Scheduling of Mobile Charging Stations

K.Snigdha

Under the guidance of: Dr. Joy Chandra Mukherjee

IIT Bhubaneswar

April 26, 2019

Introduction

- With increasing innovation and environmental awareness, Electric Vehicles (EV) are gradually replacing the traditional gas/fuel powered vehicles
- This calls for setting up of electric vehicle charging stations in cities and outskirts
- But finding locations to set up the charging stations is inconceivable in the already established settlements
- We suggest a solution to the above stated problem by proposing a new system of charging stations which includes Stationary Charging Stations (CS) and Mobile Charging Stations (MCS)

Proposed Charging System

- In the proposed charging system for electric vehicles (EV), there are a few Stationary Charging Stations (CS) fixed at a location and there are mobile vehicles where are referred to as Mobile Charging Stations (MCS).
- The MCS will be charged during night time from a charging station, and in the day time, they travel around a city to charge EVs.
- Some papers perform single objective optimization of total cost, total tardiness or the use of renewable energy sources. It is also usual to minimize peak demands.

- Each environment and EV charging station present particular characteristics, so each of them gives rise to a different model and poses its own scheduling problem.
- Some systems consider varying the charging rate of the EVs, varying power of the charging station, or varying electricity prices. Other models consider uncertainty and stochastic parameters.
- Due to the complexity of these problems, many existing optimization techniques have been applied to solve them— ABC algorithm, charging in parking lots, linear programming etc.

Objective

The objective of this project is to *schedule m MCSs in such a way that they can charge a maximum number of EVs while minimizing the total distance travelled by the m MCSs*.

Problem Statement 1 - Charging Stationary Electric Vehicles

- Whenever an EV approaches minimum charge, it stops at a junction and sends a charge request to the central node or the CS.
- To charge the batteries in the EV, one or more MCS are scheduled in the city.
- Here, the MCS are required to follow an optimal route and cover all the special nodes/junctions where the EVs are stationed.

Problem Statement 1

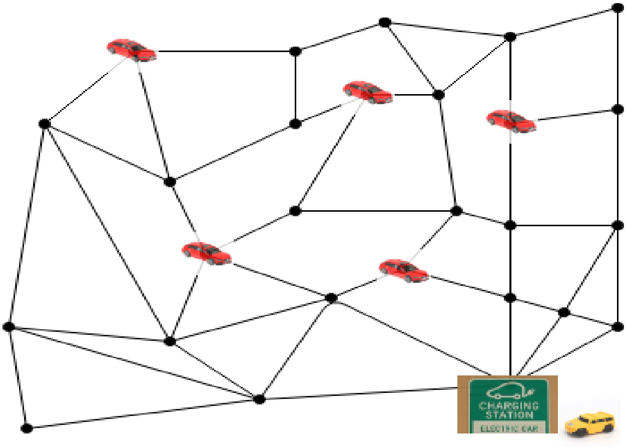

Figure 1: Single MCS Scheduling Model

Work done related to problem 1

- Prepared real time data-set consisting of road dimensions for the cities of San Francisco, Delhi and Mumbai.
- Wrote C++ code to find the all time shortest distance and path between any two points in the city using Floyd-Warshall algorithm.
- Devised and implemented a procedure to schedule a single MCS to charge k EVs using the concepts of graph theory such as finding Hamiltonian paths etc.

Procedure to schedule a single MCS to charge the stationary EVs

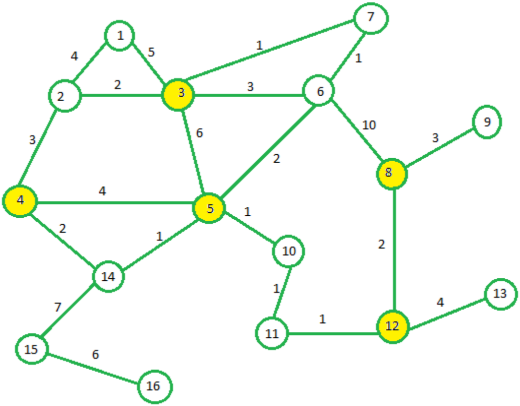

Figure 2: Example graph to show the working of single MCS scheduling

Steps to get the optimal path to be followed by the MCS

- Reduction
- Mapping $(f : v \rightarrow v \forall v \in V(G))$
- Use the below algorithm to find the Hamiltonian path
- Get the complete route to be followed by MCS in the city

