SOLUTIONS

Our algorithm solves various problems.

Solution no 1:-

Finding best route>

a) Simple problem

Our algorithm gives us the best route which will take the shortest path out of all the available charging stations. (Fig: 1a)

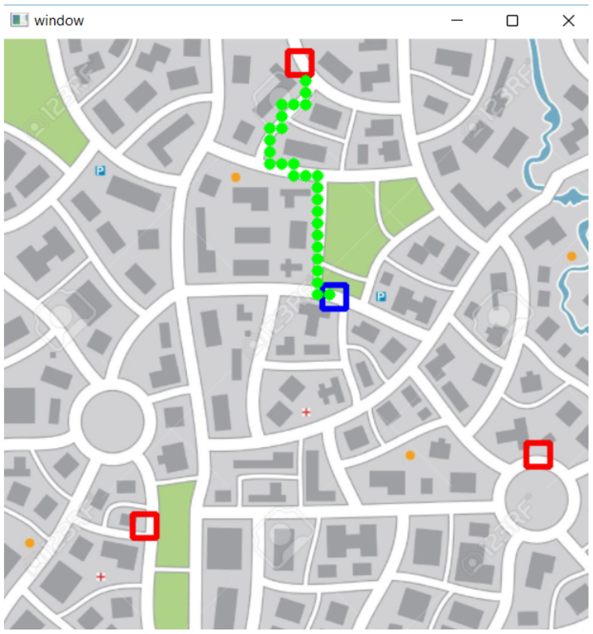

Fig: 1a

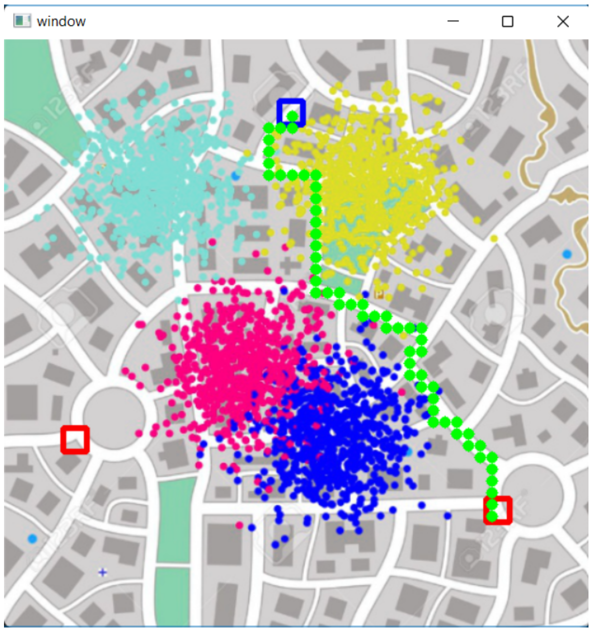

Fig: 1b

b) With traffic data.

The benefit of this grid approach instead of graph approach is that we can very easily put layers of information on top of this and our algorithm will work fine. Like we can add traffic information on top of it.(Fig: 1b)

c) With available battery data

We can also pass the information about available batteries and it will tell us if there is any charging station within our battery limit. (so here in below figure we have passed battery limit of 100 minutes and we see that in first case(Fig: 1c) distance is 112 minutes so it is not reachable while in second case(Fig: 1d) it is at distance of 44 minutes so reachable.

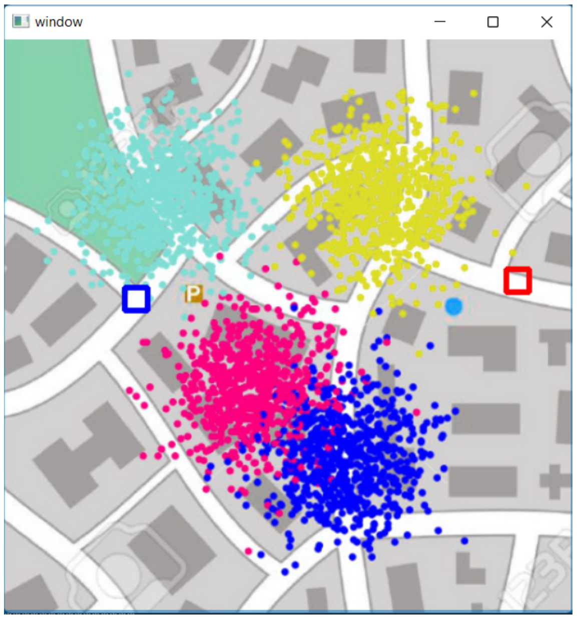

Fig: 1c

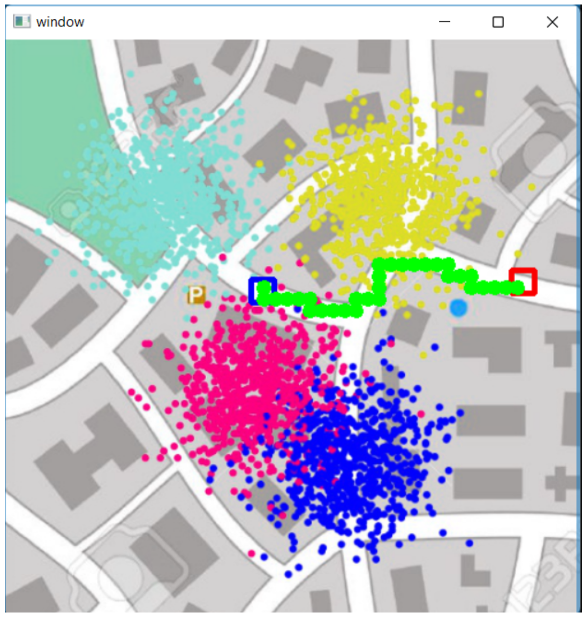

Fig: 1d

Solution no 2:-

Finding optimal Charging Stations location>

Finding the optimal location to set up a charging station is very tricky and we have to look at various factors, like where there is more demand and which is geographically the most feasible location from all places. Like for a circular city (Fig: 2a) (assuming the demand is uniformly distributed all over the city) then the optimal location would be the center of the circle.

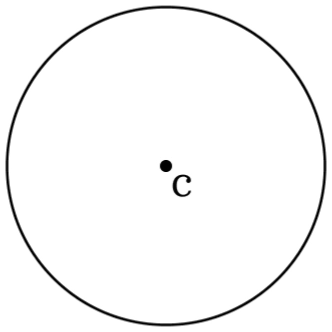

Fig: 2a

But in the case of real city map if we look at all the road connections finding geographical center of city is very difficult.

(We have to consider only roads while finding the center, we can't use apartments, houses etc. like in the below figure (Fig: 2b). Theoretically Central position would be the centroid of the triangle, but it will give use center at Q which lies on apartments and thus is invalid, we have to go along the road to find the optimal point and thus R is the optimal location to set up the Charging station)

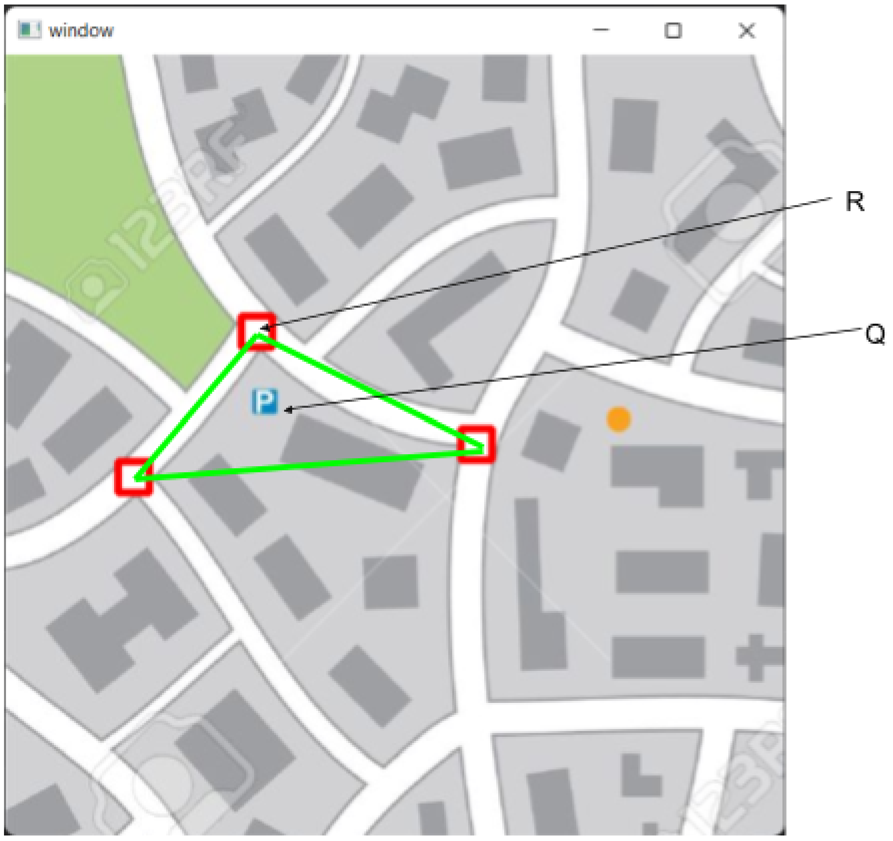

Fig: 2b

So to solve this we applied three approaches.

a) Brute force approach
a) Sliding Window Technique
b) Subblocks Technique

In **brute force** approach we calculated total return by putting CS at each point of grid (50*50) and the point corresponding to maximum total return is the optimal point.

