ASSIGNMENT 1

PROBLEM STATEMENT 3

BY:-

AKSHAT SHARMA(19HS20053) :- IITIANAKSHAT@IIKGP.AC.IN
PRERNA MALAKAR(19IM30017) :- prerna.rizi@iitkgp.ac.in
LAKSHYA MANAN JAISWAL (19IM10017) :- lakshyamanan2277@iitkgp.ac.in
SAGNIK SAHANA(19MA20046):- sagnik0403@iitkgp.ac.in

INTRODUCTION

With the energy crisis and environmental pollution problem, petroleum use and energy diversity have gained global interest and raised public concern in many countries. Transportation sector has occupied 30% greenhouse gas (GHG) in the United States. Therefore, the construction of efficient, intensive, and low-carbon transportation systems has become a controversial topic. Two major approaches to sustainable transportation are being considered: renewable energy technologies and efficient transportation operations. As one way of developing the renewable energy technologies, alternative fuel vehicles (AFVs), such as hybrid vehicles (HVs), plug-in hybrid electric vehicles (PHEVs), and electric vehicles (EVs), have been manufactured to reduce vehicle emissions. EVs that are totally powered by onboard batteries have advantages over traditional vehicles in zero energy emissions and environment protection. With the increasing improvements in battery technologies and charging station construction, EVs are gaining popularity and associated with many fields of transportation.

From the energy consumption perspective, BEVs are driven by high-efficiency motors with the possibility of implementing a regenerative braking system. Additionally, charging a BEV is less expensive than refueling a conventional vehicle because the electric energy is cheaper than its equivalent in fossil fuel.From the emissions point of view, when a BEV is used in combination with renewable sources for the electricity generation, the outcome is a reduction in emissions associated with fossil fuel combustion; therefore, BEVs are one of the best alternatives to be integrated into cities as part of a public transportation system.

The implementation of BEVs in the public transport service involves several challenges associated with the combination of the characteristics of the service with those of the vehicles. Specially, three issues are highlighted: routing of EVs' dedicated to public transport, the scheduling of charge, and the battery health.

First of all, in order to be economically and technically feasible, the BEVs must be routed to minimize the energy consumption. To achieve this, the following two steps are considered:
i) it is necessary to find the minimum consumption paths to travel between two points. For this minimization, the particular characteristic of BEVs must be taken into account, (e.g., the BEVs ability to recover energy by traveling in a way that can take advantage of the regenerative brake; ii) the optimal routes (composed by minimum consumption paths) must be determined to meet the transport demand in different places at different times while the energy consumption is minimized. The computation of the routing scheme of BEVs must be made taking into account the range of the given vehicles, imposed mainly by the available battery technologies. This consideration may require intermediate charging stages of the vehicles to extend their range.

Second, scheduling of charge must be coordinated with routing to guarantee a reliable operation while the cost of charging is minimized. In this way, the scheduling of charge must be made considering the variation of the energy tariff during the peak and non-peak hours of the day. Furthermore, the scheduling of recharge must take into account which is the quantity of energy required to perform the next travel, and the recharging time required to perform the recharge.
Finally, the battery health is considered, as it is sensitive to charging/discharging actions. Also, the battery is the most expensive component of the BEV. Hence, the planning of a charge schedule could help to increase the battery lifetime reducing operational costs.

In this report, we are going to develop both optimal and heuristics algorithms to find the routes of the EVs such that the maximum time taken by the EVs can be minimised.