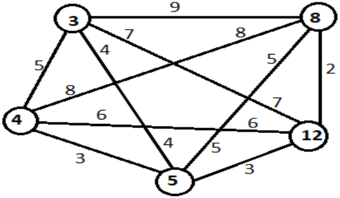

Figure 3: Subgraph formed after reducing the original graph

$$A_G = \begin{array}{c} \\ \\ \end{array}\begin{array}{cc} \begin{array}{ccccc} f(1) & f(2) & f(3) & f(4) & f(5) \end{array} & \\ \begin{bmatrix} 0 & 5 & 4 & 9 & 7 \\ 5 & 0 & 3 & 8 & 6 \\ 4 & 3 & 0 & 5 & 3 \\ 9 & 8 & 5 & 0 & 2 \\ 7 & 6 & 3 & 2 & 0 \end{bmatrix} & \begin{array}{c} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \end{array} \end{array}$$

Figure 4: Mapping for the subgraph

Algorithm to find the optimal MCS path covering k junction nodes

Algorithm 1 Algorithm to find the optimal MCS path covering k junction nodes

```
procedure GETOPTIMALROUTE
    for i = 1, 2, ...k − 2 do
        minNeighbour ← findMin(A_G[i][i + 2] : A_G[i][k]))
        if A_G[f(i)][f(i + 1)] > A_G[f(i)][f(minNeighbour)]) then
            swap(f(i + 1), f(minNeighbour))
```

1: **procedure** GETOPTIMALROUTE
2: **for** $i = 1, 2, ...k - 2$ **do**
3: $minNeighbour \leftarrow \text{findMin}(A_G[i][i+2] : A_G[i][k]))$
4: **if** $A_G[f(i)][f(i+1)] > A_G[f(i)][f(minNeighbour)])$ **then**
5: $swap(f(i+1), f(minNeighbour))$

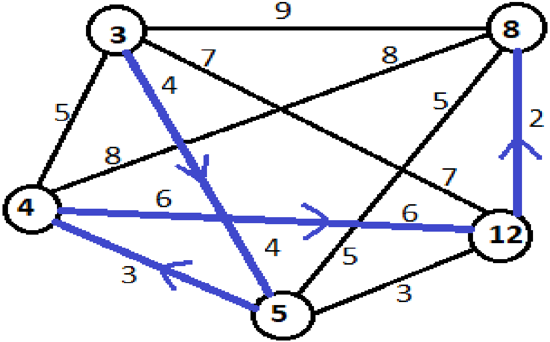

Figure 5: Hamiltonian path in subgraph

Results for Problem 1

```
snigdha@SnigdhaPC: ~/Documents/sem7/BTP
File Edit View Search Terminal Help
snigdha@SnigdhaPC:~/Documents/sem7/BTP$ g++ -std=c++11 optimalChargingRoute.cpp
snigdha@SnigdhaPC:~/Documents/sem7/BTP$ ./a.out

Total number of junctions in New Delhi where EVs are stationed = 28

Optimal route(only through EVs) to be followed by an MCV is :
5 -> 13 -> 19 -> 97 -> 101 -> 203 -> 174 -> 189 -> 237 -> 197 -> 268 -> 271 -> 2
97 -> 341 -> 317 -> 328 -> 333 -> 367 -> 373 -> 293 -> 259 -> 213 -> 209 -> 39 -
> 74 -> 149 -> 161 -> 59 -> CS

Total distance to be travelled by an MCV = 230
snigdha@SnigdhaPC:~/Documents/sem7/BTP$
```

Figure 6: Results obtained when worked on NewDelhi map

Problem Statement 2: Charging moving EVs with multiple MCS

- In this problem statement, when the EVs approach minimum charge, the send a charge request along with their source and destination
- The central server/charging station will assume that the EV will follow the shortest path and will calculate the paths for different EVs.
- It then schedules the MCSs along common paths of the travelling EVs and let them charge 5-6 EVs at the same time.
- The model now makes the best use of the available resources and also satisfies the user requirements at the same time.