In **sliding window** we took a window of 10*10 and moved over this 50*50 matrix. For each position of the window we sorted all points of which lay on the road and picked the median point as the location of the charging station. Then we calculated the return of each state and summed it. We repeated this for each window position. The CS location for which total return is maximum is the best possible CS location.

In **Subblocks Technique** we divided our whole grid into 4 sub grids (upper left, upper right, lower left, lower right). Then we calculated the return of the median point for each sub grid. The subgrid which has the maximum return is the region of our interest . We now divided this sub grid in 4 parts and repeated the process.

Brute force approach takes maximum time but it gives us the best possible locations. Next,we noticed that the result is almost the same in the Sliding window and Sub-blocks technique like brute force approach with minute error. This tells us that indeed b and c also gives us optimal CS location. This process of subblocks is more efficient than the Sliding window technique which is more efficient than the Brute force approach.

Solution no 3:-

Overhead on Charging stations>

For this I defined two types of overhead on the charging stations.

a) Dynamic overhead

b) Static Overhead

Dynamic overhead tells how many cars are there in the queue, i.e. if we reach the Charging station now, after how much time we will get the turn.

Static Overhead tells about the quality of the charging stations. It doesn't depend on time but is a quality score of that particular station. It tells about on an average when a vehicle is plugged in for charging how much time it takes to get fully charged.

So we notice Dynamic Overhead tells how much time we have to wait in line while Static Overhead tells how much time it will take to get charged. And together these two help us find a charging station which is best for us at that current moment.

But this Dynamic waiting time is not the true representation of actual waiting time(because we have not reached the charging station yet) So we need to calculate the remaining waiting time.

Remaining waiting time is the actual waiting time. Like if Dynamic waiting time of a charging station is 30 min now and time taken to reach the charging station is 45 min then remaining waiting time is 0 minutes. While if Dynamic waiting time is 30 min and time taken to reach is 20 min then remaining waiting time is 30-20 = 10 min.

Let's first look at case when there is only Static Overheads:

Cs1: 50

Cs2: 10

Cs3: 5

Now travel time is:

Cs1: 20

Cs2: 28
Cs3: 38

So total time:
Cs1: 70
Cs2: 38
Cs3: 43
So our algorithm will choose Cs2 >> Cs3 >> Cs1
Let’s verify below.

Now let’s look when we have both **Dynamic overhead** as well as **Static Overhead**:-
Case1>
Dynamic Overhead:
Cs1: 40
Cs2: 48
Cs3: 27

travel time is:
Cs1: 20
Cs2: 28
Cs3: 38

Rem overhead:
Cs1: max(0, 40- 20) = 20
Cs2: max(0, 48-28) = 20
Cs3: max(0, 27-38) = 0

Static Overhead:
Cs1: 50
Cs2: 10
Cs3: 5

Total Overhead: Rem overhead + Static Overhead
Cs1: 70
Cs2: 30
Cs3: 5
Total Time :

Cs1: 90
Cs2: 58
Cs3: 43
So our algorithm will choose Cs3 >> Cs2 >> Cs1 [Let’s see for how many CS it finds travel time]

Note: In this case we see it goes to all 3 CS in search of global optimal point.
—------
Let's now change some values:
Case2>
Dynamic Overhead:
Cs1: 20
Cs2: 48
Cs3: 47

travel time is:
Cs1: 20
Cs2: 28
Cs3: 38

Rem overhead:
Cs1: max(0, 20- 20) = 0
Cs2: max(0, 48-28) = 20
Cs3: max(0, 47-38) = 9

Static Overhead:
Cs1: 10
Cs2: 10
Cs3: 50

Total Overhead: Rem overhead + Static Overhead
Cs1: 10
Cs2: 30
Cs3: 59

Total time :
Cs1: 30

Cs2: 58
Cs3: 38+59= 97

So our algorithm will choose Cs3 >> Cs2 >> Cs1
Here **travel time** is something which we don't know a priori , to calculate it we have to calculate separately for each CS and it is a time taking process.

Our solution to this problem is as follows:
Initially we won't assign any Dynamic overhead, only static overhead and find the optimal cs. Once we find optimal cs we will calculate travel time (Cost - static_overhead). Then we will calculate the rem waiting time= max(Dynamic_overhead , rem_waiting_time)
We will now use this rem_waiting_time + static_overhead for that CS and recalculate cost.
If my next CS is some other thing, we will repeat the process. We will do this till the next CS doesn't change. And that CS will be the optimal solution. (Reason: if CS don't change mean even without applying any Dynamic_overhead to other CS they are farther. So on applying Dynamic overhead it will be even farther)

This greedy approach helps us find the same optimal solution without the need to find travel time for all Charging Stations.
Let's verify below.

Here our algorithm will choose Cs3 >> Cs2 >> Cs1
—---
Note: here also Cs3 is optimal but in this case it only looks once and finds the optimal. (so it saves time)