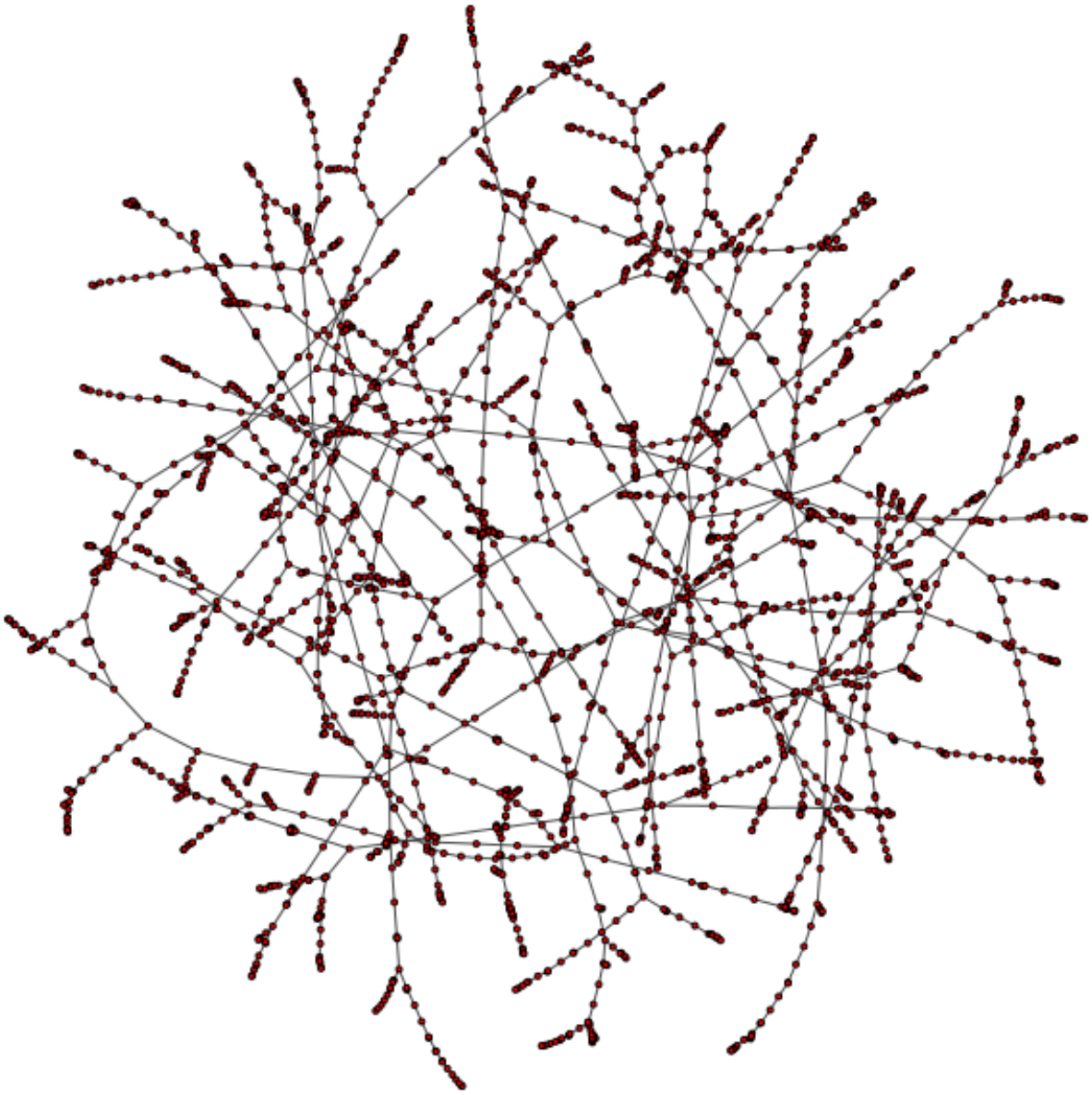


Figure 6: Minimum Spanning Tree of Graph G

Census Tract (a)	Coordinates (a)	Census Tract (b)	Coordinates (b)	Street Address(a)	Street Address (b)
Census Tract 480302	[-118.12053321 34.10309557]	Census Tract 480304	[-118.13138209 34.09626386]	Sacred Heart Retreat House - Saint Joseph Campus, 507 North Granada Avenue, Alhambra, CA 91801, United States of America	269 Woodward Avenue, Alhambra, CA 91801, United States of America
Census Tract 480302	[-118.12053321 34.10309557]	Census Tract 481002	[-118.11656383 34.09585388]	Sacred Heart Retreat House - Saint Joseph Campus, 507 North Granada Avenue, Alhambra, CA 91801, United States of America	320 Cordova Street, Alhambra, CA 91801, United States of America
Census Tract 480303	[-118.13785063 34.09645121]	Census Tract 480400	[-118.13224544 34.10349303]	402 North Marguerita Avenue, Alhambra, CA 91801, United States of America	926 North Garfield Avenue, Alhambra, CA 91801, United States of America

Table 1: Table showing Street Address at centroid location.

Table 1 below report the streets addresses near the centroid locations of a few edges from the MST.

From the table above we observe that the street address of endpoints are very close to each other. This makes sense intuitively because the MST connects all the addresses such that the lowest cumulative mean travel time is achieved. This is done by connecting the nodes that have a small distance between them as the mean travel time will be impacted by that and also be smaller. Hence by choosing edges with the smallest weights we end up with nodes that are relatively close to each other.

2.3 Question 11

In the triangle Inequality the sum of lengths of any two sides of a triangle is greater than that of the third side. Using this we randomly sample 1000 sets of 3 points and calculate the percentage of points that hold this inequality rule. **91.2%** of the triangles in the graph are valid.

2.4 Question 12

TSP consist of an undirected weighted graph where the nodes are locations and edges represents the streets with the edge weights representing the distance between the locations. We make use of the 1-approximation algorithm to find the upper bound of the TSP. The algorithm goes as follows.

- Find the Minimum Spanning Tree (MST)
- Create a multi-graph from the MST
- Find the Euler cycle in the multi-graph to create a tour sequence
- Search for each edge in tour sequence and use Djikstra's algorithm to find the shortest path between the nodes that contain the edges.
- Find the approximate TSP cost given by the weighted sum of the edges in the tour sequence.

We find our MST cost to be **269085** , approximate TSP to be **421489** and upper-bound value ρ to be **1.57**

2.5 Question 13

Figure 7 below shows a plot of the trajectory that Santa has to travel.

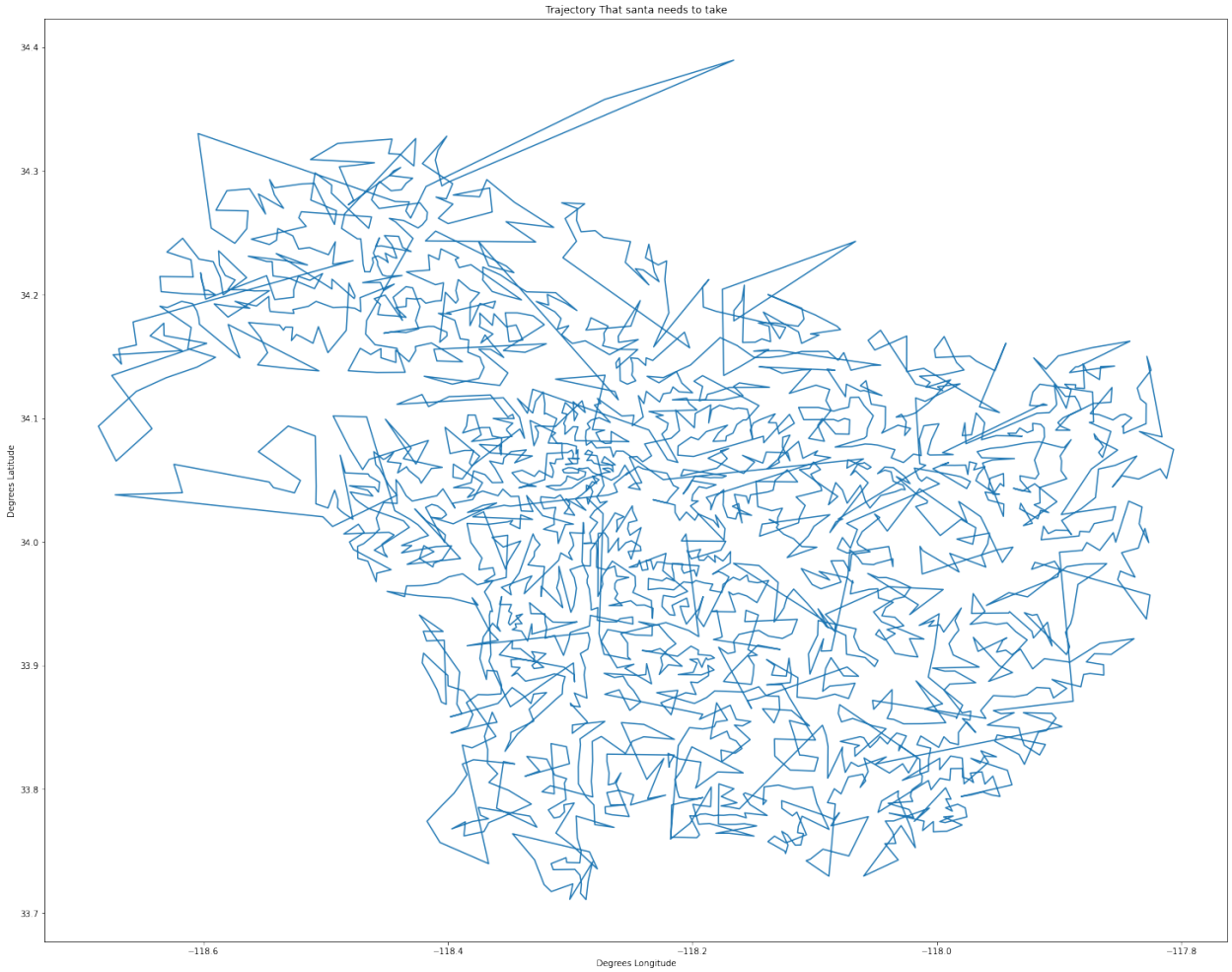


Figure 7: Santa Trajectory

2.6 Question 14

Delaunay Triangulation is an algorithm used to estimate the maps of streets without any information about them. It is a triangulation algorithm for a given set of discrete points such that no points in the set of points lies within the circumcircle of any triangle. Figure 8 below shows a plot of the Road mesh of LA using Delauney triangulation. Figure 9 shows the Graph G_Δ produced from this process.

