-Assignment 1-

Optimal & Heuristic solutions for routing Electric Vehicles in a Network with Charging Infrastructure

submitted by

Suraj K | 16AE30032

Department of Aerospace Engineering

Vishal Anand | 19ME10080

Department of Mechanical Engineering

Duhita Wani | 19ME10082

Department of Mechanical Engineering

Abhishek Kumar Singh | 19HS20002

Department of Humanities and Social Sciences

Prof Arijit Mondal

Center for Excellence in Artificial Intelligence

IIT Kharagpur

1.1 Introduction

Computing the shortest path between two locations in road networks is a challenging task in the vehicles routing area and related transportation, distribution and logistics industry. Electromobility has gained substantial momentum in industrial development and production and research has been conducted on the influence of various parameters on intelligent route planning.

Finding the best route from one place to another for some notion of 'best' has significantly gained importance over the last decade especially since the rise of electric vehicles accompanied by range anxiety and the predicament of creating a charging infrastructure for convenient and efficient mobility. Most importantly, for the foreseeable future, EVs have limited driving range, charging stations are still much rarer than gas stations, and recharging is time-consuming. Thus, routes can become infeasible (the battery runs empty), and fast routes may be less favourable when accounting for longer recharging time. With more and more digitisation and adoption of mobile phones, map applications have been emboldened by the availability of significantly more data. Increasingly, vehicular navigation systems have become more autonomous and dynamic thanks to the availability of fast and reliable route planning solutions.

While great amounts of research have been conducted in the area of route planning, the switchover of mobility into electric vehicles demands new solutions as the optimisation parameters have changed: battery capacity, cruising range are severely limited, while driving down-hill and breaking allow for recuperation of energy. Charging itself is a time-intensive process and therefore not viable en route. Hence new approaches are required.

1.2 Problem Description

A city network is considered where a set of electric vehicles which may require to be charged during its journey from some source to some destination are to be routed.

Assumptions:

1. There exists n cities (v1, v2, . . . , vn) and the distance between cities vi and vj be eij (if two cities are not connected directly then eij = ∞ and eij = eji).

2. Each city has a single charging station which can charge one EV at a time. Consider a set of k EVs namely P1, P2, . . . , Pk.

3. For each EV the following information is provided as Input-

- Sr - source node
- Dr - destination node
- Br - battery charge status initially
- cr - charging rate for battery at a charging station (energy per unit time)
- dr - discharging rate of battery while traveling (distance travel per unit charge)
- Mr - maximum battery capacity
- sr - average traveling speed (distance per unit time).

Assume that all vehicles start their journey at t = 0 and Pr reaches its destination at t = Tr. The p[roblem is to route all the vehicles from their respective sources to destinations such that max{Tr} is minimized.

Both optimal and heuristic algorithms are discussed.

1.3 Route Planning Algorithms

In this section, we will explore dynamic route planning algorithms both optimal and heuristic. Hybrid algorithms employing both optimal and heuristic algorithms also exist but are not considered for the purpose of this assignment.

Optimal algorithms find the global optimal solution by exploring the whole set of available solutions whereas heuristic based approaches explore a subset of the available solutions and usually find an approximate optimal solution that comes close to those of the global optimal solutions.

The most common optimal solutions are Dijkstra and Incremental Graph. They find the shortest path from one node to another in the road network. Heuristic based approaches include A*, Genetic Algorithms, Ant Colony Optimization and Tabu search. In order to reduce the computation time during the search process, they accept the best route possible under certain constraints (time, search space etc).

In order to identify which route is more appropriate to the driver request, a set of metrics need to be determined to compare the different available routes that the electric vehicle takes from the current

location to the destination. Following below is a number of significant metrics that routing algorithms use to make the optimisation decision:

- Travel distance: This is the basic criterion for shortest path finding. There is a length associated with each road. Usually the shortest route would mean searching the route where the vehicle has to travel the shortest duration.

- Travel Time: In this case, the fastest route is considered to be the optimal path than the one with the shortest route. The shortest route and the fastest route may be different because of congestion in traffic, speed limit and other restrictions. The fastest path is also a continually changing path as the situation on the road changes.

- Travel cost: The travel cost estimates situations on the path where consideration would have to be given to the toll tags, fuel consumption levels in the journey.

- Charging rate: In the case of electric vehicles, the charging rate determines the time expended in charging at a particular station, and the duration the following vehicle (if any) would have to wait for its turn to charge.