Working of Multiple MCSs Scheduling Model

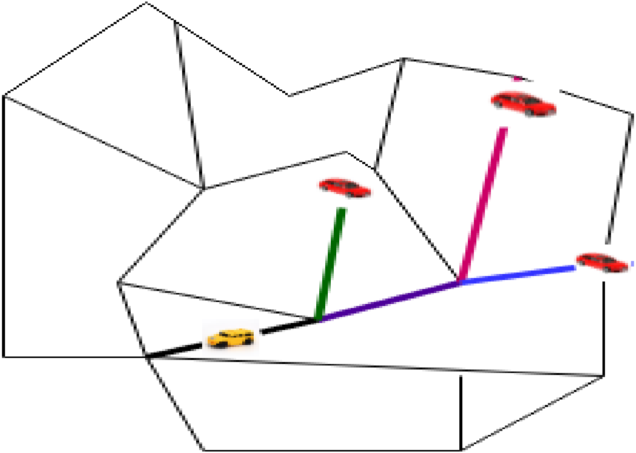

Figure 7: Working of Multiple MCSs Scheduling Model

Work done related to problem statement 2

- Implemented an algorithm to predict real time location of EVs and schedule the m MCSs accordingly
- A typical charge request from an EV consists of $< EV_{id}, time\_of\_request, current\_location, destination, E_{max}, current\_state\_of\_charge >$, where E_{max} is the maximum battery capacity of the EV.

- Given a city map, the current location, and the destination, the server can find the shortest path.
- From E_{max} and current state-of-charge, the server can calculate the current charge in the battery of the EV as (E_{max} * current_state_of_charge).

- Here, we are scheduling m MCSs in such a way that they can charge a maximum number of EVs while minimizing the total travel distance by the m MCSs.
- We refer to this problem as **Multiple Mobile Charging Station Scheduling (MMCSS) problem**.

Problem Statement 2

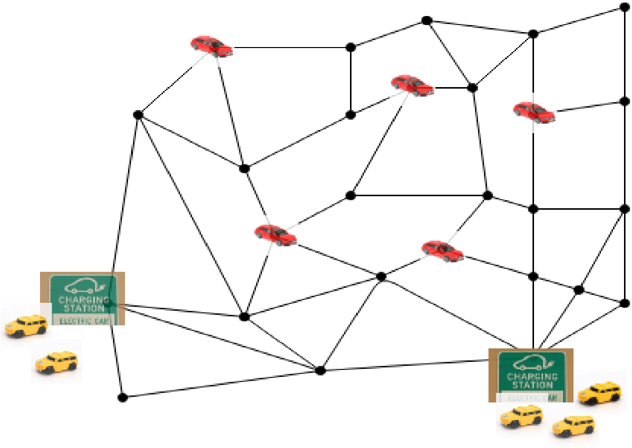

Figure 8: Multiple MCS Scheduling Model

Procedure to schedule multiple MCS to charge all the moving EVs

- Some paths in the city were assumed to be major paths which would be common travel route for most of the EVs.
- Based on these main paths, clustering of the individual EV paths takes place.
- An array of linked lists data structure was used which stores the cluster heads/edges which have some EVs going through it
- The MCS are scheduled to arrive at these edges/heads of each cluster and charge the vehicles going pass it.

Algorithm to cluster the n EV paths and find cluster heads/edges

```
procedure clusterEVPaths
    for i = 1, 2, ...EVPaths.size() do
        maxCommonNodes ← 1;
        belongsToMainPath ← false;
        for j = 1, 2, ...mainPaths.size() do
            v1 ← findCommonNodes(EVPaths[i],mainPaths[j]);
            if v1.size > maxCommonNodes then
                maxCommonNodes ← v1.size;
                belongsToMainPath ← true;
                commonNodesVector ← v1;
                clusterIndex ← j;
        if belongsToMainPath == false then
            mainPaths.push_back(EVPaths[i]);
            commonNodesVector ← EVPaths[i];
            clusterIndex ← mainPaths.size-1;
        if clusters[clusterIndex].isEmpty() then
            clusters[clusterIndex].pushback(commonNodesVector);
        v2 ← findCommonNodes(commonNodesVector,clusters[clusterIndex])
        if v2.isEmpty() then
            clusters[clusterIndex].pushback(commonNodesVector)
        else
            clusters[clusterIndex] = v2;
```

Results for Problem 2

```
snigdha@DESKTOP-I644OGA:/mnt/c/Users/hp/Downloads/BTP-master/BTP-master$ g++ -o comPath findingCommonPath2.cpp
snigdha@DESKTOP-I644OGA:/mnt/c/Users/hp/Downloads/BTP-master/BTP-master$ ./comPath
Number of coordinates = 378
Number of EVs = 15
Enter the source and destination for each EV
351 219
276 223
268 214
298 266
211 62
70 42
175 163
211 139
89 16
44 22
363 271
219 224
186 162
172 199
275 351
219 <- 220 <- 221 <- 268 <- 267 <- 266 <- 289 <- 368 <- 362 <- 361 <- 358 <- 356 <- 354 <- 352 <- 351 <- start
223 <- 221 <- 268 <- 267 <- 270 <- 271 <- 273 <- 274 <- 275 <- 276 <- start
214 <- 212 <- 211 <- 220 <- 221 <- 268 <- start
266 <- 267 <- 270 <- 294 <- 295 <- 298 <- start
62 <- 65 <- 66 <- 67 <- 82 <- 83 <- 84 <- 213 <- 212 <- 211 <- start
42 <- 43 <- 66 <- 65 <- 69 <- 70 <- start
163 <- 145 <- 144 <- 143 <- 142 <- 140 <- 139 <- 175 <- start
139 <- 138 <- 137 <- 136 <- 135 <- 215 <- 214 <- 212 <- 211 <- start
16 <- 31 <- 32 <- 33 <- 34 <- 37 <- 38 <- 39 <- 40 <- 88 <- 89 <- start
22 <- 23 <- 29 <- 42 <- 43 <- 44 <- start
271 <- 270 <- 286 <- 287 <- 288 <- 289 <- 368 <- 362 <- 363 <- start
224 <- 223 <- 221 <- 220 <- 219 <- start
162 <- 163 <- 145 <- 144 <- 143 <- 142 <- 140 <- 139 <- 175 <- 174 <- 173 <- 172 <- 186 <- start
199 <- 198 <- 197 <- 185 <- 186 <- 172 <- start
351 <- 352 <- 312 <- 313 <- 314 <- 318 <- 342 <- 335 <- 336 <- 275 <- start
final clusters:
221      266 267      289 368 362      221 220 219      351 352
65 66      212 211
163 145 144 143 142 140 139 175
16 31 32 33 34 37 38 39 40 88 89
22 23 29 42 43 44
199 198 197 185 186 172
```

Figure 9: Clusters/common paths for a set of EV paths

References

- Ming Zhu, Xiao-Yang Liu, Linghe Kong, Ruimin Shen, Wei Shu, Min-You Wu, The Charging-Scheduling Problem for Electric Vehicle Networks,IEEE
- Y. Ammar, A. Buhrig, M. Marzencki, B. Charlot, S. Basrour, K. Matou, and M. Renaudin, Wireless sensor network node with asynchronous architecture and vibration harvesting micro power generator, 2005
- Dhananjay P. Mehendale, Polynomial Algorithms for Shortest Hamiltonian Path and Circuit, ResearchGate, 2014
- M. Padberg and G. Rinaldi A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, 1991
- Jorge García Álvarez, Miguel Ángel González, Camino Rodríguez Vela and Ramiro Varela, Electric Vehicle Charging Scheduling by an Enhanced Artificial Bee Colony Algorithm, ResearchGate,2018
- Github link of the BTP code : github.com/Snigdha-09/BTP

Thank You!