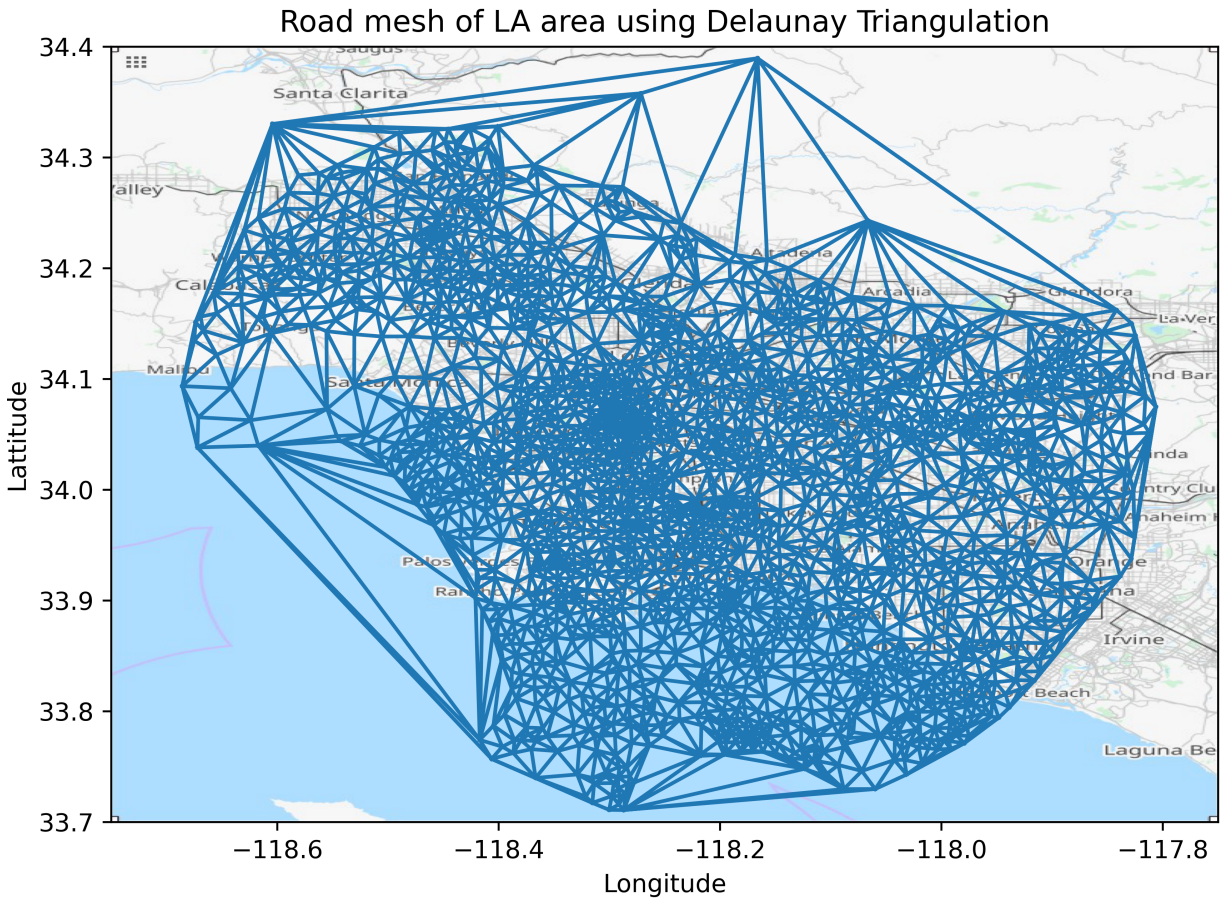


Figure 8: Road Mesh of LA using Delaunay Triangulation

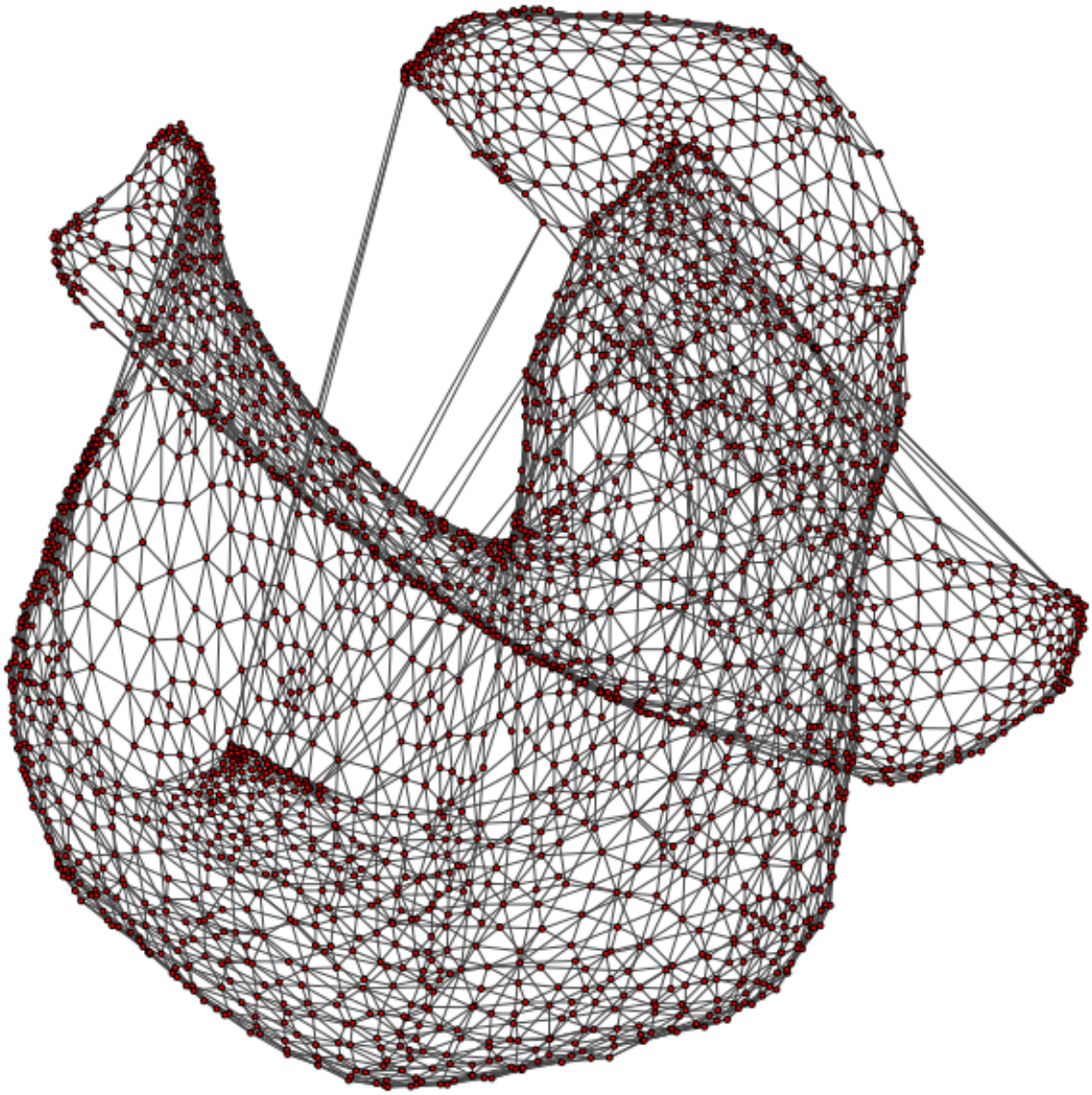


Figure 9: Graph produced from the Triangulation edges

From Figure 8 we notice that most of the road structures of LA has been extracted from downtown area. We also see that there are roads going over the mountains and oceans which do not exists. This is because of how the Delaunay Triangulation algorithm works. Delaunay Triangulation tries to fit every triangle inside a circumcircle. Figure 9 gives a clearer representation of the nodes representing the locations and edges created by the triangulation process.

2.7 Question 15

Traffic flow derivation is as follows:

- Distance = $\frac{Velocity \quad of \quad Car \times Mean \quad Travel \quad Time}{3600}$

- Distance between cars =
$$0.003 + \frac{2 \times Velocity \quad Of \quad Car}{3600}$$
- Number of cars =
$$\frac{2 \times Distance}{Distance \quad between \quad cars}$$
- Traffic Flow (cars/hours) =
$$\frac{60 \times 60}{Mean \quad Travel \quad Time} \times Number \quad of \quad cars$$

  Traffic flow derivation is given by Traffic Flow =
$$\frac{3600 \times Velocity \quad of \quad Car}{5.4 + Velocity \quad of \quad Car}$$

  where Velocity of Car is:
$$\frac{69}{Mean \quad Travel \quad Time} \times \sqrt{(latitude_{point1} - latitude_{point2})^2 + (longitude_{point1} + longitude_{point2})^2}$$

2.8 Question 16

The max number of cars per hour between Malibu [34.04,-118.56] and Long Beach [33.7,-118.18] is **11074** cars.The minimum number of edges coming in the node and out the node is the number of edge-disjoint paths. The number of disjoint paths between the two spots is 4. Figure 10 demonstrate that. We can see that Malibu has 4 incoming and out-going edges.

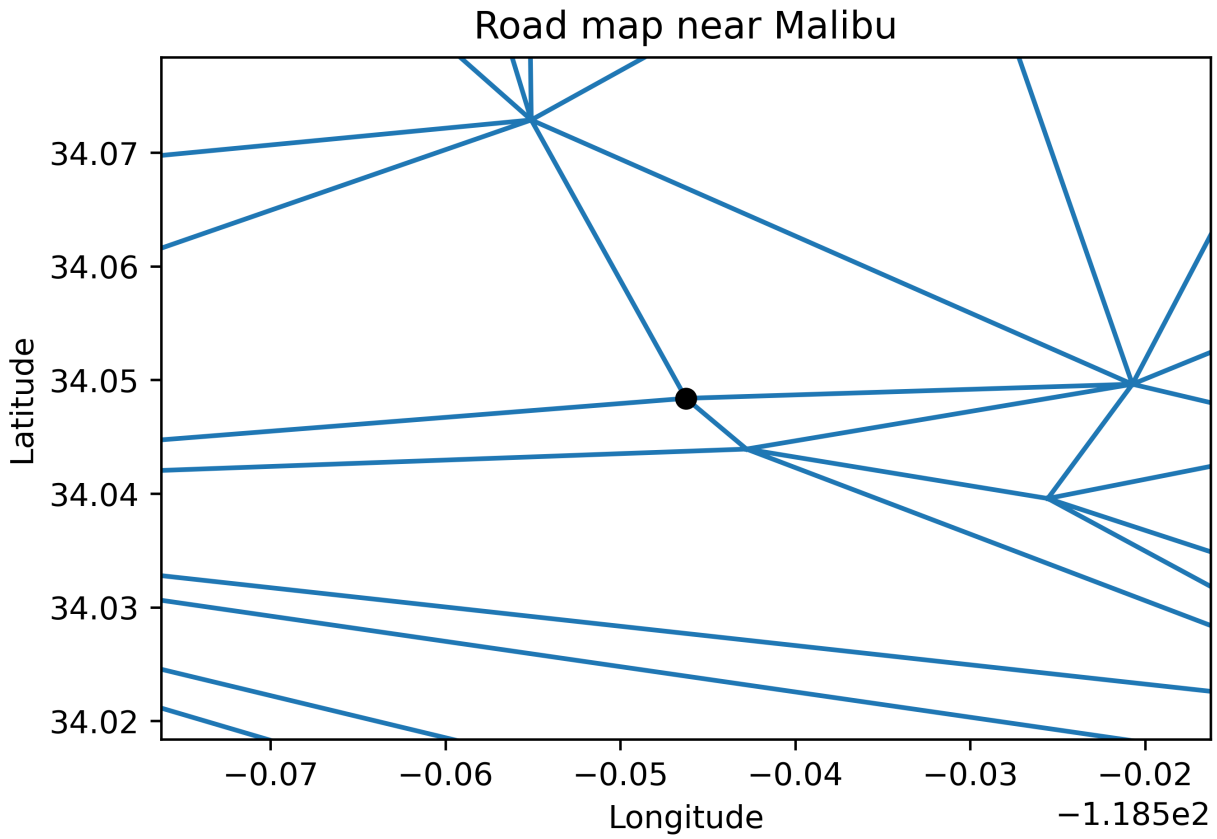


Figure 10: Road Map near Malibu

2.9 Question 17

To prune the graph we apply a threshold of **800** on the travel time of the roads via the induced graph .With this threshold, some of the large edges created by the Delaunay Triangulation to fit triangles within the circumcircle are removed. Figure 11 and 12 shows the resulting road map of and graph $\tilde{G}_\Delta$ of LA after the thresholding process.