- Discharging rate: The discharging rate is a significant factor in determining whether an electric vehicle can even afford to take a particular path. It can only take a particular path if the current charge in its battery is enough to take it to a charging station or destination before it runs out. An additional factor would still be necessary in cases such as these in case of congestion at a charging station such that the addition of the wait time would still mean the path takes less time than the alternatives.

To measure the performance of any of the algorithms three key metrics are considered generally- computational complexity, scalability, quality of the best route.

1. Computational Complexity- In situations where the path metrics like traffic etc are continually updated, an algorithm that finds an optimal route with less travel time and more computational time may not always be the most apt.

2. Scalability- An efficient algorithm in a small road network with fewer cars may not be the most appropriate one for a larger network.

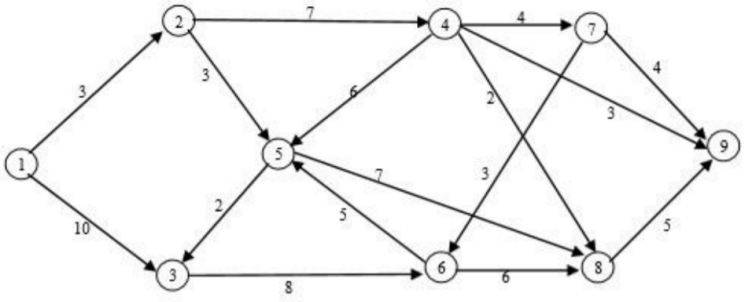
3. Quality- This metric is used to compare the different best routes calculated by different heuristics according to the same metrics in order to determine which algorithm is calculating the closest solution to the problem.

1.4 Dijkstra Algorithm

This algorithm is usually used for finding the shortest paths from source to all vertices in the given graph. Given a graph and a source vertex in the graph a route of minimum weight connecting two specified vertices are determined, source and destination, in a weight graph (digraph) in a transportation network. It is introduced by the famous Dutch computer scientist Edsger W. Dijkstra, was recognized as the best algorithm that can be applied to get the shortest path from a node to any other nodes hence this problem is sometimes called the single-source shortest paths problem.

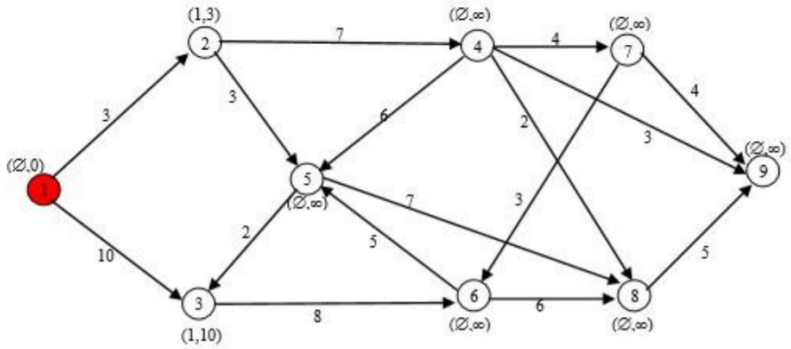
Since the speed of the EVs is constant the shortest path route will be the optimal route for minimizing the travelling time of EVs. With the Dijkstra Algorithm shortest path to all nodes from the source node can be found efficiently which solves a major part of our problem. We maintain two sets, one set contains vertices included in the shortest-path tree, and the other set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.

To understand the algorithm lets go step by step

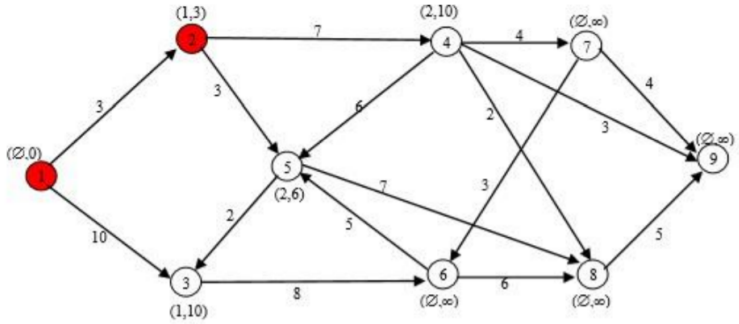


Step 1: Label each node with a pair of labels (pj, dj), where pj is the node preceding node j in the existing shortest path from 1 to j, dj is the length of this shortest path. Node 1 with permanent labels (Ø,0). Label all other nodes in the graph with temporary labels (Ø,∞).
We colour the nodes with permanent labels red.

