Names: Suleiman Sulaimanov, Demir Altay
Link to colab:
https://colab.research.google.com/drive/1Q8SORBoKJFMoL1ycVSpmbNvLwB1Fl62R?usp=sharing

Objectives:

- Gain a thorough understanding of how an agent can be assisted in navigating a stochastic environment, which simulates a real world ride sharing app.
- Build an efficient algorithm to service the most amount of passengers within the graph in the least possible time that is scalable.
- Within the algorithm handle how vans will pickup clients prioritizing the ones closest to them at any given moment.
- Dynamically path the van to dropoff the passengers in such a way that maximizes efficiency

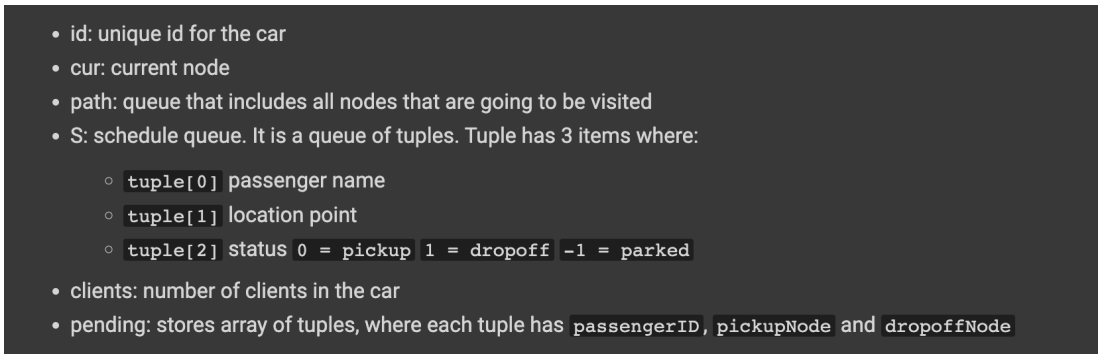
Assumptions

- First 4 minutes of the simulation cars will stay parked because R queue updates on 4th minute
- All vans are empty when the simulation starts
- In the required graph with the connectivity of 2, some nodes are not connected at all. We believe it is because in small cities cars cant go everywhere.
- With the 4-node average connectivity, a more advanced city is shown. Cars can get to the more nodes

Roles:

Suleiman Sulaimanov:

- Implemented schedule() function that adds pending orders to queue ***S***
- Implemented generateUID() function
- Implemented updatePath() function that adds a new path to the existing path
- Introduced a van object that has following structure:



- Implemented timer logic for simulation where path updated every 1 min while new orders come in every 4 mins
- Implemented algorithm for finding the closest available van to the recently generated passenger
- Implemented scheduling algorithms for cases where there are 1 and 2 pending passengers

- Implemented an algorithm that updates van locations. The algorithm also picks up and drops off passengers if the current node of the van is the same as the top element in the ***S***
- implemented generateVans() function that creates n number of vans

Demir Altay

- Was responsible for running all simulations
- Implemented scheduling algorithm for 3 passenger case
- Implemented generateOrders() function
- Implemented dist(a,b) function that returns distance between nodes a and b using the Dijkstra algorithm
- Was responsible for debugging

Approach

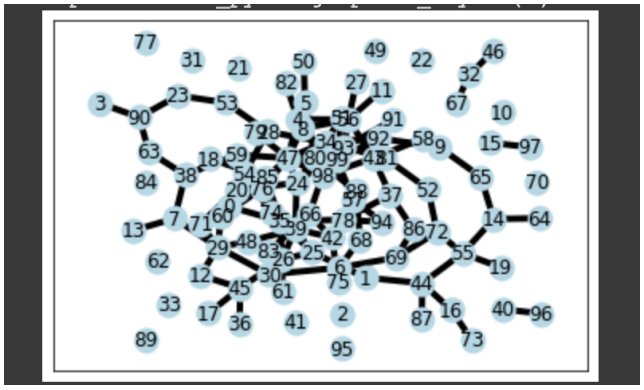
If min==4

- Generate requests R
- Iterate through requests
    - For every request iterate through vans and find the closest available van and add the request to his pending orders
- After finding the closest fans iterate through vans again
    - For every van take his pending orders and schedule pickups and dropoffs in most effective way

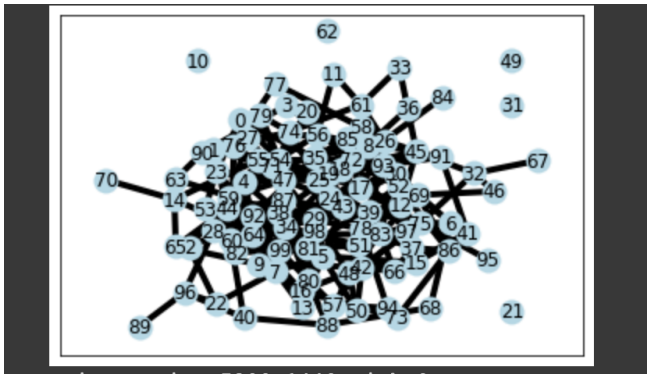
If min == 1

- Update the van's location every minute by updating the 'path' array in a dict
- If the current node of the van is the same as the top element in the S queue, we pop top element from S

Average connectivity of 2



Average connectivity of 4


62
10
49
11
33
77
61
84
3
20
36
31
0
79
74
56
58
85
82
26
45
91
67
90
76
55
54
35
72
93
30
32
70
63
23
4
47
25
98
17
52
69
46
14
53
44
92
38
87
24
43
39
12
6
41
28
60
64
34
29
98
78
83
97
75
652
82
99
81
5
51
37
86
95
9
7
80
48
42
66
15
96
16
22
13
57
50
94
73
68
21
89
40
88