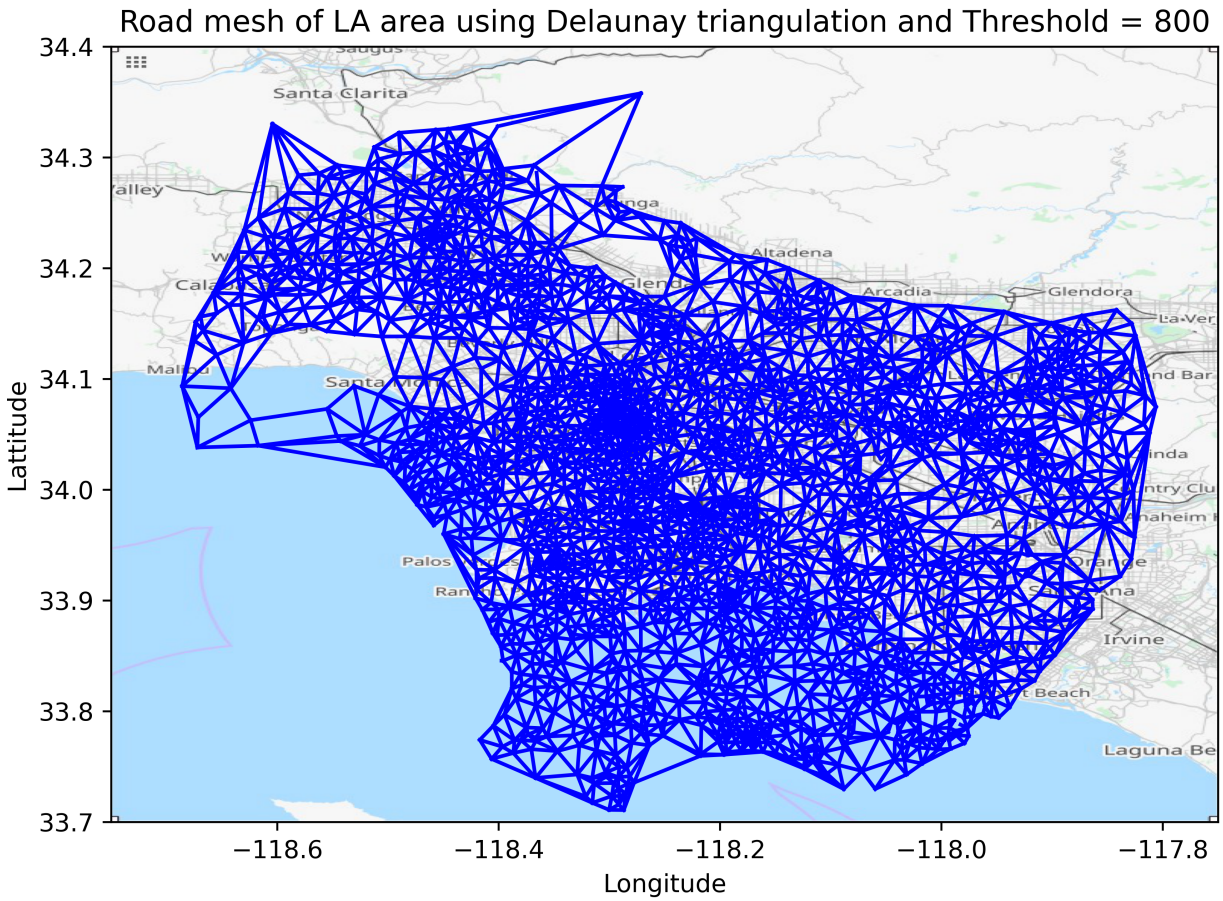


Figure 11: Road mesh of LA area using Delaunay Triangulation with Threshold (800)

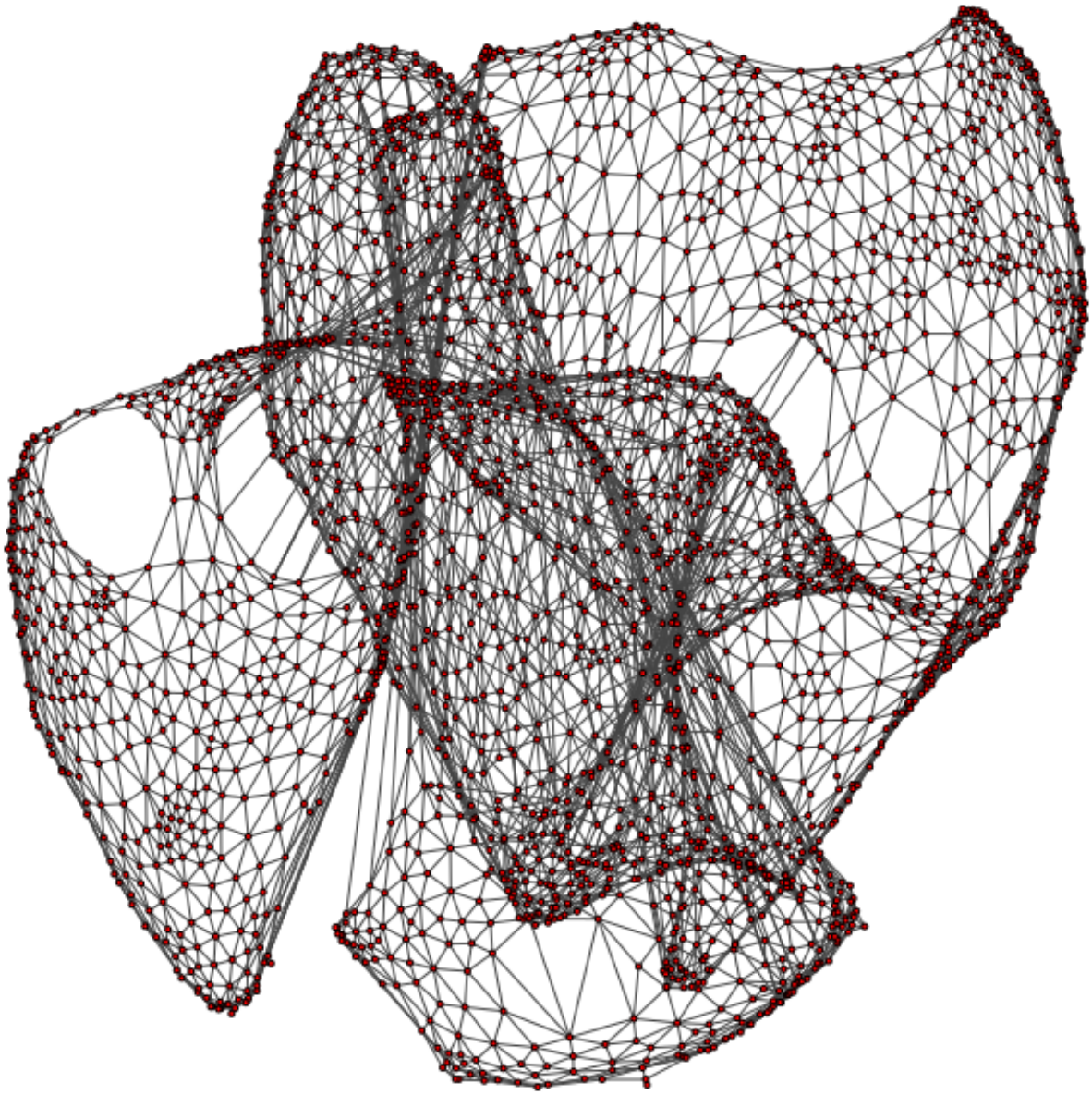


Figure 12: Graph produced from the Triangulation edges with Threshold (800)

From figure 11 the non existing routes over the ocean and the Topanga mountains have been removed. This shows that the Thresholding process worked. The road mesh looks very close to the Los Angeles road map with the edges bordering the coast and not crossing over into the oceans.From Figure 12 we see that edges that were very long but connected neighbouring nodes have been removed.

2.10 Question 18

After pruning the graph, we did not see a change in the amount of disjoint paths.The number of disjoint paths between the two spots remained **4**. The maximum flow has not changed and remained to be **11074** cars/hour . This could be due to the fact that there are multiple paths that exist between Long Beach and Malibu that cars can take which means the capacity of the road is still large enough. It also has to do with how the max-flow algorithm operates. The maximum flow is sum of the flow across the minimum cuts which

are edges with the least weights. It is unlikely that the non-existent paths will have the lowest weights and hence not contribute to the maximum flow at all hence why there would be no significant difference in the maximum flow.

3 Construct New Roads

*All the results are computed on the Pruned graph from Question 17.Time

3.1 Question 19

Strategy 1: To find top 20 pairs of vertices which maximise the extra distance given as:

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

The source and destination of the new roads found using Strategy 1 are:

Source	Destination
Census Tract 800504	Census Tract 670328
Census Tract 400501	Census Tract 930200
Census Tract 800504	Census Tract 670416
Census Tract 400404	Census Tract 930200
Census Tract 400501	Census Tract 930200
Census Tract 670328	Census Tract 800504
Census Tract 401202	Census Tract 930200
Census Tract 401202	Census Tract 930200
Census Tract 400402	Census Tract 930200
Census Tract 800504	Census Tract 670416
Census Tract 404202	Census Tract 930200
Census Tract 401001	Census Tract 930200
Census Tract 401201	Census Tract 930200
Census Tract 401312	Census Tract 930200
Census Tract 404000	Census Tract 930200
Census Tract 800202	Census Tract 670328
Census Tract 403802	Census Tract 930200
Census Tract 403802	Census Tract 930200
Census Tract 930200	Census Tract 401303
Census Tract 401203	Census Tract 930200