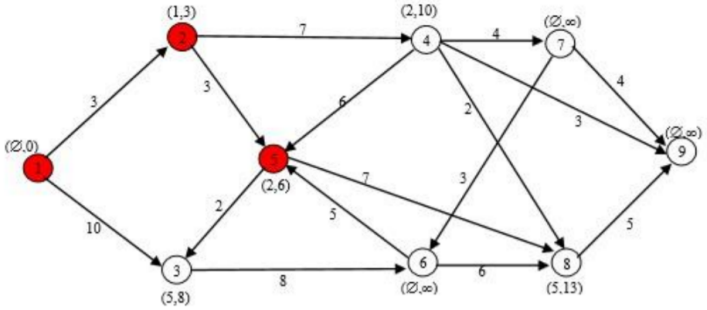
Step 2: Initialise two set one empty(*shortest*) and one with all other vertices. Now pick the node with a minimum dj value that is not in *shortest*. Hence first source node 1 will be selected.



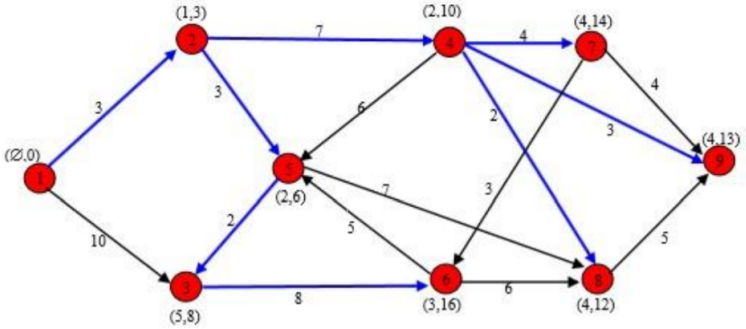
Step 3: Update the labels of vertices adjacent to the piked vertice and pick the node with a minimum dj value that is not in the *shortest* set already.



Repeat step 3 until the *shortest* does include all vertices of the given graph



The Shortest Paths from 1 to all other nodes:



pseudo code for Dijkstra

```
1:      function Dijkstra(Graph, source):
2:      for each vertex v in Graph:           // Initialization
3:      dist[v] := infinity// initial distance from source to vertex v is set to infinite
4:      previous[v] := undefined  // Previous node in optimal path from source
5:      dist[source] := 0  // Distance from source to source
6:      Q := the set of all nodes in Graph  // all nodes in the graph are unoptimized - thus are in Q
7:      while Q is not empty:      // main loop
8:      u := node in Q with smallest dist[ ]
9:      remove u from Q
10:     for each neighbor v of u:  // where v has not yet been removed from Q.
11:     alt := dist[u] + dist_between(u, v)
12:     if alt < dist[v]      // Relax (u,v)
13:     dist[v] := alt
14:     previous[v] := u
15:     return previous[ ]
```

Complexity

Time complexity: Θ(V^2)

Space complexity: Θ(V)

1.5 A* Algorithm

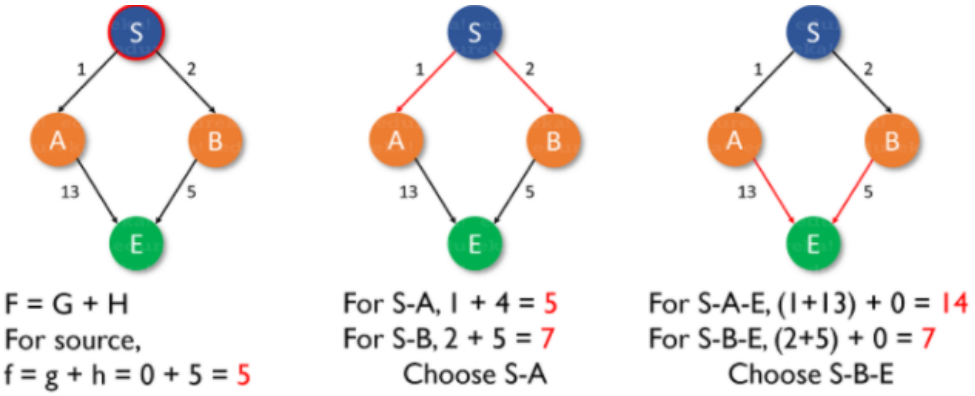
A* is like Dijkstra's Algorithm in that it can be used to find a shortest path. A* is like Greedy Best-First-Search in that it can use a heuristic to guide itself. In the simple case, it is as fast as Greedy Best-First-Search .