Problem statement:

In our question,we have considered a city network where we need to route a set of electric vehicles which may require to be charged during its journey from some source to some destination.We are assuming n cities(v1,v2,.....vn) and the distance between two cities vi and vj be eij.If two cities are not connected directly then eij= ∞ and eij=eji. Every city has a single charging point which can charge

one car at a time .Consider a set of k EVs namely P1,P2….Pk. For each EV the following information is provided :-

S_r = Source Code

D_r = Destination Code

B_r = Battery Charged Status initially

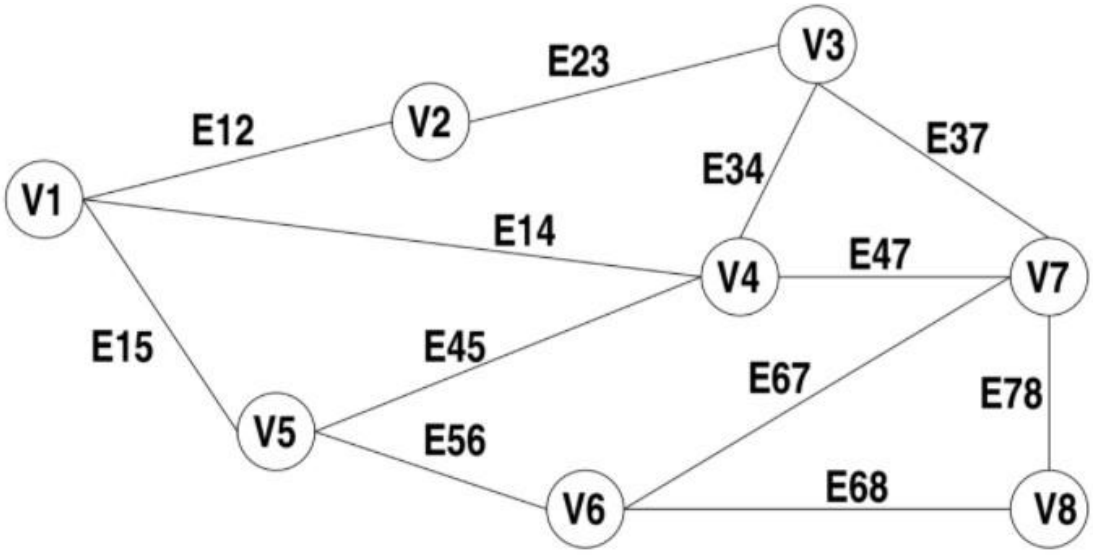
c_r = Charging rate for a battery at a charging station (energy per unit time)

d_r = discharging rate of battery while travelling(distance travel per unit charge)

M_r = Maximum battery capacity

s_r =average travelling speed(distance per unit time)

Assuming that all vehicles start their journey at t=0 and Pr reaches its destination at t=Tr.We need to route all the vehicles from their respective sources to destinations such that max{Tr} is minimized.We will develop optimal as well as heuristics algorithms.



Challenges in the problem statement:-

Our problem consists of various challenges.Some of them are as follows:-

- Battery Capacity:- Different batteries have their different charging and discharging rates. And there is an upper limit on its capacity.So we need to charge them if possible to reach its destination.

- Charging Station:- We have a charging station in every city, but only one car can be charged at a time.

- Average Speed :- Every car has its own average speed which we have to see so that a car can reach its destination without having any issues like the battery of a car getting discharged in between the two cities.

- Cities Connection:- In this problem, all cities are not directly connected.So we have to check how a car would reach its destination to a specified path.

Used algorithm:

Dijkstra algorithm: It is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree. It can be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions).
Dijkstra's algorithm is very similar to Prim's algorithm Like Prim's MST, we generate a *SPT (shortest path tree)* with a given source as root. We maintain two sets, one set contains vertices included in the shortest path tree, the other set includes vertices not yet included in the shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Why are we using distance as a metric to calculate optimal time path with Dijkstra?
To understand why we have decided to optimise distance for each car to find the optimal journey time, let us see the formulation of total journey time for a car:

Total Journey time = Time required to traverse the distance between the cities + Time required to charge at the charging stations + Waiting time at a city

Let us now analyse each of these terms individually:

1. Time required to traverse the distance between the cities: This value is given by *Total path length/Average speed of the car.* We can observe that time required to travel through the cities is directly proportional to the total length of the path. Therefore, we can optimize i.e, minimize total path length to minimize time required to travel through the cities.