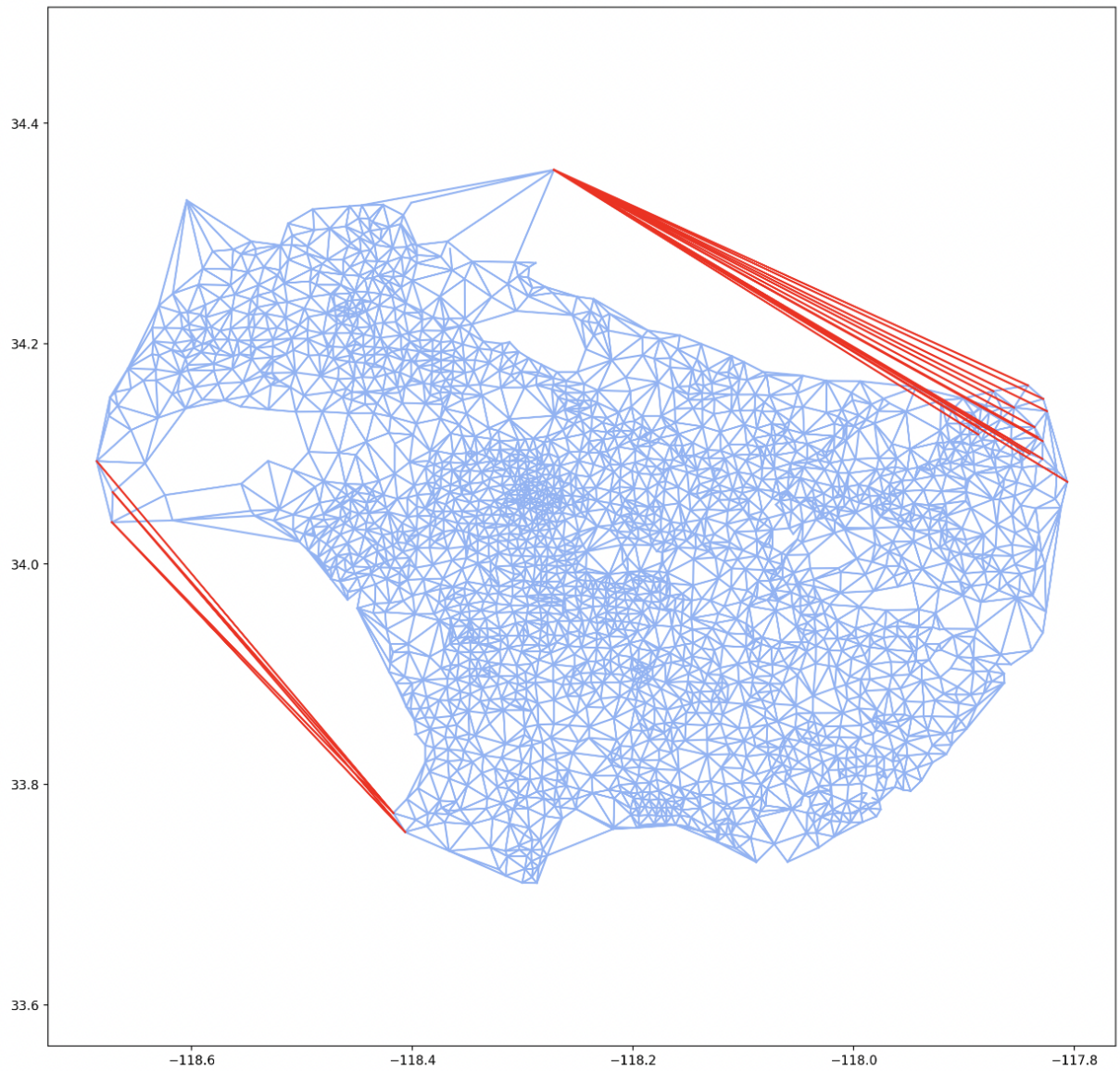


Figure 13: Resultant graph with Strategy 1 impact

Time Complexity of the strategy 1:

If n is number of vertices with 20 new roads that need to be added.
Time to find all pairs shortest path: n^3
Time to find all pair euclidean distances: n^2
Computing extra distance and sorting extra distances = $n + nlog(n)$
Total time : $n^3 + n^2 + n + nlogn$

3.2 Question 20

Strategy 2: To find top 20 pairs of vertices which maximise the weighted extra distance given as:

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

where frequency(v, s) denotes the demand of one edge in network.
The source and destination of the new roads found using Strategy 2 are:

Source	Destination
Census Tract 404201	Census Tract 930200
Census Tract 402403	Census Tract 930200
Census Tract 403802	Census Tract 930200
Census Tract 400501	Census Tract 930200
Census Tract 404201	Census Tract 930200
Census Tract 403901	Census Tract 930200
Census Tract 403323	Census Tract 930200
Census Tract 403312	Census Tract 930200
Census Tract 405701	Census Tract 930200
Census Tract 800203	Census Tract 670416
Census Tract 401002	Census Tract 930200
Census Tract 403703	Census Tract 930200
Census Tract 405600	Census Tract 930200
Census Tract 137504	Census Tract 262601
Census Tract 401201	Census Tract 930200
Census Tract 134002	Census Tract 262601
Census Tract 800504	Census Tract 670413
Census Tract 404100	Census Tract 930200
Census Tract 403703	Census Tract 930200
Census Tract 930200	Census Tract 001505

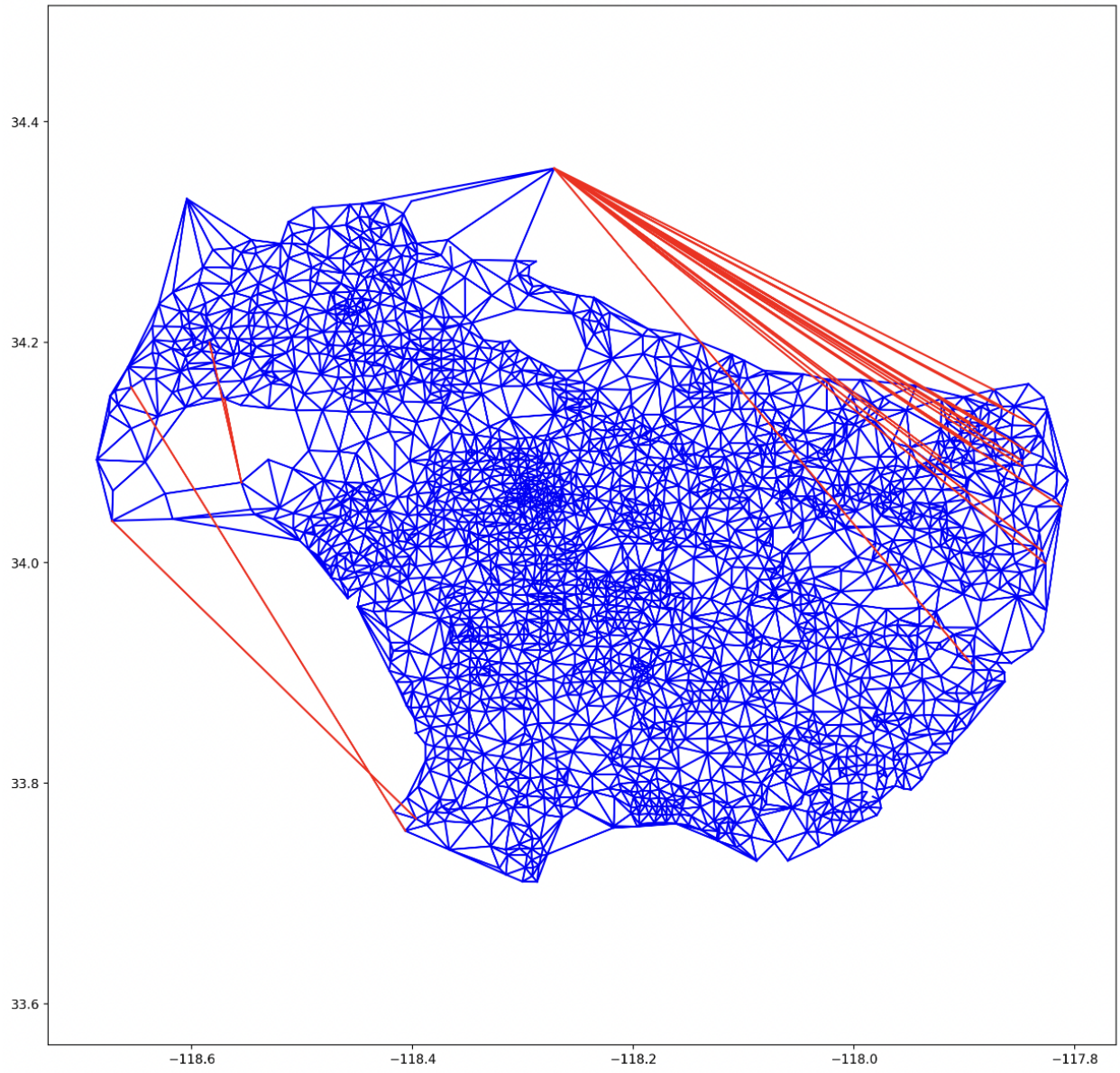


Figure 14: Resultant graph with Strategy 2 impact

Time Complexity of the strategy 2:

If n is number of vertices with 20 new roads that need to be added.
Time to find all pairs shortest path: n^3
Time to find all pair euclidean distances: n^2
Computing weighted extra distance and sorting them = $n + nlog(n)$
Total time : $n^3 + n^2 + n + nlogn$

3.3 Question 21

Strategy 3: To find top 20 pairs of vertices which maximise the extra distance with dynamic updates to graph and resulting metrics at each modification.

The source and destination of the new roads found using Strategy 3 are:

Source	Destination
Census Tract 670328	Census Tract 800504
Census Tract 400501	Census Tract 930200
Census Tract 137501	Census Tract 262601
Census Tract 404301	Census Tract 930200
Census Tract 430003	Census Tract 930200
Census Tract 430301	Census Tract 930200
Census Tract 670328	Census Tract 800506
Census Tract 430501	Census Tract 930200
Census Tract 461200	Census Tract 930200
Census Tract 139802	Census Tract 980019
Census Tract 800203	Census Tract 088105
Census Tract 310201	Census Tract 101300
Census Tract 460301	Census Tract 930200
Census Tract 800504	Census Tract 670324
Census Tract 800101	Census Tract 088105
Census Tract 621204	Census Tract 800504
Census Tract 800101	Census Tract 651302
Census Tract 800506	Census Tract 670326
Census Tract 301400	Census Tract 980020
Census Tract 800203	Census Tract 110003

