ECE232E: LARGE SCALE SOCIAL AND COMPLEX NETWORKS: DESIGN AND ALGORITHMS

Project 4: Graph Algorithms

June 13, 2020

Wanli Gao, UID: 105431975
Yifan Zhang, UID: 805354474
Tianyi Zhao, UID: 804380974

1 Stock Market

1.1 Return correlation

Question 1:
In theory, the upper bound and lower bound should be as below:
The upper bound of ρ_{ij} is : 1.0
The lower bound of ρ_{ij} is : -1.0
Actually in our group's practice, the upper bound and lower bound should be as below:
The upper bound of ρ_{ij} is : 1.0
The lower bound of ρ_{ij} is : -0.199
Justification:
The use of the log-normalized return can reduce the skewness of the data. Influence of very large values will be reduced using log-normalized return. Besides, using log-normalized return gives better additivity when calculating continuously compounding return.

1.2 Constructing correlation graphs

Question 2:
The histogram for the un-normalized distribution of edge weights is shown below:

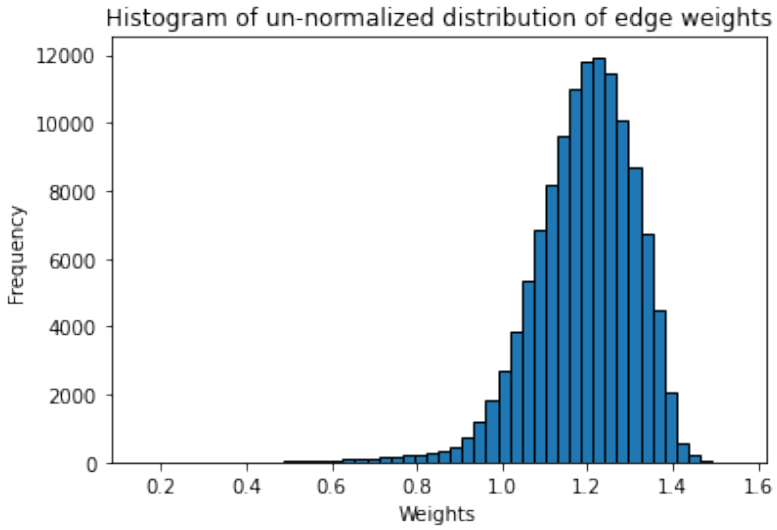

Figure 1.1. Histogram of un-normalized distribution of edge weights

1.3 Minimum spanning tree (MST)

Question 3:
We extract the MST of the correlation graph and categorize each stock into a sector. We also color-code the nodes based on sectors, the MST is shown as follows:

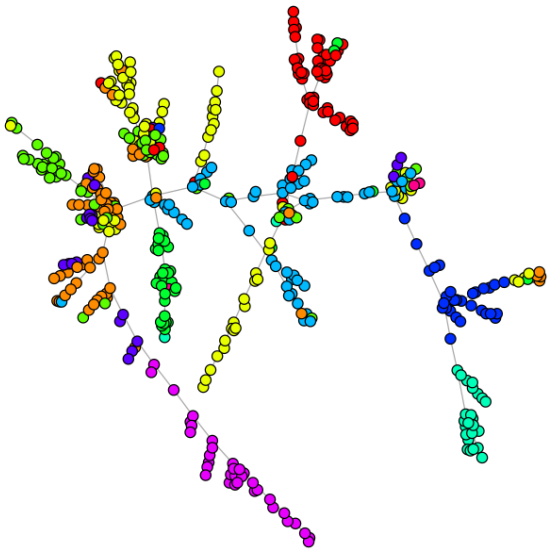

Figure 1.2. MST of the correlation graph

From the MST we can see that stocks with same sector are more likely to be connected together in the MST. This is because the MST clusters nodes with lower weights together. Nodes with the same sector tend to have higher correlation and thus lower weights, so in the MST they are closer to each other than nodes with different sectors.

1.4 Sector clustering in MST's

Question 4:
The value of α for case 1: 0.829
The value of α for case 2: 0.134
Difference:
In case 1, we can know that the probability of a particular node with a particular sector depends on Q_i (the set of neighbors of node i). Since the MST is constructed in a way that the neighbors are the stocks with high correlation values so that they have the high probability to be the same sector.
In case 2, the probability equals to nodes in the same sector divided by all the nodes, which contains all the nodes in the graph into consideration.
As the result of above, the α in case 1 is larger than α in case 2.

1.5 Correlation graphs for weekly data

Question 5:
We extract the MST from the correlation graph based on weekly data. We sample the stock data weekly

on Mondays, use the sampled data to create the graph and plot the MST of the graph.

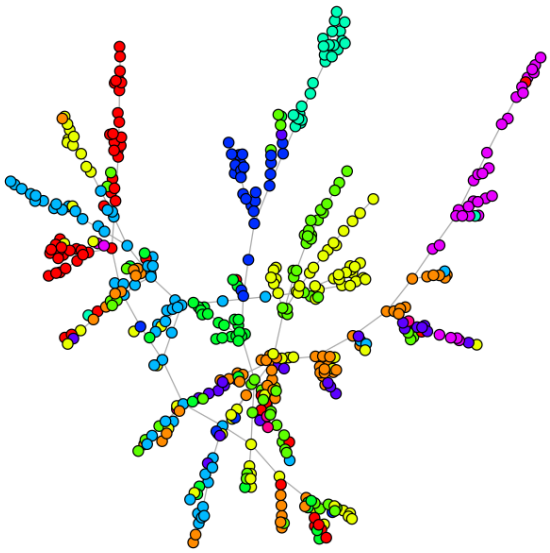

Figure 1.3. MST of the correlation graph for weekly data

Still, from the plot we can see that stocks with same sector are more likely to be connected together in the MST for weekly data. However, compared to the MST in Question 3, the clustering of same sectors is not so obvious. Stocks tend to have less connectivity with stocks with same sectors. This can be explained by the fact that stocks with different sectors may share more similarities in the long term.

2 Let's Help Santa!

2.1 Download the Data

We download the data "Travel Times by Month (All Days), 2019 Quarter 4", for Los Angeles area, provided by "Uber Movement". We use only data in December.

2.2 Build Your Graph

Question 6:
We build the graph using the given data and clean the graph. The number of nodes and edges in G is

2649 and 1003858, respectively.

2.3 Traveling Salesman Problem

Question 7:
We build a minimum spanning tree (MST) of graph G. Then we randonly pick 10 edges of the MST and report the physical distance of the two endpoints of these edges:

Table 1. Physical distance of the two endpoints of 10 randomly picked edges

Start point	End point	Physical distance
254	455	0.527
285	1801	0.513
1534	1538	0.77
1294	1295	0.26
1812	2275	0.69
471	2232	2.07
2293	2303	2.28
1940	1941	0.46
1926	1928	0.88
111	112	0.50

From the results we can see that the physical distance is quite small for all 10 edges. This matches our intuition, as MST connects nodes in the graph with lowest weight. Therefore the nodes in MST are likely to be in the same adjacent zone and the endpoints of an edge tend to be close to each other.

Question 8:
The percentage of triangles in the graph (sets of 3 points on the map) that satisfies the triangle inequality with 1000 random samples is 87.7%.

Question 9:
We apply the 1-approximate algorithm to solve the traveling salesman problem (TSP) problem on G. In the algorithm, we find the minimum spanning tree T, create a multigraph G' by using two copies of each edge of T and find an Eulerian walk of G; and an embedded tour.
The 1-approximate TSP cost is 455586.89, and the optimal TSP cost is the sum of the MST weight, which is 269084.62. The upper bound on the empirical performance of the approximate algorithm: 1.693

Question 10:
The trajectory that Santa has to travel is shown as follows:

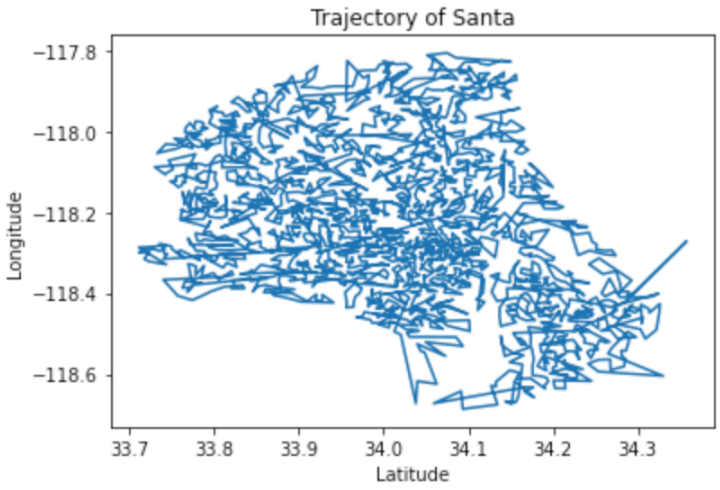

Figure 2.1. The trajectory that Santa has to travel

2.4 Analysing Traffic Flow

We analyse the maximum traffic flow between Malibu and Long Beach in this part.

2.5 Estimate the Roads

Question 11:
The plot of road mesh that we obtained using Delaunay triangulation algorithm is shown as follow:

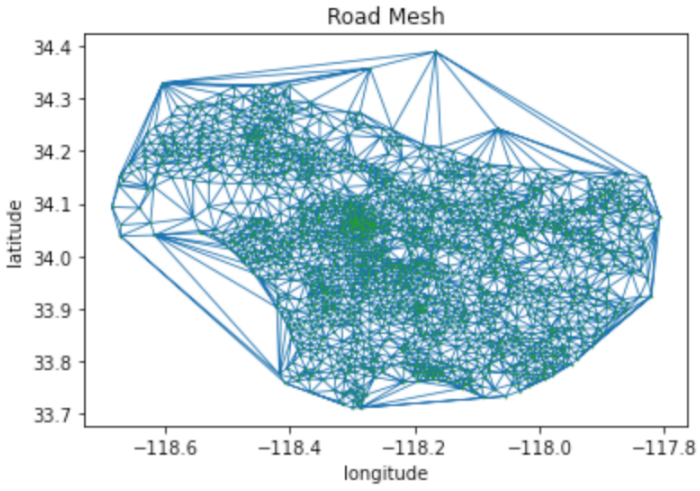

Figure 2.2. Road mesh obtained using Delaunay traingulation algorithm

Delaunay triangulation algorithm finds the triangles which contains no vertex in the circumcircle inside in the given graph. From the plot we can see that the triangles are plotted and the condition of the derived triangles is satisfied. In the derived graph the nodes are highly connected by triangles genereted by the

algorithm, although there are some extremely long edge which has no real meaning.
We create a graph G_Δ whose nodes are different locations and its edges are produced by triangulation. The graph is shown as follow:

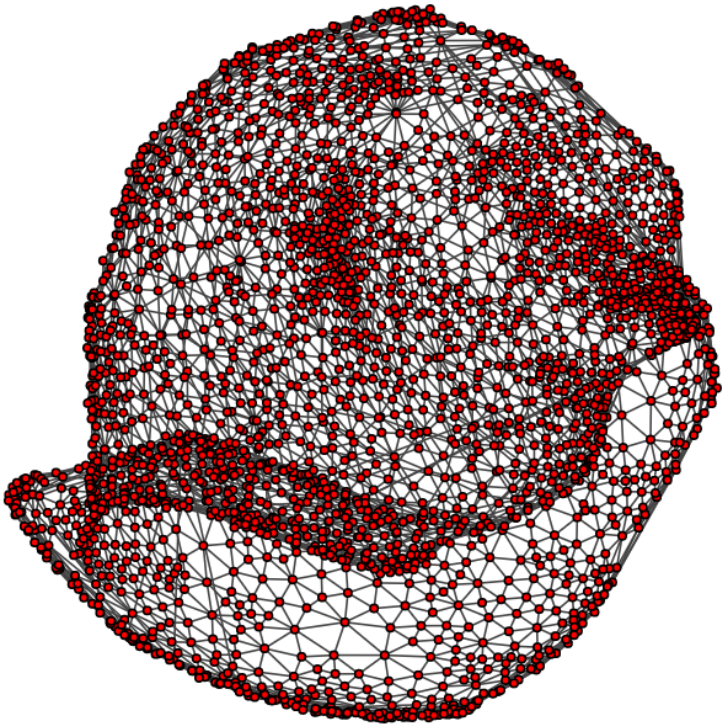

Figure 2.3. G_Δ

2.6 Calculate Road Traffic Flows

Question 12:
We consider the length that a single car takes to be the length of the car and the distance between two cars, which can be expressed by:

$$d = 0.003 + 2v,$$

where v is the mean speed of car (miles/second), which can be calculated by distance of each edge divided by the mean travel time of that edge.
The capacity of a given edge can be written as:

$$maxflow = \frac{3600v}{0.003 + 2v} \times 2$$

Multiply by 2 means there are two lanes in each direction.

2.7 Calculate Max Flow

Question 13:
Id of Malibu: 1711

Id of Long Beach: 675
Using the equation derived in Question 12, we can compute the maximum number of cars that can commute per hour from Malibu to Long Beach is 15611.6.
The number of edge-disjoint paths between the two spots is 6. The number of edge-disjoint paths matches what we see on our road map. As we can see from the zoom in graph, there are 6 roads that lead out from Malibu and 6 roads that lead in to Long Beach. Therefore the maximum number of edge-disjoint path is 6. Consider the high-connectivity of the graph, it makes sense to have 6 edge-disjoint paths between the two spots.

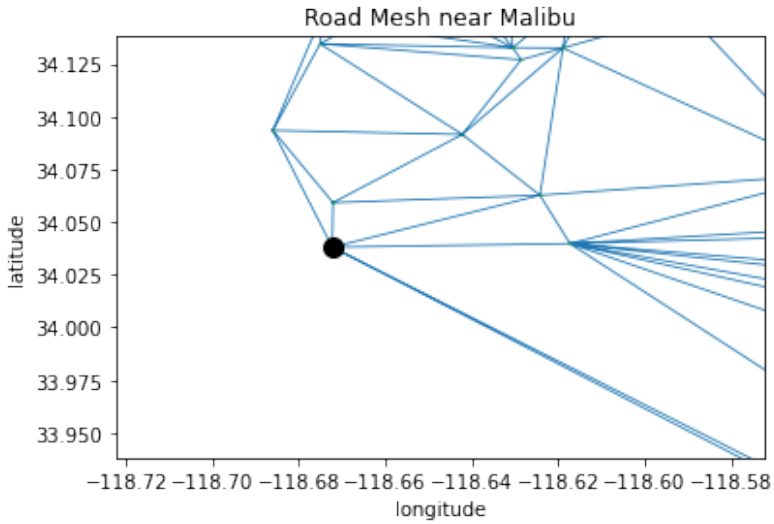

Figure 2.4. Road mesh near Malibu

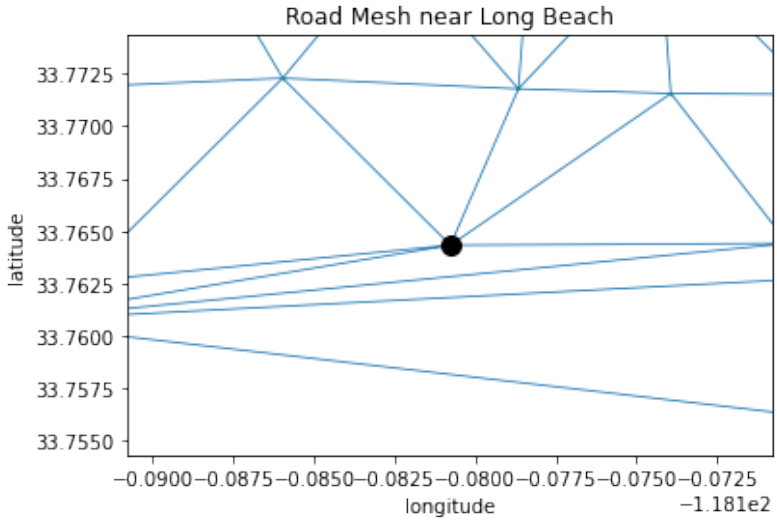

Figure 2.5. Road mesh near Long Beach

2.8 Prune Your Graph

Question 14:
To remove the unreal roads in G_Δ, we apply a threshold 12min (720s) to the original graph. Edges with

travel time over this number are removed. The resulting graph $\tilde{G}_\Delta$ is shown as follow:

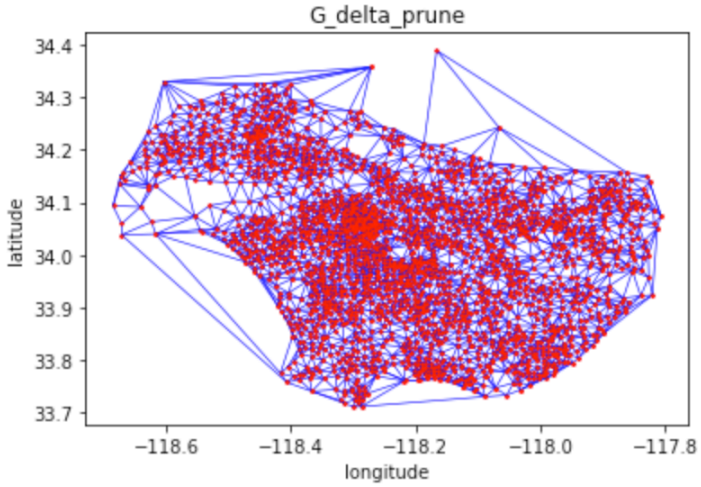

Figure 2.6. $\tilde{G}_\Delta$

From the prune graph we can see that some edges are removed from the original graph, for example, some edges near the beach and hills are disappeared. This means the threshold method works. However, it may also remove some real roads, especially when a low threshold is applied. Therefore we use 720s as our threshold to avoid removing too much real roads as well as removing unreal ones.

Question 15:
Id of Malibu: 1711
Id of Long Beach: 675
The maximum number of cars that can commute per hour from Malibu to Long Beach in the pruned graph is 11368.4. The number of edge-disjoint paths between the two spots is 4. This number is smaller than the number of edge-disjoint paths in G_Δ. This is because edges with travel time over 720s are removed from the original graph. From the zoom in graph we can clearly see that the number of roads that lead out from Malibu reduces to 4 while the number of roads lead in to Long Beach is still 6. Therefore the number of edge-disjoint paths also reduces to 4.

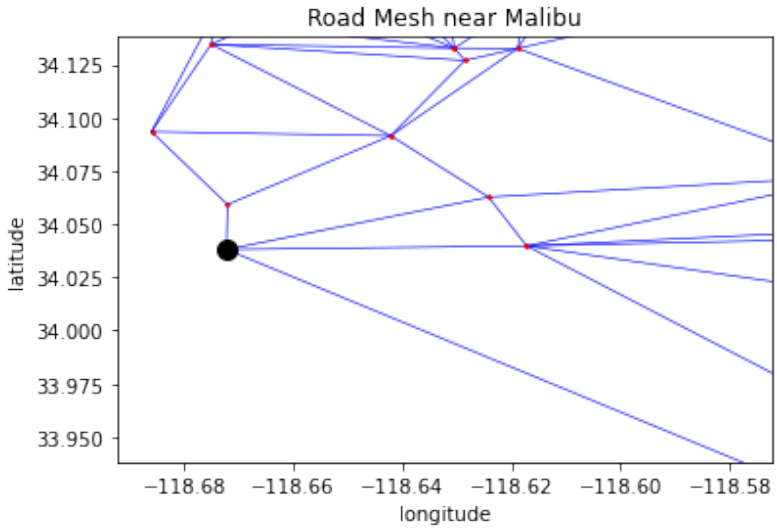

Figure 2.7. Road mesh near Malibu in $\tilde{G}_\Delta$

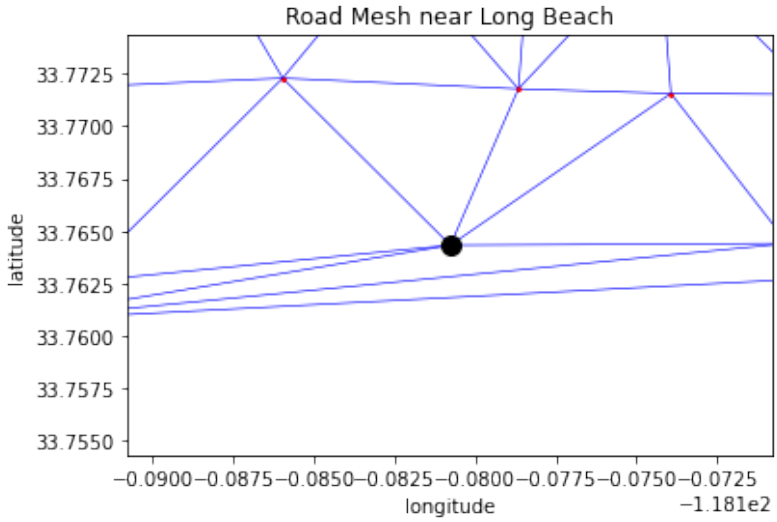

Figure 2.8. Road mesh near Long Beach in $\tilde{G}_\Delta$

2.9 Define Your Own Task

In this part, we take the Uber data again from December 2019 in Los Angeles, but generate a directed graph this time. Similarly, we will only keep the largest strongly-connected component for simplicity. The goal is to solve the TSP problem for the directed graph (Asymmetric TSP, or ATSP).
We first generate the directed graph G using the same data as before. The directed graph G has 2639 nodes, and 1690440 edges.
We researched and found an approximate algorithm for the ATSP probelm [1], which assumes a complete directed graph which satisfies the triangle inequality.
We then tested the triangle inequality of G and find that there are about 80% of 1000 randomly-sampled triangles satisfy the triangle inequality. Due to this high fraction, therefore, we could assume the triangle inequality holds for the whole graph, to solve the ATSP.
While our graph is not complete, since it's strongly connected, we could still adapt the above algorithm

as a starting point. Our full algorithm will be divided into two steps:
Step 1: Find a minimum-cost cycle cover for the input graph.
Step 2: Delete one random edge from each cycle, and then connect the cycles in a random order, with the shortest paths between each pair of connected cycles.
Step 1 is adpated from [1] and has been proved polynomial in time. Step 2 is our modified algorithm, which is also polynomial in time: the number of cycles is polynomial in the number of total nodes, the edge deletion and cycle order are both random and thus polynomial in the number of cycles. Finally, finding the shortest path is also polynomial in time. Therefore, our proposed algorithm as a whole should be a reasonable approximate of the optimal tour and should be polynomial in time complexity. Our resulted graph from the above algorithm has 2639 nodes together with 3048 edges, and the tour is plotted in Figure 2.9.

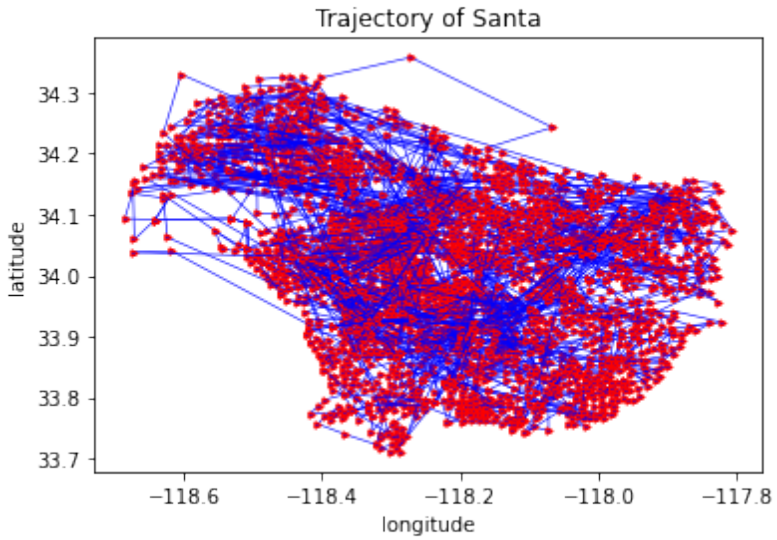

Figure 2.9. The trajectory that Santa has to travel

This ATSP algorithm yields a total cost of 1436792.5 for the input graph (with 2639 nodes and 1690440 edges). It only takes several seconds for the algorithm to run on a normal CPU, which again proved that its time complexity is polynomial. Besides, as mentioned above, the 1-approximate TSP cost is 455586.8 for the undirected graph. We could observe that the ATSP cost is roughly 3.15 times of the TSP approximate cost. This result is quite pleasant, given that the new graph is directed and we've made a lot assumptions and approximations.
If we could further explore on this task, we might need to design more efficient algorithms on step 2, to make a better approximation for the whole task.

3 References

[1] L. Chandran and L. Ram. "On the relationship between ATSP and the cycle cover problem". In: *Theoretical Computer Science* 370.16 (2007), pp. 218–228.