2. Time required to charge at the charging stations: This time quantity also depends only on the total path length of a particular route. Following is its mathematical proof:

   Time required to charge at the charging stations of a car = Total path length x Discharge rate of battery - Initial charge on the battery

   Here, initial charge on the battery of a car is a constant and so is its discharge rate. Therefore the time required to charge the battery at the charging stations of a car is dependent only on the total path length.

3. Waiting time at a city: In the problem statement it is mentioned that the charging rate of a car's battery is only dependent on the car and not the charging station. This means the time required to charge is not dependent on the specifications of the charging station as they are the same for all cities and only depends on the path length. Therefore the cars can charge at those cities automatically where the charging stations are unoccupied up to the max charge(if required) and need not wait in a city where the charging station is already occupied thereby reducing the waiting time to virtually zero. Ofcourse, in edge cases where the number of cars is much greater than the number of cities or the max battery capacity of the cars are very low, we may need to consider the waiting time as it becomes significant and the optimal path may diverge from the shortest path due to high traffic.

Algorithm explanation:

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbours and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbour B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.
4. When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

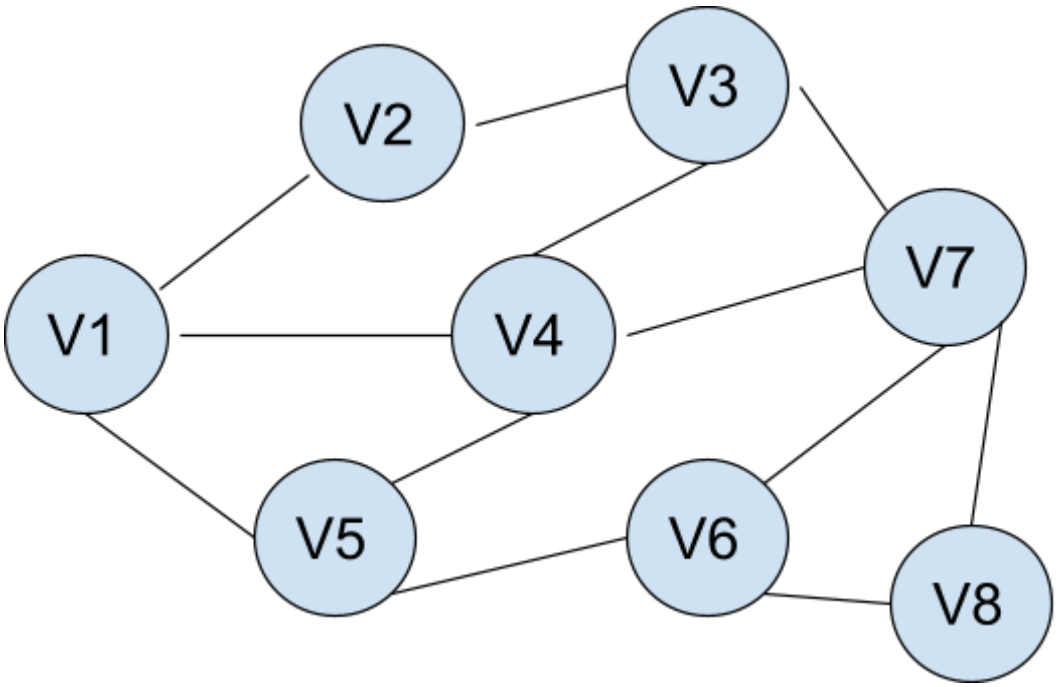
When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

Optimal algorithm in dijkstra's(Explanation): Dijkstra's algorithm requires a priority queue at each of N iterations, where N is the number of network nodes. The best general-purpose priority queues, take O(N^2) cost to find the minimum element. This implies a total running time of O(C*(N^2)) time.