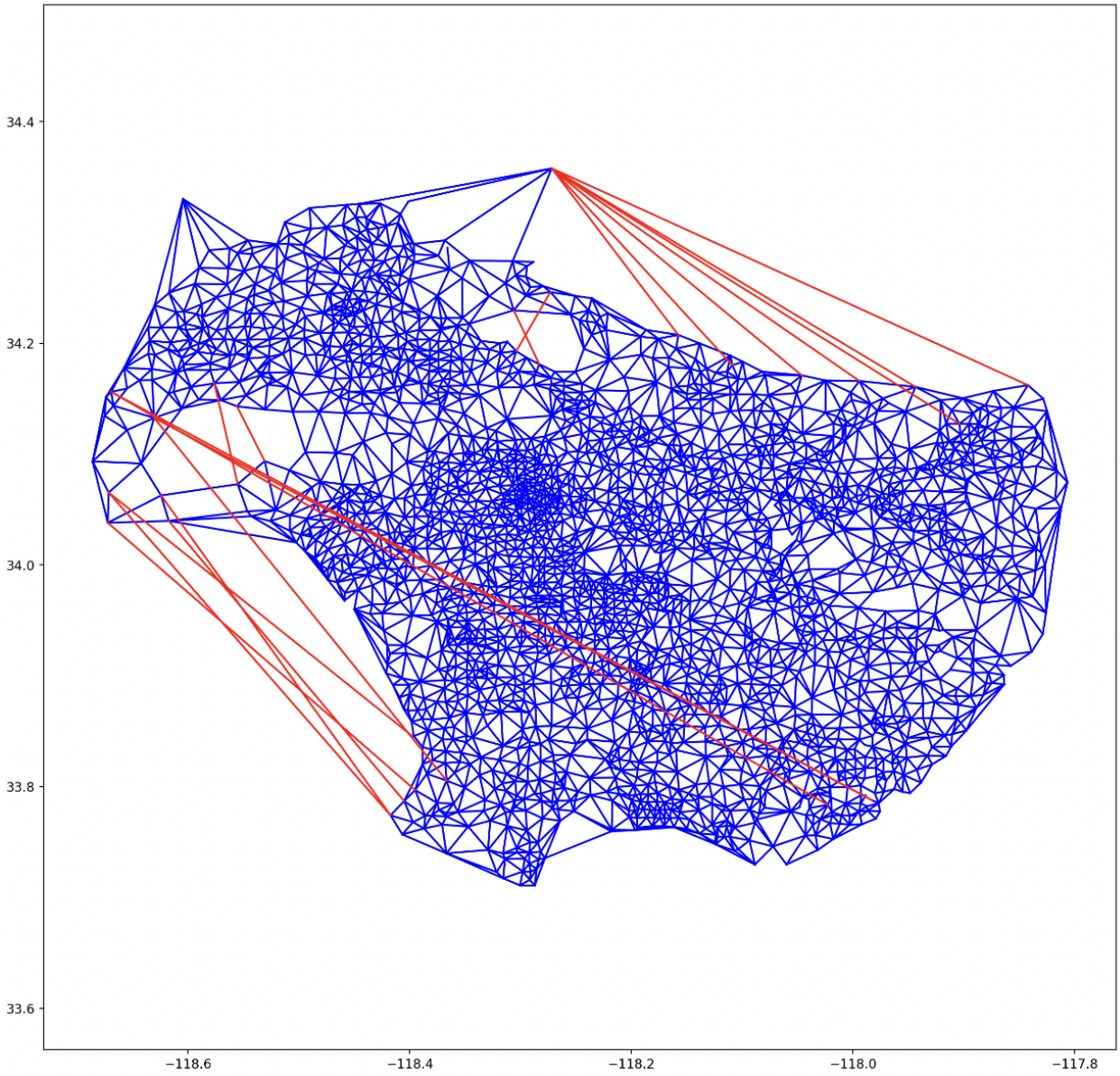


Figure 15: Resultant graph with Strategy 3 impact

Time Complexity of the strategy 3:

If n is number of vertices.
Time to find all pairs shortest path: $20 * n^3$
Time to find all pair euclidean distances: n^2
Computing extra distance and sorting them = $20(n + nlog(n))$
Total time : $20 * n^3 + n^2 + 20(n + nlog(n))$

3.4 Question 21(b)

Strategy 4: To find top 20 pairs of vertices which have the maximum extra travelling time and create a road between them:

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)

Note: To make the computations fast, we assumed that finding the shortest path between two vertices using travel times as edge weights and shortest path with geo distance would lead to same path.

The source and destination of the new roads found using Strategy 4 are:

Source	Destination
Census Tract 670328	Census Tract 800504
Census Tract 670416	Census Tract 800504
Census Tract 670328	Census Tract 800202
Census Tract 670413	Census Tract 800504
Census Tract 670416	Census Tract 800202
Census Tract 800504	Census Tract 670328
Census Tract 800504	Census Tract 670416
Census Tract 137402	Census Tract 980019
Census Tract 137504	Census Tract 262601
Census Tract 670326	Census Tract 800504
Census Tract 138000	Census Tract 262601
Census Tract 800101	Census Tract 670328
Census Tract 800101	Census Tract 670416
Census Tract 800203	Census Tract 670328
Census Tract 134001	Census Tract 262601
Census Tract 138000	Census Tract 980019
Census Tract 670406	Census Tract 800504
Census Tract 800203	Census Tract 670416
Census Tract 134002	Census Tract 262601
Census Tract 137401	Census Tract 980019

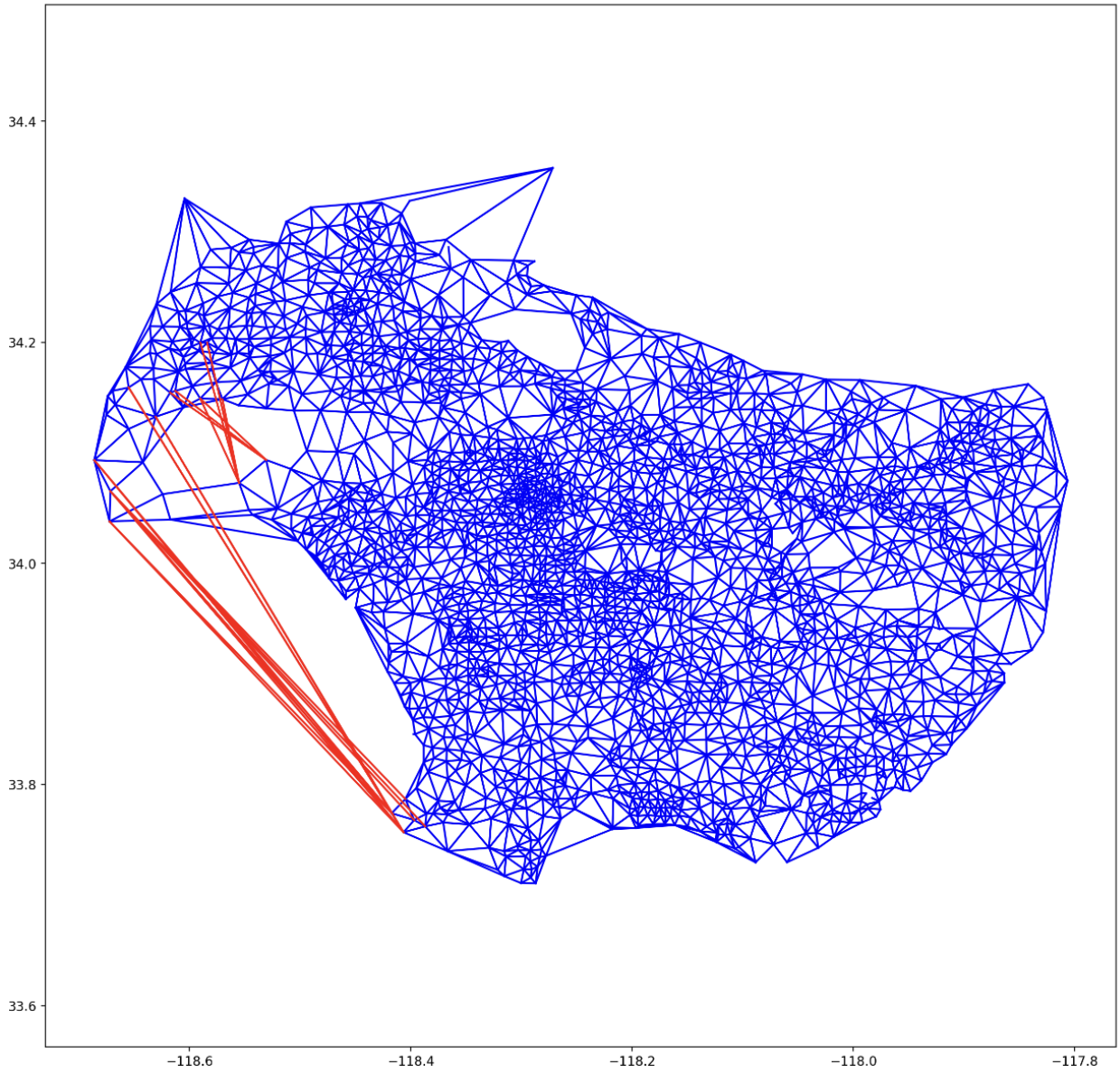


Figure 16: Resultant graph with Strategy 4 impact

Time Complexity of the strategy 4:

If n is number of vertices with 20 new roads that need to be added.

Time to find all pairs shortest path: $2 * n^3$ (Once with travel time as edge weights and once with geo distance as edge weights)

Time to compute travel speeds: n^2

Time to find all pair euclidean distances: n^2

Computing extra travelling time and sorting them = $n + nlog(n)$

Total time : $2 * n^3 + 2 * n^2 + n + nlog(n)$

3.5 Question 22

Strategy 5: To find top 20 pairs of vertices which maximise the extra travelling time defined in strategy 4 with dynamic updates to graph and varying resulting metrics with each graph update.

The source and destination of the new roads found using Strategy 5 are:

Source	Destination
Census Tract 670328	Census Tract 800504
Census Tract 137402	Census Tract 980019
Census Tract 137504	Census Tract 262601
Census Tract 400501	Census Tract 930200
Census Tract 670328	Census Tract 800506
Census Tract 408005	Census Tract 930200
Census Tract 404301	Census Tract 930200
Census Tract 430003	Census Tract 930200
Census Tract 500100	Census Tract 930200
Census Tract 501600	Census Tract 930200
Census Tract 430301	Census Tract 930200
Census Tract 621204	Census Tract 800504
Census Tract 139703	Census Tract 980019
Census Tract 433401	Census Tract 930200
Census Tract 430501	Census Tract 930200
Census Tract 800506	Census Tract 670326
Census Tract 310100	Census Tract 101210
Census Tract 461200	Census Tract 930200
Census Tract 310201	Census Tract 980020
Census Tract 800101	Census Tract 621326

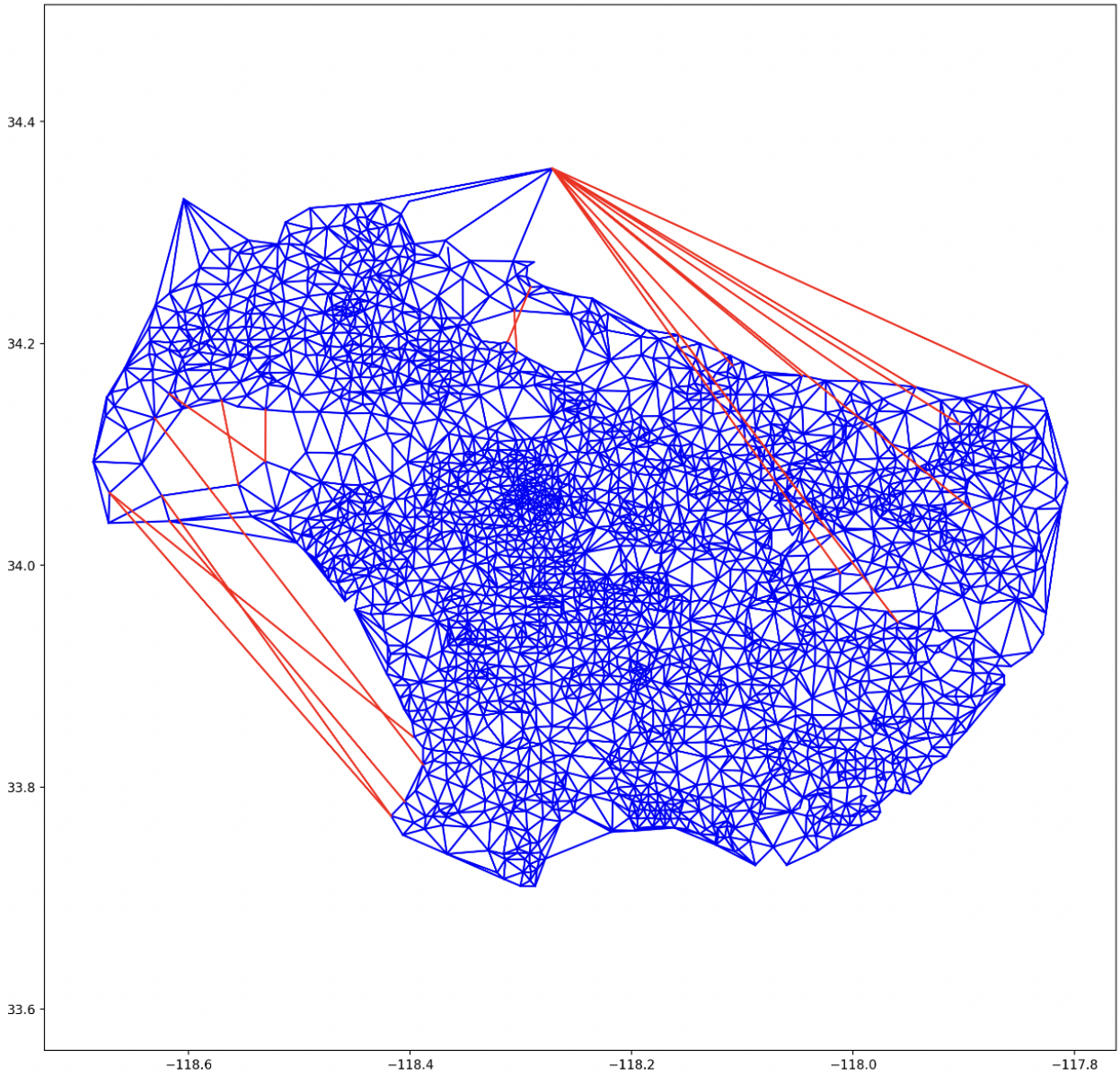


Figure 17: Resultant graph with Strategy 5 impact

Time Complexity of the strategy 5:

If n is number of vertices with 20 new roads that need to be added.

Time to find all pairs shortest path: $20 * 2 * n^3$ (Once with travel time as edge weights and once with geo distance as edge weights)

Time to compute travel speeds: $20 * n^2$

Time to find all pair euclidean distances: n^2

Computing extra travelling time and sorting them = $20 * (n + nlog(n))$

Total time : $20 * 2 * n^3 + 21 * n^2 + 20 * (n + nlog(n))$

3.6 Question 23

(a)

In strategy 1 we selected the edges based on extra distance which is the difference between shortest path distance and the euclidean distance between the pair of vertices. This is a naive approach and we can see that it selected most of the vertices which lie along the two concavities in the graph as these vertices are difficult to reach from other well connected vertices but at the same time their euclidean distances with other vertices on the concavities. Strategy 2 is a slight modification which uses more data to make this decision. We weighted the extra edge distances from strategy 1 with simulated frequency of usage of the edges, since the more visited nodes must be given preferences for new road construction. Some new vertices crop up with this strategy as compared to the Strategy 1. The time complexity of both the strategies is same but Strategy 2 is better as compared to Strategy 1 since we are using more useful data to derive the results.

(b)

Strategy 3 is similar to Strategy 1 in the sense that it's optimisation task is the same but Strategy 3 is updating the graph on the fly with each new road found to be constructed. Since the graph is updated after new roads are added, the graph metrics (shortest paths and extra distance) will change and thus we are incorporating the effect of new changes while making the next decisions which removes redundancy in selecting the same vertices as we can see in the results. Strategy 3 is much more diversified in terms of selecting nodes.The time complexity of Strategy 3 quite high as compared to Strategy 1 which is approx. 20 times to that of strategy 1 but clearly it's much better than Strategy 1 which doesn't incorporate the new changes with each successive update of the network.

(c)

Strategy 4 is finding new roads to construct by optimising the travel times between the nodes as compared to Strategy 1 which is optimising the extra distance. Both of the strategies are static in nature. The time complexities of both the approaches are quite similar but slightly higher for Strategy 4 since it runs Floyd-Warshall algorithm twice. Also while implementing strategy 4, to make the computations fast, we assumed that finding the shortest path between two vertices using travel times as edge weights and shortest path with geo distance would lead to the same path which might not always be true and Strategy 4 would become much more computationally expensive since then we need to run Djiktsra's algorithm to find the shortest path nodes and then we need to accumulate the travel times along this path. The complexity of such an approach of the order of $O(n^4)$ and is much more complex. Strategy 4 is a better strategy than 1 because it takes into account the traffic constraints inherently which is not there for strategy 1. Also we can see that the road lengths are small for strategy 4 vs strategy 1 and thus is much more cost efficient.

(d)

If we want to improve the overall road network by constructing 20 new roads so as to reduce the travelling distances, the dynamic method would be a better strategy to do so. The problem with static strategy is that, we are running optimisation task once and adding the edges in bulk. But this is sub-optimal way given graph properties change with each new edge being added and thus will lead to a new optimisation task at every iteration. Thus overall to minimise the travelling distances given 20 new roads need to be constructed the dynamic approach is better as compared to static approach.

But both of these are not optimal strategies because in both of the methods we are just focused on finding 20 edges which are bad as per given strategy and constructing an edge but this doesn't necessarily mean that we are going to improve the overall road network i.e. reducing the overall travelling distances. So the task can be framed as below optimisation problem.

$$C = \sum_{\forall (v,s) \in V(G) X V(G)} \text{Shortest Path Distance}(v, s) \tag{1}$$

where V(G) is the set of vertices in graph G

Goal: min C by iteratively adding 20 new edges between any two vertices in Graph.
Now the above problem is NP-complete, since when adding any edge, the graph is changed dynamically. The

problem can be solved with a brute force approach. Time complexity for this approach would be: At each step evaluate all the possible edges that can be added and move forward with that for (N - 1) remaining steps. Total Steps = 20. Every possibility will lead to (n * n) new graphs with one new edge. Thus total time complexity is,

$$T = \sum_{i=1}^{20} (n^2 - E(G))^i, \text{n is number of nodes in G, E(G) = no. of edges in G} \quad (2)$$

(e) New strategy which we have come up with revolves around the fact that roads should be such that they connect more nodes together. We saw in the earlier strategies that the new roads were along concavities and thus while travelling on that road one can't easily reach other nodes. In this new strategy Strategy 3 is modified that is we are still optimising the extra distance while updating the graph in each iteration, but the constraint is that the new roads should be able to capture atleast 20 nodes in it's vicinity. That is given a new road can be modelled by straight line: Ax + By + C (where x and y are longitudes and latitudes respectively), we need to choose top 20 roads which have atleast 20 nodes having perpendicular distance from this road as less than 5 miles and that the road itself should be chosen from the Strategy 3 optimisation.

The source and destination of the new roads found using custom strategy are:

Source	Destination
Census Tract 800504	Census Tract 087200
Census Tract 800504	Census Tract 110606
Census Tract 800202	Census Tract 001503
Census Tract 800102	Census Tract 001707
Census Tract 554511	Census Tract 800504
Census Tract 800203	Census Tract 021815
Census Tract 106111	Census Tract 400501
Census Tract 800102	Census Tract 503702
Census Tract 534102	Census Tract 800102
Census Tract 533601	Census Tract 800102
Census Tract 554700	Census Tract 800504
Census Tract 552100	Census Tract 800504
Census Tract 800202	Census Tract 532400
Census Tract 800203	Census Tract 021814
Census Tract 137103	Census Tract 021815
Census Tract 800102	Census Tract 532400
Census Tract 553200	Census Tract 800504
Census Tract 800202	Census Tract 269100
Census Tract 800102	Census Tract 011300
Census Tract 553504	Census Tract 800504

The time complexity for this Strategy is quite high as compared to above implemented strategies but the results are much more ideal for the case it creates much more connected roads and the issue with unreal roads along concavity is resolved.

Time Complexity of the new strategy:

If n is number of vertices with 20 new roads that need to be added.

Time to find all pairs shortest path: $20 * 2 * n^3$ (Once with travel time as edge weights and once with geo distance as edge weights)

Time to find all pair euclidean distances: n^2 Time to find an edge from sorted extra distance optimised edges which satisfy the 20 node vicinity constraint: $n^2 = K$

Computing extra travelling distance = $20 * (K + nlog(n))$

Total time : $20 * 2 * n^3 + n^2 + 20 * (n^2 + nlog(n))$

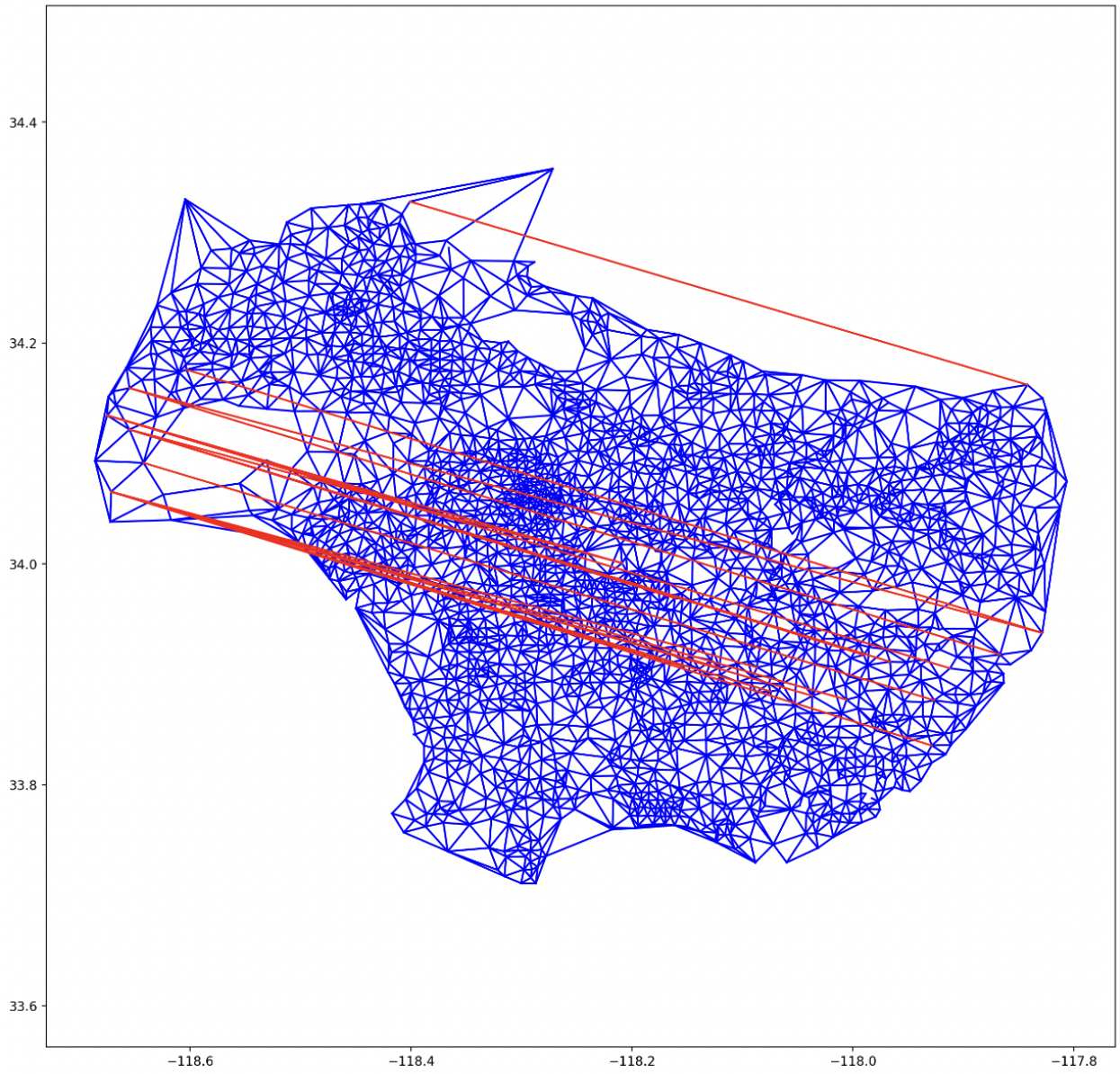


Figure 18: Resultant graph with New strategy

4 Our own Task

In this task, we decided to explore the Travelling salesman problem more. For the start we started with Travelling Santa problem and tried to estimate how much of speed Santa would require to complete the trajectory of LA to deliver the gifts and what is the total distance that he needs to cover on Christmas night. For the second part we explored some of the TSP solving algorithms, studied them and evaluated them on the basis of their runtimes on a custom dataset.

Finding some facts about Santa's journey on Christmas night:

We computed the total distance that Santa needs to travel in LA on Christmas night from 12AM - 6AM i.e 6 hours to deliver gifts. The trajectory of TSP is shown below which is computed in one of the earlier questions. We assumed that on average Santa needs to deliver around 1 chosen gift in one given coordinate

and that to deliver one gift (i.e.) to climb down the chimney and come out, Santa takes 5 seconds.

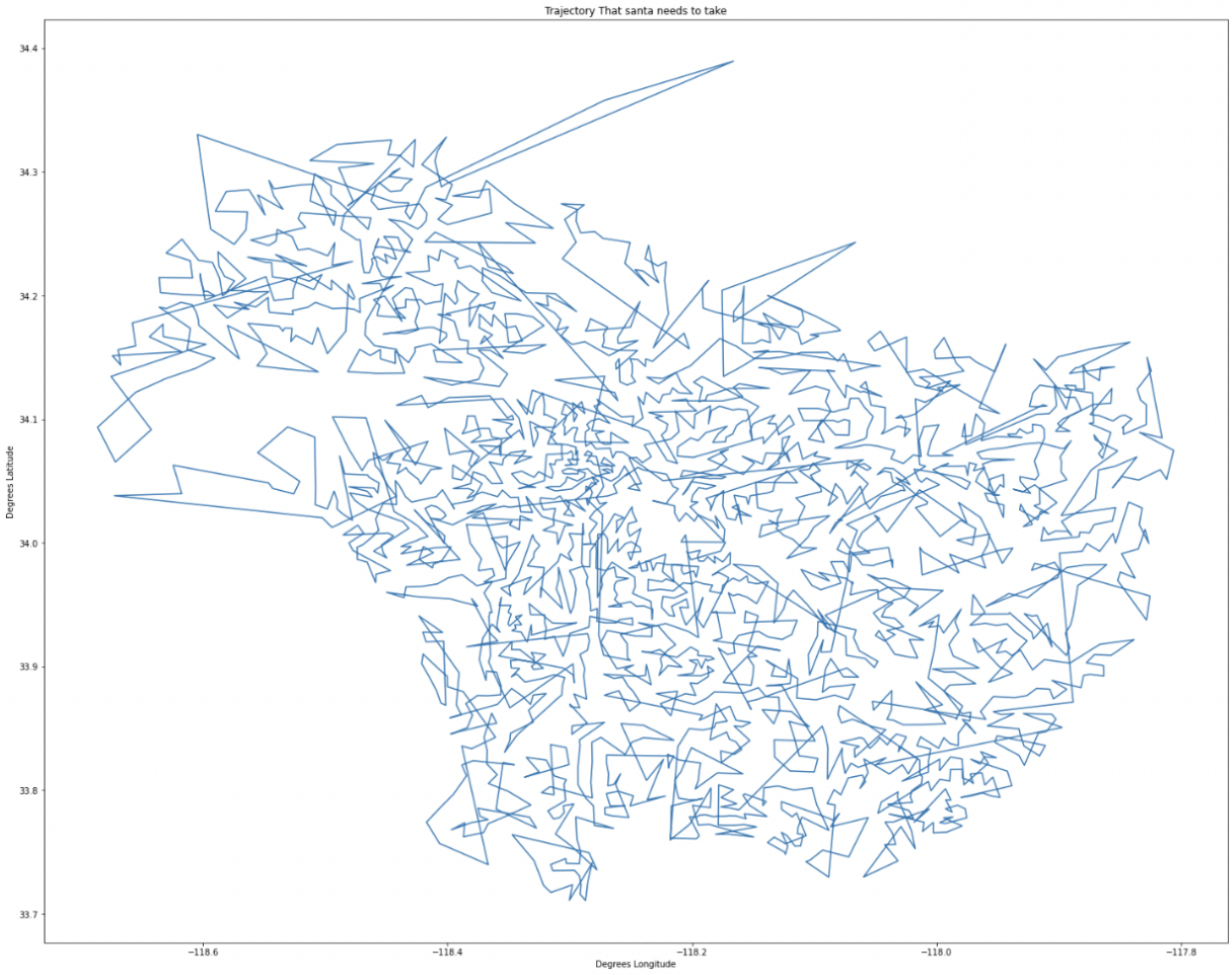


Figure 19: Santa's Trajectory

The total distance that Santa needs to travel along above trajectory is given by: **3963.955** Km. This distance is computed using **Haversine** formula since we know the latitudes and longitudes of the places that Santa needs to visit in LA.

We derived the formula for the speed with which Santa needs to travel which is shown below:

$$Speed(Km/hr) = \frac{\text{Total Distance}}{6 - n * t}, \text{t is time to deliver gift, n is number of nodes} \tag{3}$$

We derived the speed as: **1709.0107** Km/ hr

Comparison of different TSP Algorithms:

We evaluated 4 approaches to solve the Travelling Salesman problem. 1). Brute Force Approach 2). Naive Dynamic Programming Approach. 3). Greedy Algorithm. 4). 2-Opt Swap approach.

The dataset used is a fully connected Graph dataset, where the nodes (734 in number) denote the (x, y) coordinates of each city.

1). **Brute Force Approach**:

This algorithm searches every permutation of cycles and returns the shortest one. It's a simple algorithm that guarantees the optimal solution, but has **O(N!)** where N is the number of nodes in Graph, and is therefore not feasible to run on a large dataset. We ran this algorithm on a derived subgraph of (2 - 10) number of nodes and the runtimes are shown below. The TSP costs for this approach are optimal as compared to other algorithms.

2). **Dynamic Programming Approach**:
We assume that the initial travelling cost is equal to 0. Next, the TSP distance value is calculated based on a recursive function. If the number of cities in the subset is two, then the recursive function returns their distance as a base case. On the other hand, if the number of cities is greater than 2, then we'll calculate the distance from the current city to the nearest city, and the minimum distance among the remaining cities is calculated recursively. Finally, the algorithm returns the minimum distance as a TSP solution.

$$C(S, i) = minC(S - \{i\}, j) + distance(j, i) \tag{4}$$

where j belongs to S, j != i and j != 1.
C(S, i) be the cost of the minimum cost path visiting each city in set S exactly once, starting at city 0 and ending at city i.

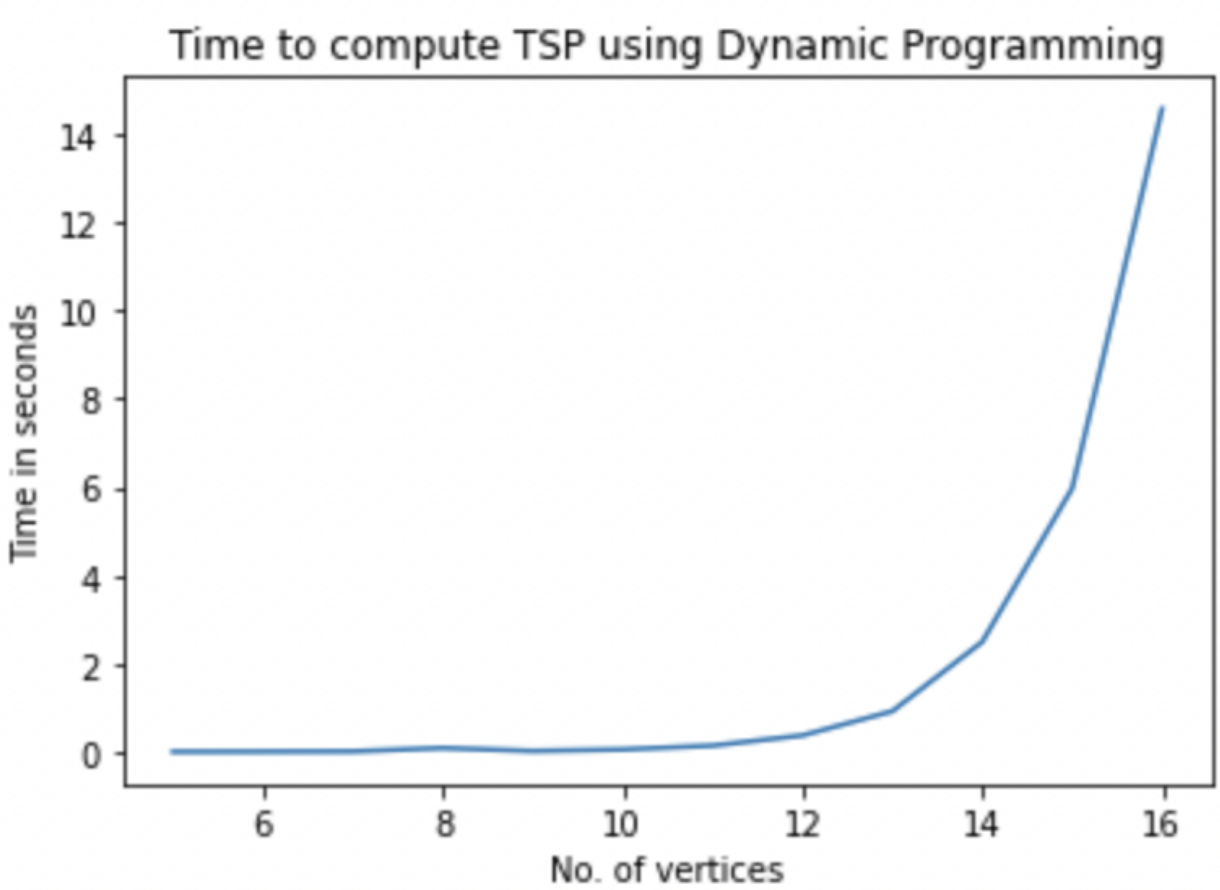


Figure 20: Dynamic Programming Method Runtimes

We were able to run this algorithm for 16-17 nodes in a reasonable amount of time and the results are shown above. Also this is an optimal strategy to find the min. TSP cost but the runtimes are very high and is not feasible for large networks.
Time complexity of dynamic programming approach is given as: $O(N^2 * 2^N)$

2). **Greedy Approach**:
The above two algorithms : DP approach and Brute force approach give optimal results but they are not feasible for big networks. Thus we use algorithms which give approximate estimate of the TSP cost, one of which is Greedy Algorithm. The algorithm is given as:
1). Start at an arbitrary city.
2). Go to the next closest unvisited city.
3). Repeat (2) until all cities have been visited, save path cost.
4). Start again at (1) with a different initial city.
5). Repeat (1-4) until all possibilities have been exhausted and return the shortest path.
The runtime for this approach is given as: $O(N^3)$.

We were able to run this algorithm on quite a high number of nodes 200 just to verify that it can be used for large networks but the TSP Cost is quite high for Greedy Method and thus the path found is very unrealistic. The runtimes for this algorithm are shown below.

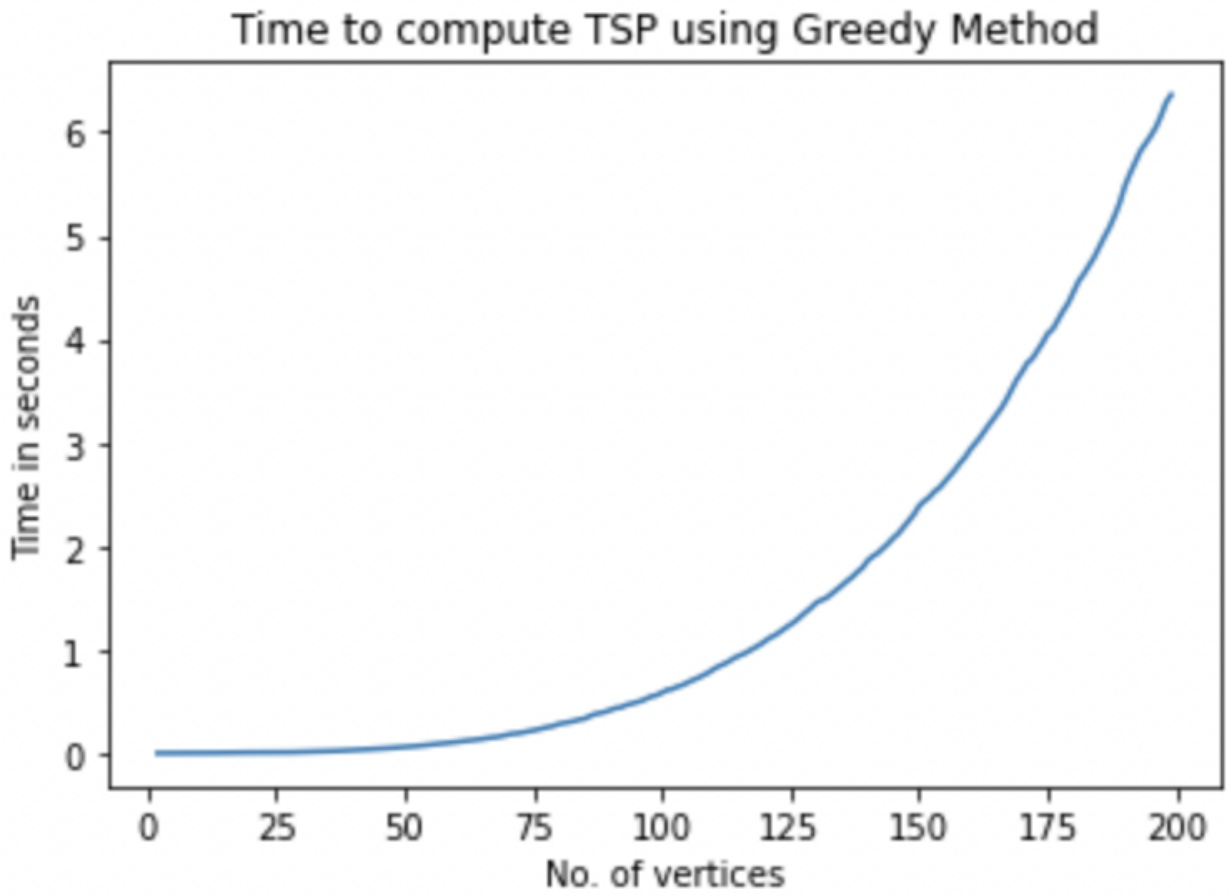


Figure 21: Greedy Method Runtimes

2). **2-Opt Swap Algorithm**:
We see that although Greedy Approach is fast its TSP cost is very high. We therefore evaluate one more algorithm called 2-Opt Swap which is both fast and results is low TSP cost when compared to greedy approach. The algorithm is given as:
1). Start with a random route.
2). Perform a swap between two edges.
3). Keep new route if it is shorter.

4). Repeat (2-3) for all possible swaps.
5). Repeat (1-5) for M possible initial configurations.
The time complexity of this approach is: $O((M * N)^2)$
This approach has a high runtime when compared to Greedy Approach but very less when compared with optimal methods and thus can give good TSP trajectories with low TSP cost in reasonable amount of time for big networks. The runtimes are shown below for this approach.

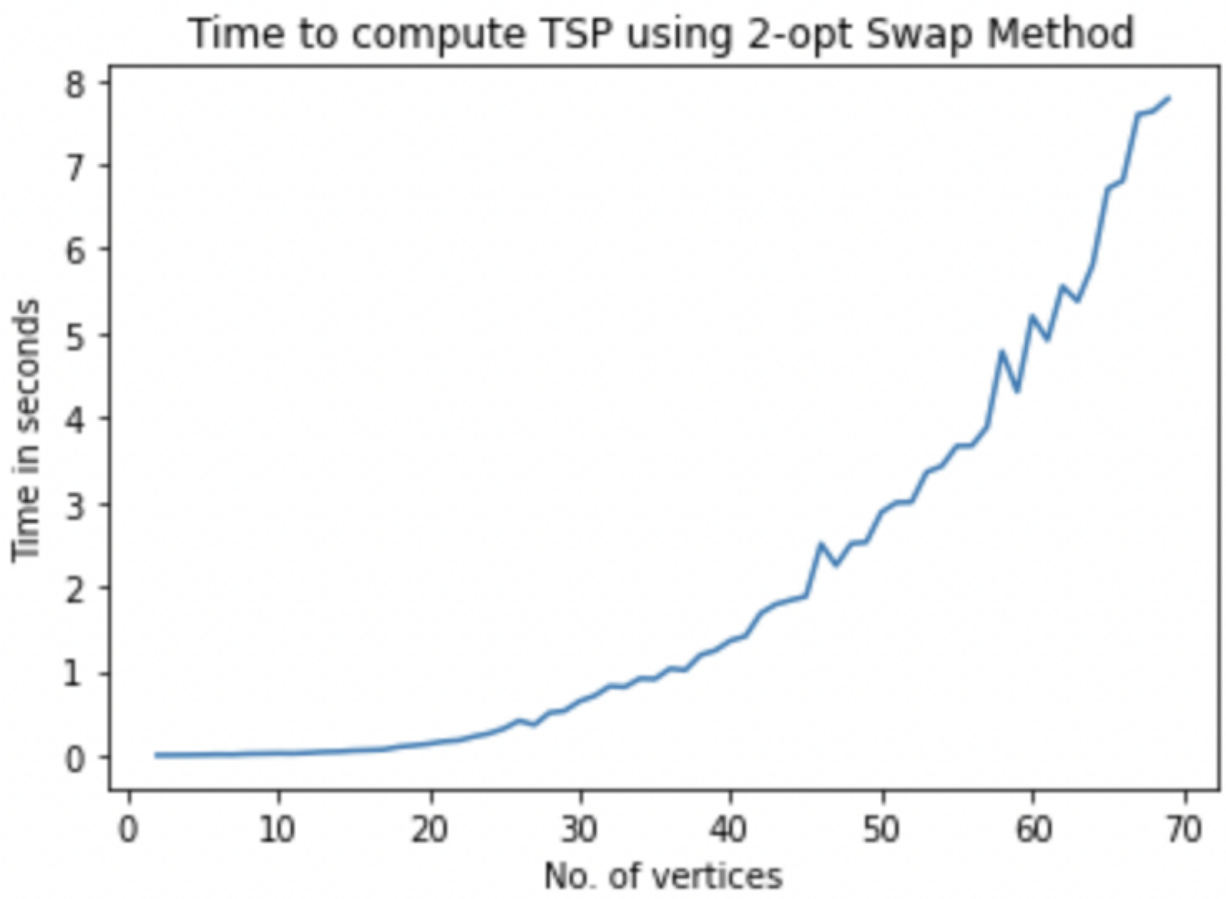


Figure 22: 2-Opt Swap Approach Runtimes

The intuition behind the algorithm is that swapping two edges at a time untangles routes that cross over itself.

A comparison graph for the above discussed approaches is shown below. There a lot more algorithms which give better estimates of TSP with low trajectory costs and we would like to study them in future. Some of the approaches are : Particle Swarm Optimisation approach, Nearest Neighbors Approach and recent advancements using Neural Networks.

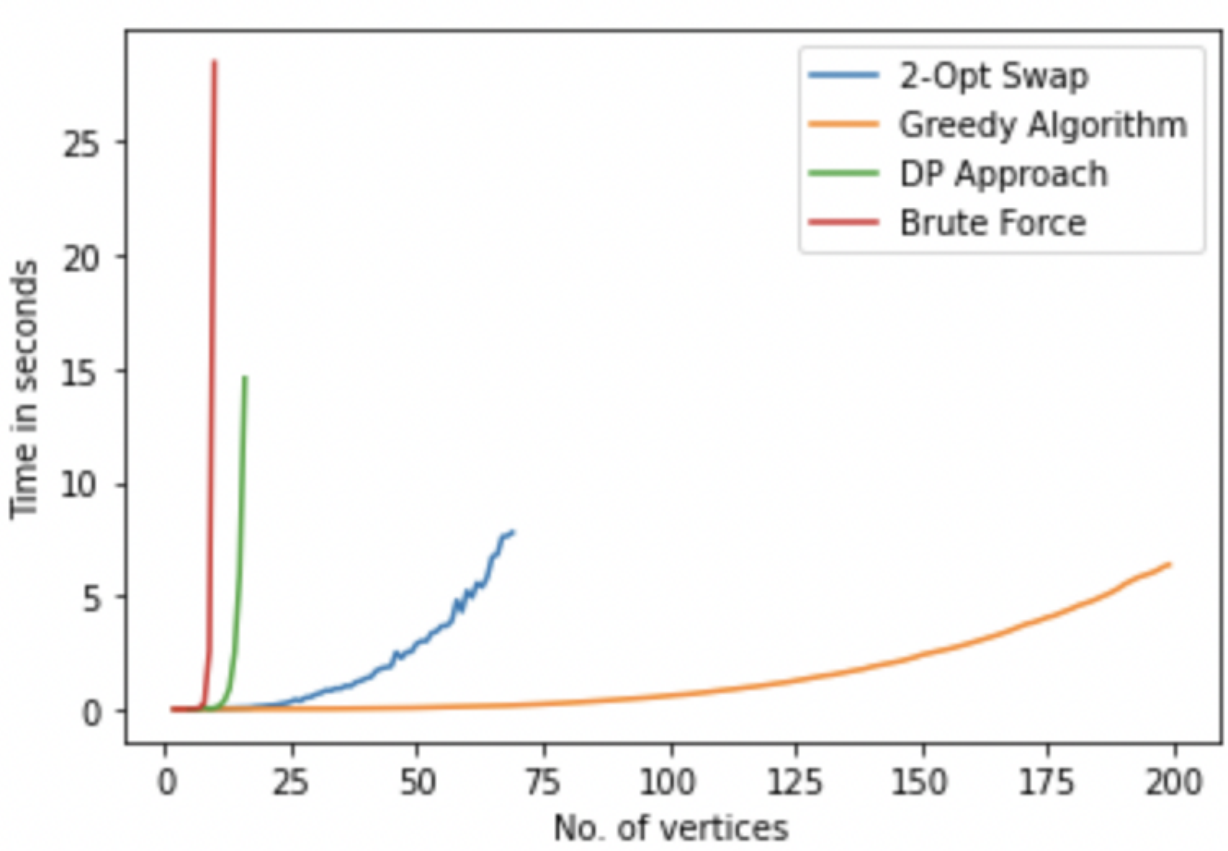


Figure 23: Comparison of runtimes