The secret to its success is that it combines the pieces of information that Dijkstra's Algorithm uses (favoring vertices that are close to the starting point) *and* information that Greedy Best-First-Search uses (favoring vertices that are close to the goal). In the standard terminology used when talking about A*, g(n) represents the *exact cost* of the path from the starting point to any vertex n, and h(n) represents the heuristic *estimated cost* from vertex n to the goal. In the above diagrams, the yellow (h) represents vertices far from the goal and teal (g) represents vertices far from the starting point. A* balances the two as it moves from the starting point to the goal. Each time through the main loop, it examines the vertex n that has the lowest f(n) = g(n) + h(n).

Let me just show you a simple example to help you understand how this algorithm works.

Suppose we have a small graph with the vertices: S, A, B, E where S is the source and E is the destination.

Remember that the cost to enter the source and destination is always 0.



The heuristic values are:

S -> 5
A -> 4
B -> 5
E -> 0
Let's use the formula and calculate the shortest path from the source to the destination now.

f = g + h where g is cost to travel and h is the heuristic value.

To reach Source:

$$f(S) = 0 + 5 = 5$$

The paths from S to other vertices:

$$f(S\text{-}A) = 1 + 4 = 5$$

$$f(S\text{-}B) = 2 + 5 = 7$$

So, we firstly will choose the path of S -> A as it is the least.

The paths from A and B to the Destination:

$$f(S\text{-}A\text{-}E) = (1 + 13) + 0 = 14$$

$$f(S\text{-}B\text{-}E) = (2 + 5) + 0 = 7$$

After calculation, we have now found that B later has given us the least path. So, we change our least path to S-B-E and have reached our destination. That is how we use the formula to find out the most optimal path.

Pseudo Code for A*

```
let the openList equal empty list of nodes
let the closedList equal empty list of nodes
put the startNode on the openList (leave it's f at zero)
while the openList is not empty
  let the currentNode equal the node with the least f value
  remove the currentNode from the openList
  add the currentNode to the closedList
  if currentNode is the goal
    You've found the end!
  let the children of the currentNode equal the adjacent nodes
  for each child in the children
    if child is in the closedList
      continue to beginning of for loop
    child.g = currentNode.g + distance between child and current
    child.h = distance from child to end
    child.f = child.g + child.h
    if child.position is in the openList's nodes positions
      if the child.g is higher than the openList node's g
        continue to beginning of for loop
    add the child to the openList
```

Complexity
Time Complexity: O(b*d)
Space Complexity: O(b^d)

1.6 Conclusion

The original problem is of minimizing the max trip time for a battery-electric vehicle on the road networks. As battery capacity is limited, stops at charging stations may be inevitable. Without recharging, large parts of the road network are simply not reachable by an EV, rendering long-distance trips impossible. For conventional cars, the broad availability of gas stations and short refuel duration allows neglecting this issue in route optimization. Careful route planning is crucial, since charging stations are scarce and recharging is time-consuming. In this work, we have found the route for fastest travel(capacity-constrained), considering extra time charging stops when necessary. Using fixed scalar arc

weights corresponding to, e. g., driving time. But respecting battery constraints, minimize overall trip time, including time spent at charging stations make the problem very complicated and **NP-hard**.

What could have done is the use of a label-setting search, which is capable of propagating continuous tradeoffs induced by charging stops using labels of constant size. Since the problem is NP-hard, it is not guaranteed polynomial running times. For faster queries, we could have used heuristic approaches that offer high (empirical) quality.

1.7 References

- https://www.hindawi.com/journals/mpe/2017/5098183/#B12
- https://mediatum.ub.tum.de/doc/993196/file.pdf
- (PDF) Efficient Energy-Optimal Routing for Electric Vehicles. (researchgate.net)
- https://www.researchgate.net/publication/271458021_A_comparative_study_of_vehicles'_routing_algorithms_for_route_planning_in_smart_cities
- (PDF) Heuristics for Electric Vehicle Charging Station Allocation Problem (researchgate.net)
- Bangladesh Journal of Multidisciplinary Scientific Research (cribfb.com)