Applying Costs To Edges:

After the data collection process on the identified routes, it was time to apply the formula to calculate the edge costs and complete the graph.
By applying these values to the above mentioned formulae, the cost of each node was determined.
Now that we have the entire data organized in a graph with nodes and weighted directed edges, it is simpler to apply Dijkstra's shortest path algorithm on it.

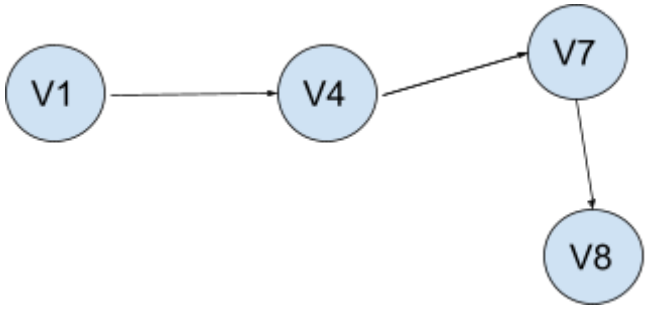
Code demonstration: Let us now take an example with 4 cars and 8 cities, and try to find the optimal path for the cars such that the maximum of the time taken by the cars to complete their journeys can be minimised.
Please refer to the annexure for details of cities and cars used in this example.



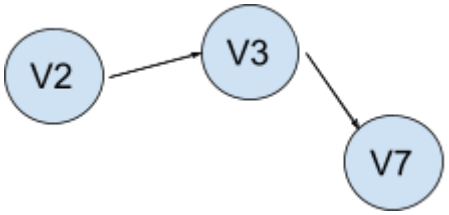
*Edge lengths are not representative of actual distances between the cities

Our algorithm has devised the following route for the cars: C1, C2, C3 and C4:

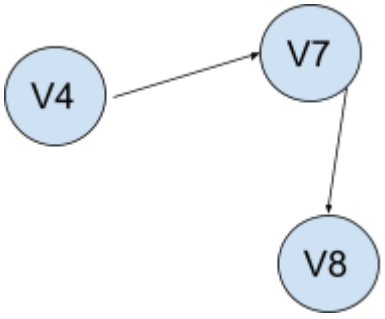
C1: V1 -> V4 -> V7 -> V8



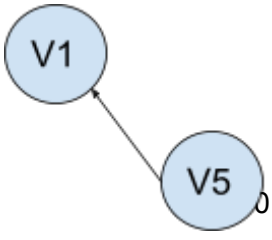
C2: V2 -> V3 -> V7



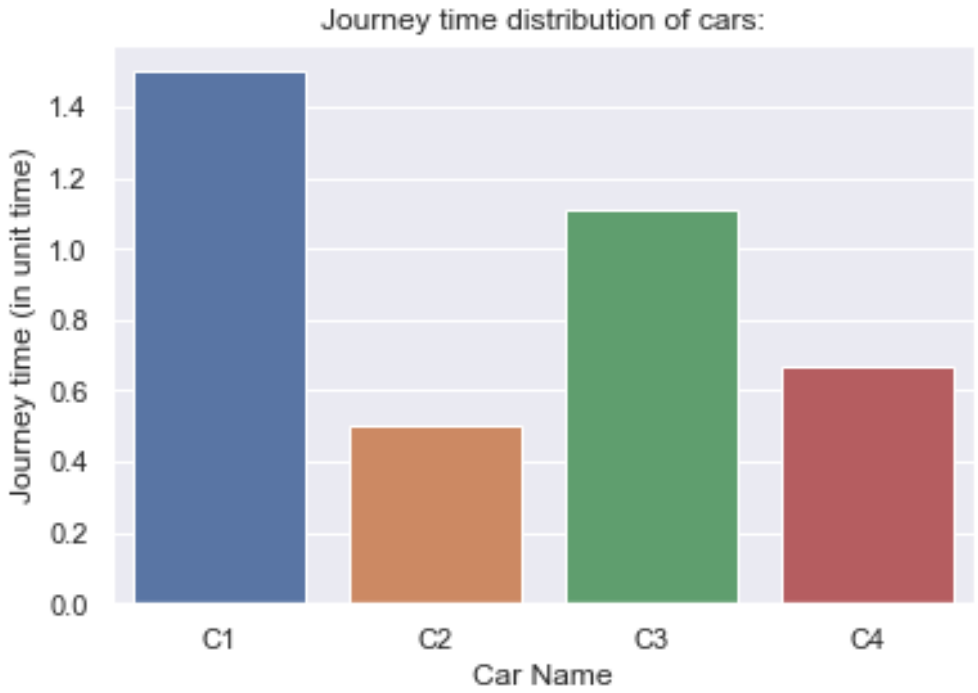
C3: V4 -> V7 -> V8



C4: V5 -> V1



Optimal journey time for the cars is given below:
C1: 1.5 , C2: 0.5 , C3: 1.11 , C4: 0.67



CONCLUSION: We now proceed towards the end of the discussion on this Electric Vehicle Routing Problem(EVRP). Summarising this report, we have seen how to solve the EVRP with given constraints. We saw why Dijkstra's shortest path algorithm was our choice to solve this problem statement along with its detailed working and why path length was chosen as the optimization metric instead of time. Finally we were able to see the performance of our algorithm via an algorithm and the optimal routes charted for each car by the EV.
At last it is worth considering that the addition of seemingly ordinary constraints to our model so that it gets closer to reality, astonishingly increases the difficulty of the problem. For example, considering changes like degradation of the battery, varying number of charging stations for each city, limited allowable EVs at a time in a city, e.t.c may require the use of or, maybe the development of algorithms that are fundamentally different from that which we have used in our current model. One may even need to scrap the entire algorithm and start again from scratch. Let us use this example as a food for thought and take this opportunity to really appreciate the work done by the great thinkers of the past and those who are among us.

ANNEXURE:

- **DETAILS OF CITIES AND CARS IN THE EXAMPLE:**

Details of Cars:

1. C1:
    a. Source : V1
    b. Destination : V8
    c. Initial charge : 5
    d. Charge rate : 4
    e. Discharge rate : 3
    f. Maximum battery capacity : 100
    g. Average speed : 10
2. C2:
    a. Source : V2
    b. Destination : V7
    c. Initial charge : 30
    d. Charge rate : 10
    e. Discharge rate : 15
    f. Maximum battery capacity : 200
    g. Average speed : 20
3. C3:

    a. Source : V4
    b. Destination : V8
    c. Initial charge : 4
    d. Charge rate : 4
    e. Discharge rate : 2
    f. Maximum battery capacity : 100
    g. Average speed : 15

4. C4:
    a. Source : V5
    b. Destination : V1
    c. Initial charge : 6
    d. Charge rate : 2
    e. Discharge rate : 2
    f. Maximum battery capacity : 110
    g. Average speed : 15

Edge cost/distances between adjacent cities:

1. V1:
    a. V2:8
    b. V4:4
    c. V5:10
2. V2:
    a. V1:8
    b. V3:4
3. V3:
    a. V2:4
    b. V4:7
    c. V7:6
4. V4:
    a. V1:4
    b. V3:7
    c. V5:8
    d. V7:6
5. V5:
    a. V1:10

    b. V4:7
    c. V6:5
6. V6:
    a. V5:5
    b. V7:1
    c. V8:2
7. V7:
    a. V3:1
    b. V4:4
    c. V6:3
    d. V8:5
8. V8:
    a. V6:2
    b. V7:5

REFERENCES:

https://www.hindawi.com/journals/jat/2017/4252946/

http://www.hindex.org/2014/p520.pdf

https://arxiv.org/ftp/arxiv/papers/1310/1310.0145.pdf

https://www.osti.gov/servlets/purl/1158748

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

https://www.sciencedirect.com/topics/computer-science/dijkstra-algorithms

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm