EV Charging Station Network Planner

Submitted in partial fulfillment of the requirement for the degree of
Bachelor of Technology
In
Computer Science and Engineering

by
DHANANJAY SUNIL MENON
18BCE2330

Under the guidance of
Dr. Boominathan Perumal
SCOPE
VIT, Vellore

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Executive Summary

One of the main concerns of buying an electric vehicle is the charge time, and the availability of charging stations in the region. With the arrival of vehicles like Tata Nexon EV and MG ZS EV, India is slowly moving towards mass production of affordable EV vehicles. However, to encourage the public to buy EV vehicles, charging stations have to be set up at every corner of the country.

The travelling salesman problem is an NP-Complete problem that has a variety of applications such as the shortest route finder on google maps, and the most optimal path a machine takes to drill holes into a microchip and route planning. It has been an active topic of discussion for decades.

This study proposes a method of using the travelling salesman problem to set up an optimal network for electric vehicle charging stations in all the districts In India. It also proposes a method for solving the travelling salesman problem for a large number of coordinates using the google maps API.

Key Words : Travelling Salesman Problem, Google Maps distance matrix API, EV charging stations.

Contents

Introduction

Objective

This project aims to propose a network of EV charging stations throughout India, such that an electric vehicle can travel from any point to another in India. The travelling salesman problem would be used to show that all districts in India can be covered with an electric vehicle.
The objective of this project is to:

- Perform a large-scale travelling salesman problem using the google maps API.
- Create a software to display the route including the charging stations from any origin to any destination in India

Motivation

India is a vast land that covers an area of around 3.287 million km. As the world slowly transitions to electric vehicles, one of the main investments the country should make is on the setting up of electric charging stations. However, this should be done economically. There are many factors that have to be considered when setting up a network of EV charging stations. The parameters include population density and distance between two charging docks.

Background

India currently has 749 districts. Finding the most optimal path for a large number of coordinates can be challenging. By solving for the travelling salesman problem, we can achieve algorithms with low cost to arrive at a good solution. The travelling salesman problem has many applications. For example, planning the most optimal route to deliver a set of packages to different routes.

Technical Specification

Python 3.0 will be the language that is used for the project.

I. GOOGLE MAPS API

Google Maps Platform provides various APIs developers can use the Google Map technology for their websites and programs. This project uses four google maps API's.

Distance Matrix API

This API takes in a set of origins and destinations as a request and returns the distance between all the origins and destinations in a matrix format. In this project, this API is used for solving the travelling salesman problem using real road distances. The distance matrix API is also used for finding the nearest charging station when calculating the route of a vehicle.

Directions API

This API is used for plotting the direction between points. The directions API takes in the origin, destination, and waypoints as a request. (Waypoints are the intermediate stops between the origin and the destination). The response is a GeoJSON file. GeoJSON files are used for encoding a variety of geographic data.

JavaScript API

This API is used for plotting routes on a map and displaying it to the user. Points, and lines can be plotted on the map.

Places API

This API is used in the ‘Nearby Attractions’ function. This API helps display places in a certain category with the radius of a point. This API also gives additional features like reviews and the opening and closing times of places.

II. TKINTER

The Tkinter package is a Python interface that is available on Windows, Mac OS and UNIX systems. This package is used for the display of a user interfaces along with the integrated maps. TkinterMapView is an interactive map renderer widget for the python Tkinter library. Placing widgets and drawing paths are possible within this map.

III. GOOGLE OR TOOLS

OR tools is an open source software created by google used for combinatorial optimization problems. The travelling salesman problem for asymmetric data points can be solved by using this tool. The Google OR Tools is used for the divide and conquer algorithm to solve the TSP for the smallest unit – a district.

Design Approach and Details

Design Approach / Materials & Methods

For solving the travelling salesman problem, a dataset of coordinates is required. For this, the current list of petrol stations are used for generating an initial dataset. Later, a more economical network would be laid out using the travelling salesman problem. The input would be a list of coordinates of EV charging stations. The output would be a list of coordinates in the order that would cover the most optimal path. The longest edge in the route would also be displayed, to show that it is within the range of mileage.

A software is also be created to display the route, and the time taken, for a trip from an origin to a destination.

Divide and Conquer Algorithm

The travelling salesman problem can be solved by dividing a long list of coordinates into smaller sections. This project uses coordinates from all over India, which have already been divided district wise. Hence, each district is used as the smallest unit to solve for the travelling salesman problem.

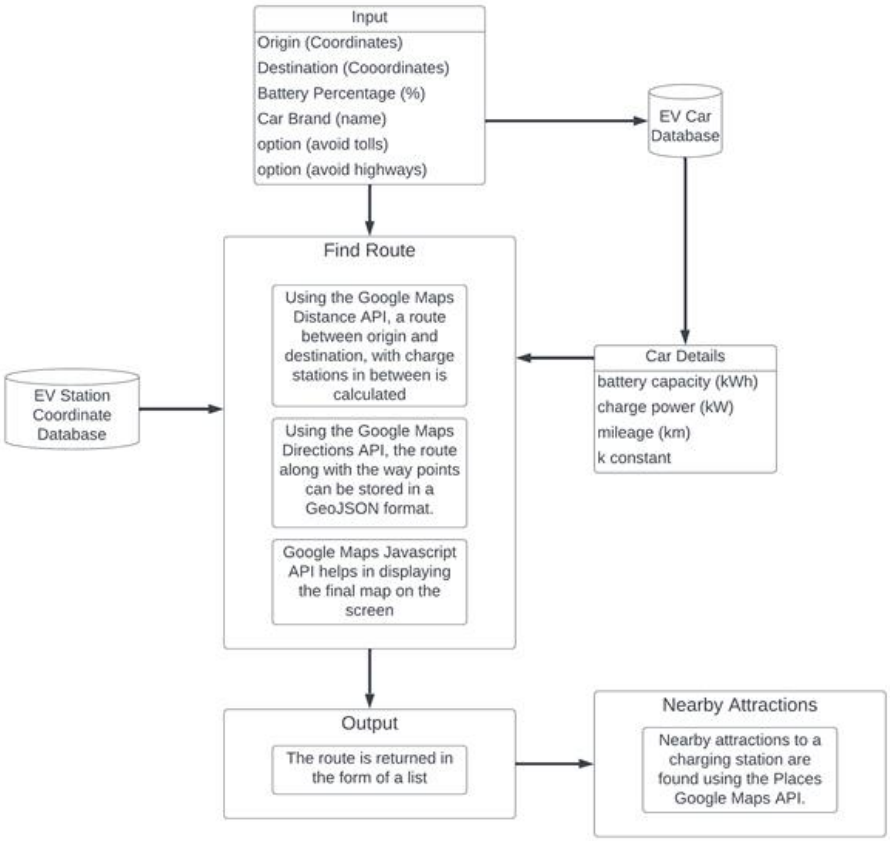
Once the most optimal path is found for each district within a state, the paths are combined to form an optimal path for a state. By combining paths from each state, the optimal path for the entire country can be found.

Finding EV vehicle route

Given an origin, destination, an EV vehicle and the initial charge, this project can find a route that consists of EV charging stations along the way so the vehicle does not run out charge mid-trip. This is useful for long distance trips.

Initially, the mileage of the vehicle is calculated from its current position. If the destination does not fall within this mileage, then an EV station is found within the mileage. The EV charging station should be selected such that the deviation of path from origin to destination should be minimal. Also, the vehicle has to travel a sufficient distance before reaching an EV station.

Nearby Attractions is an additional feature which locates places under a certain category within a given radius. This is so that a user can find places to spend time while the vehicle is charging. The charging time generally would not exceed 20 minutes. In this project, the category of places listed are food outlets. The radius is determined based on the wait time (time taken by the vehicle to gain sufficient charge).



Input for finding a route:

- Origin
- Destination

- EV Car Brand
- Battery Percentage
- AvoidTolls, and AvoidHighways Option.

Output for finding a route:

- The route with stop-by EV-Stations
- The distance and time taken for each trip and stop
- Nearby attractions for each petrol station

Electric Vehicle Details

The details of electric vehicles are saved in the form of a csv file. There are four attributes to electric vehicles:

- Battery Capacity (kWh)
- Charge Power (kW)
- Mileage (km)
- K constant (km/kWh)

Car Name	Battery Capacity (kWh)	Charge Power (kW)	Mileage (km)	k (km/kWh)
MG ZS EV	49	75	270	5.510204082
Tata Tigor EV	26		306	11.76923077
Tata Nexon EV	30.2		213	7.052980132
Mini Cooper SE	28.9	49	185	6.401384083
Jaguar I-Pace	84.7	104	380	4.486422668
BMW iX xDrive40	71	150	350	4.929577465
Porsche Taycan	71	225	410	5.774647887
BYD E6	71.7	100	400	5.578800558
udi e-tron 55 quatt	86.5	155	365	4.219653179
Tesla S	95	250	560	5.894736842
Tesla 3	57.5	170	380	6.608695652
Tesla X	95	250	465	4.894736842
Tesla Y	75	210	415	5.533333333

For EV vehicles, we know that the time taken for a journey would be the sum of travel time and charging time.

Time Taken = travel time + charging time.

The travel time can be calculated using the formula below

$$Travel\ Time = \frac{distance}{avg.speed}$$

The time taken to charge a vehicle can be written in terms of the power stored in the battery and the charge power (also known as the speed of charging) :

$$Charge\ time = \frac{Power\ (kWh)}{Charge\ Power\ (kW)}$$

The distance/mileage received from the amount of charge (power) stored in the vehicle can be described in the following way.

$$Distance = k \times Power(kWh)$$

In the above formula, k is a constant dependent on the vehicle's performance. Generally, the average value of k can be taken as 4 miles per kWh.

Codes and Standards

Divide and Conquer Algorithm

To get route distances between places, instead of straight-line distances, the google matrix distance matrix API is used. A distance matrix API request can have a maximum 12 places as origin and destination. Hence, multiple requests must be sent in-order to create a larger distance matrix.

This is done by dividing the list of coordinates to portions of 10. So, for 'n' places, the number of matrices required are

$$No: of\ matrices\ =\ [ceil\left(\frac{n}{10}\right)]^2$$

The matrices of size 10 are combined to form a larger matrix. For a district, the number of EV charging stations does not exceed 40. Hence, a maximum of 16 matrices may be used to make a larger matrix. The following figure shows the method of combining matrices.

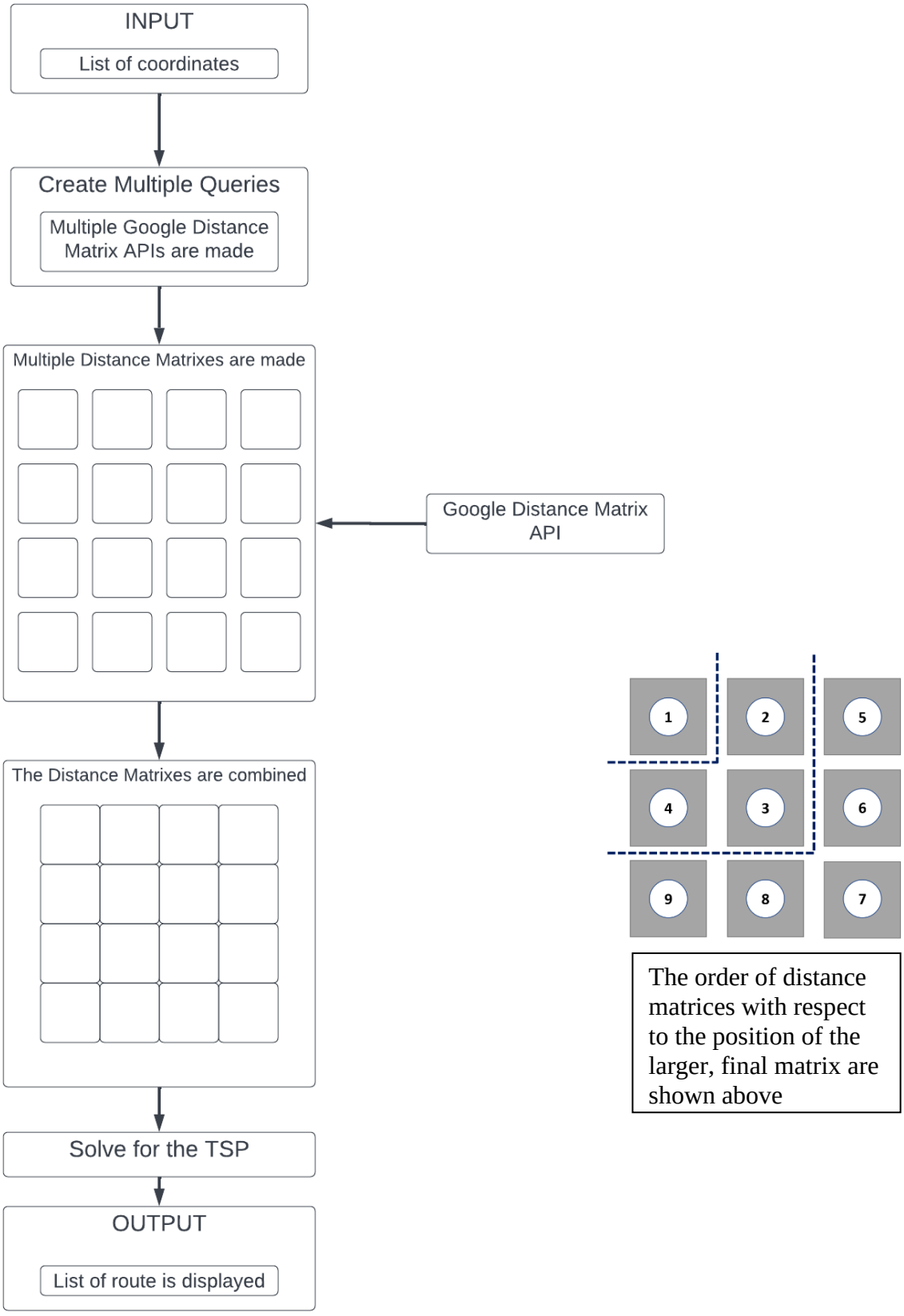


Figure 1: Solving for Travelling Salesman Problem using the Google Maps Distance Matrix API

The most optimal route for each district is calculated using Google's OR tools. The optimal route of all districts in a state are connected to each other such that they form an optimal route for a state. Similarly, the optimal route of all states is connected to each other such that they form an optimal route for the whole country.
The Google Maps Distance API is used for the calculation of road distances between places.

Finding EV vehicle route

The coordinates of EV Stations, details of EV vehicles are stored in different CSV files. Tkinter for python is used to set up a user interface and display the map when necessary.
The geopy library is used for calculation of straight line distances, meanwhile the Google Maps distance matrix API is used to calculate straight line distances.
Directions API is used to calculate the road route between two points. The output for a Directions API request is a JSON file. This JSON file contains an encrypted line called the polyline. Using the polyline library in python, this encrypted line can be decrypted to display a list of points. By connecting this list of points with straight lines, a road route is generated.
Below is the function to get the distance and time between an origin and a destination for road distances.

```
def get_distance_and_time(self,origin,destination,avoid_tolls,
avoid_highways):
    avoid_string = "ferries,indoor"
    if(avoid_tolls):
        avoid_string = avoid_string + ",tolls"
    if(avoid_highways):
        avoid_string = avoid_string + ",highways"
    origin_url = str(origin[0]) + "," + str(origin[1]) + "|"
    destination_url = str(destination[0]) + "," +
str(destination[1]) + "|"
    data_type = "json"
    endpoint =
f"https://maps.googleapis.com/maps/api/distancematrix/{data_type}
"
    params = \
        {"origins": origin_url,
         "destinations": destination_url,
         "mode":"driving",
         "avoid":avoid_string,
         "key": api_key}
    url_params = urlencode(params)
    url = f"{endpoint}?{url_params}"
    print(url)
    r = requests.get(url)
```

```
    distance =
r.json()['rows'][0]['elements'][0]['distance']['value']
    distance_km = float(distance/1000)
    time =
r.json()['rows'][0]['elements'][0]['duration']['value']
    R = []
    R.append(distance_km)
    R.append(time)
    return R
```

Tkintermapview is an interactive map renderer for the Tkinter library for python. The below code sets up the map widget on a tkinter frame

```
map_widget_obj = tkintermapview.TkinterMapView(self.root_i2,
width=800, height=600, corner_radius=0)
```

On this widget, markers and paths can be displayed with the below code:

To set up marker

```
map_widget_obj.set_marker(x_coordinate, y_coordinate, text="Pin")
```

To draw path

```
path_1 = map_widget_obj.set_path(list_of_coordinates)
```

Interface 1

The user is able to enter the origin and destination. The user is also able to enter the vehicle type, initial charge, option to avoid tolls and option to avoid highways. After pressing the enter button, the user is taken to the next screen.

Interface 2

After entering the required details and pressing the enter button, the user is taken to an overview of the route to follow. The user can press a breakdown button to see a breakdown of the route (Interface 3)

Interface 3

The user can see the duration of trip between two EV stations and the final battery percentage after the trip. The user can also see the charge time required at a charge station.

Interface 4

In the breakdown interface (Interface 3), there is a button for EV station stops titled "Nearby Attractions". This interface shows the nearby food outlets from the EV station which are within walking distance. On selecting a place, the directions to that place are displayed.

Constraints, Alternatives and Tradeoffs

Straight line distances and Road Distances

Calculating road distances consumes more time than calculating the straight line distances between two points. To initially find nearby EV stations, straight line distances are used. This could lead to an over-estimation of mileage. To solve this issue, a buffer charge of 30% is kept in calculations to prevent the vehicle from running out of charge mid trip.

Comparison of Algorithms

The algorithm for travelling salesman problem in this project takes into consideration Google Map road distances. Other algorithms of the travelling salesman problem involve straight line distances. The comparison between the divide and conquer algorithm and the Concorde TSP solver would compare straight line distances.

Schedule, Tasks and Milestones

Understanding of the travelling salesman problem

The travelling Salesman Problem has a variety of algorithms to find an optimal route. These include cheapest insertion, k-opt Algorithm and cutting plane method. The coordinates of all the districts in India were used to make comparisons between the algorithms. A comparative study between each algorithm was created. Concorde TSP Solver was found to be the most accurate Travelling Salesman Problem to date.

1st January – 31st January

Setting up Google Maps API

The google maps API was studied and used to generate a solution to the travelling salesman problem using road distances. Google Maps API was also used to generate directions, display a map and find nearby attractions.

1st February – 28th February

Generating an EV Routing Model

The first version of the program was created to find a route provided an origin, destination and car model. This involved:

- Studying the parameters of EV vehicles in India.
- Creating a dataset of EV charging stations.
- Creating an algorithm to generate a route consisting of EV stations.
- Plotting and drawing routes using Google Maps API

15th February – 31st March

Creating an Interface

Tkinter library of python was studied and used to create a user interface. Tkintermapview is a library that is used to display maps in the Tkinter interface.

01st April - 30th April

Working on improvements

Improvements made in this project over time include adding a nearby attractions feature. This feature involves showing a list of attractions near an EV station. The other improvement was including an option of avoiding tolls and avoiding highways.

01st May – 22nd May

Divide and Conquer Algorithm

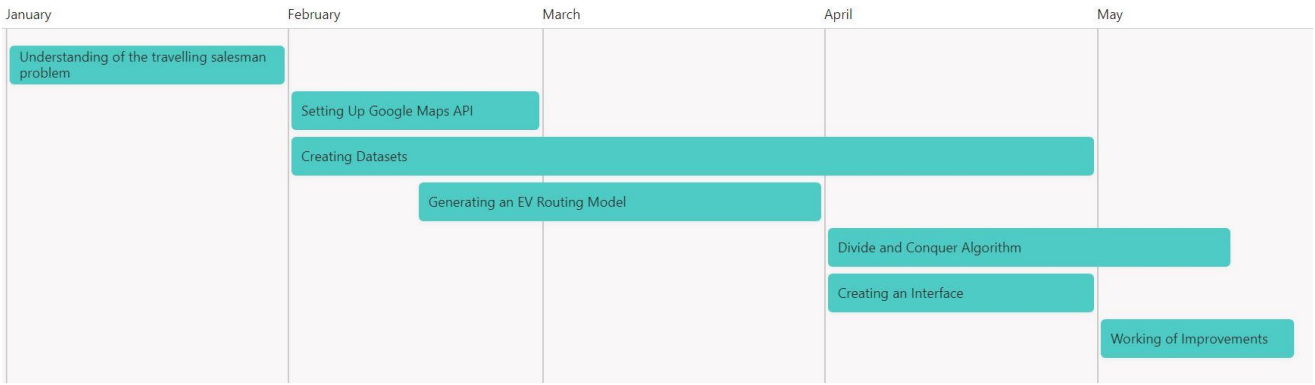
This involved extracting data from csv datasets of coordinates of EV stations to generate an optimal route using the divide and conquer algorithm.

01st April – 15th May

Creating datasets

Datasets of all districts in India, and datasets of all petrol stations in India were not available. To work on a project that focusses on India, the datasets had to be manually made.

01st February – 30th April

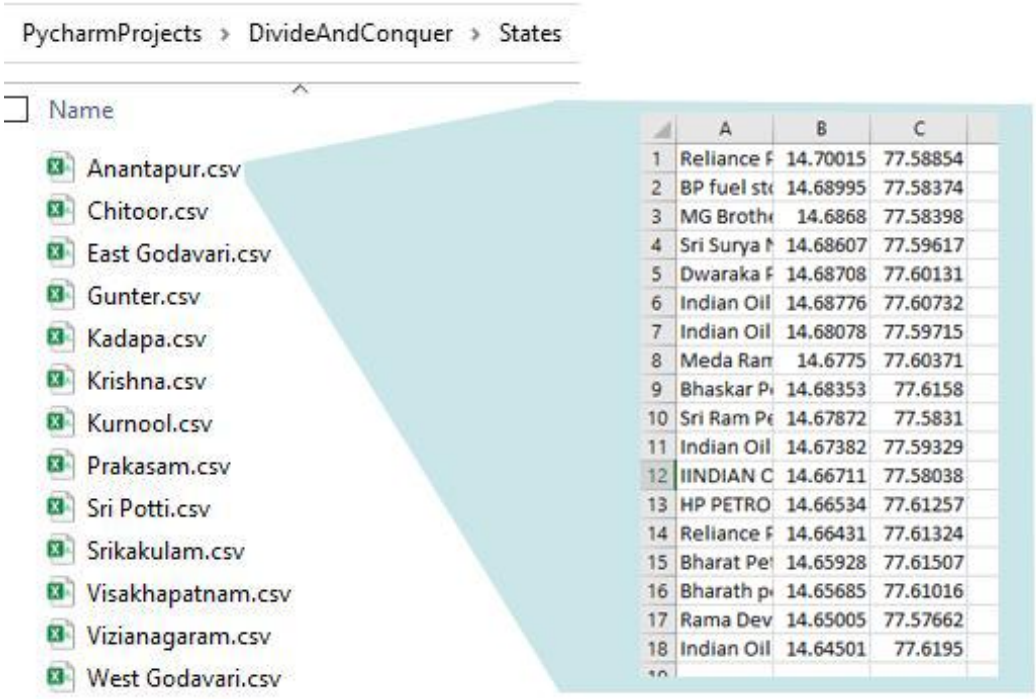


Project Demonstration

Divide and Conquer Algorithm

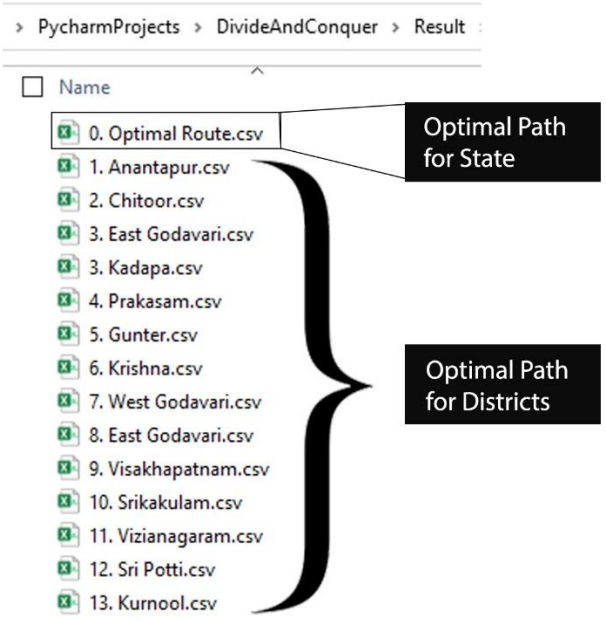
The divide and conquer algorithm takes a set of coordinates for each district as input. The resulting output are multiple csv files showing the most optimal path between the coordinates.

Input



	A	B	C
1	Reliance F	14.70015	77.58854
2	BP fuel st	14.68995	77.58374
3	MG Brothe	14.6868	77.58398
4	Sri Surya M	14.68607	77.59617
5	Dwaraka F	14.68708	77.60131
6	Indian Oil	14.68776	77.60732
7	Indian Oil	14.68078	77.59715
8	Meda Ran	14.6775	77.60371
9	Bhaskar P	14.68353	77.6158
10	Sri Ram Pe	14.67872	77.5831
11	Indian Oil	14.67382	77.59329
12	IINDIAN C	14.66711	77.58038
13	HP PETRO	14.66534	77.61257
14	Reliance F	14.66431	77.61324
15	Bharat Pet	14.65928	77.61507
16	Bharath p	14.65685	77.61016
17	Rama Dev	14.65005	77.57662
18	Indian Oil	14.64501	77.6195

Output

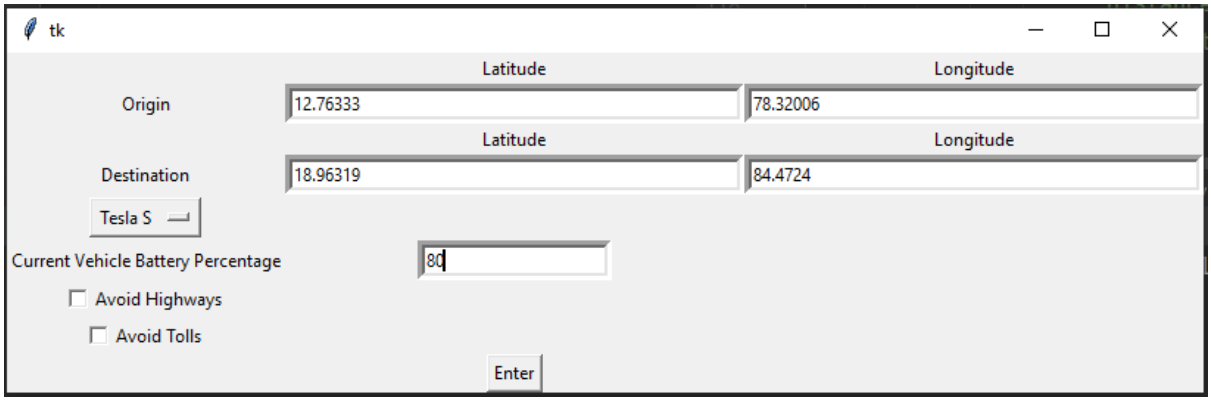


Optimal Path for India

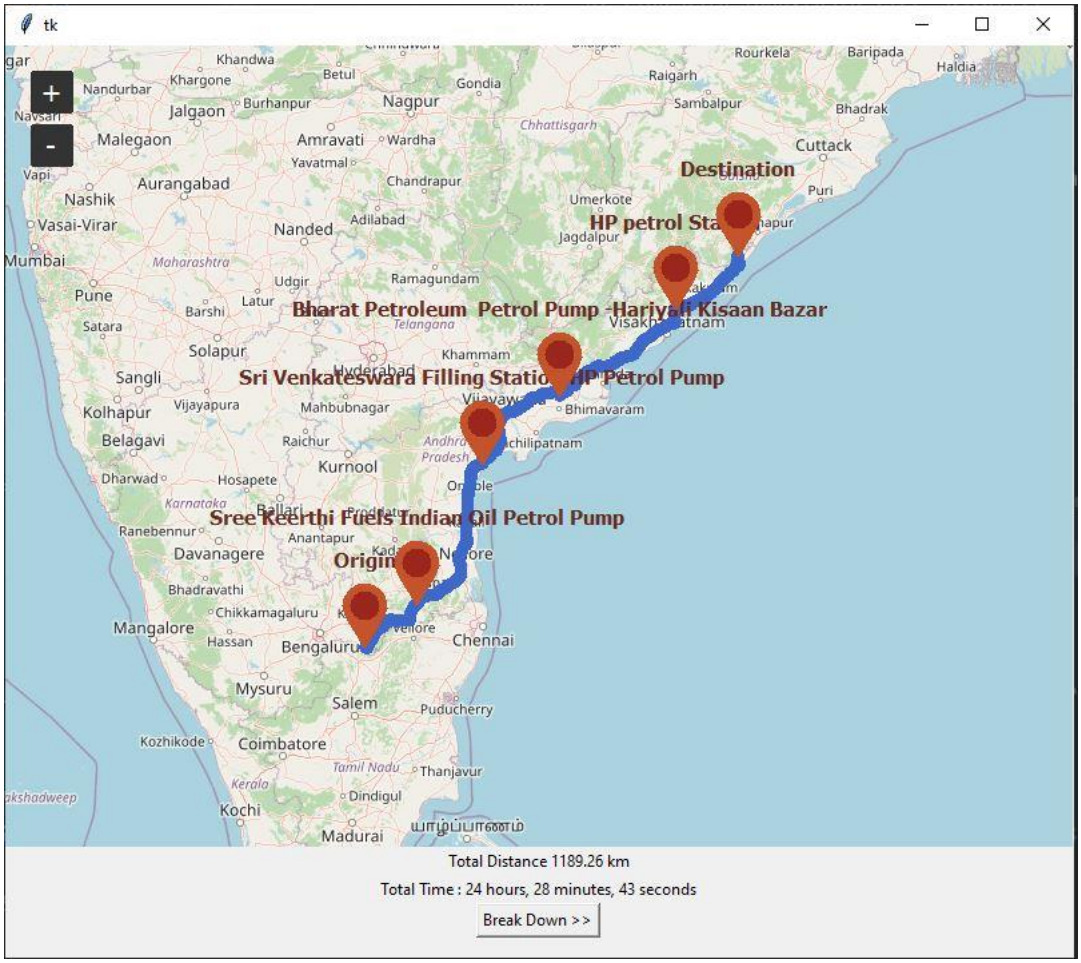
	A	B	C	D	E
1	origin	destinatic	distance	state	
2	[12.9908, '	[14.65928,	264648	Andhra Pradesh	
3	[14.65928,	[14.66431,	589	Andhra Pradesh	
4	[14.66431,	[14.66534,	134	Andhra Pradesh	
5	[14.66534,	[14.68353,	2473	Andhra Pradesh	
6	[14.68353,	[14.68776,	1718	Andhra Pradesh	
7	[14.68776,	[14.68607,	2389	Andhra Pradesh	
8	[14.68607,	[14.68078,	654	Andhra Pradesh	
9	[14.68078,	[14.67382,	2009	Andhra Pradesh	
10	[14.67382,	[14.65005,	4151	Andhra Pradesh	
11	[14.65005,	[14.66711,	2139	Andhra Pradesh	
12	[14.66711,	[14.67872,	1397	Andhra Pradesh	
13	[14.67872,	[14.68995,	1368	Andhra Pradesh	
14	[14.68995,	[14.6868, '	456	Andhra Pradesh	
15	[14.6868, '	[14.70015,	2972	Andhra Pradesh	
16	[14.70015,	[14.68708,	2728	Andhra Pradesh	
17	[14.68708,	[14.6775, '	1218	Andhra Pradesh	
18	[14.6775, '	[14.65685,	2901	Andhra Pradesh	
19	[14.65685,	[14.64501,	1974	Andhra Pradesh	
20	[14.64501,	[13.2134, '	261489	Andhra Pradesh	
21	[13.2134, '	[13.21587,	1873	Andhra Pradesh	
22	[13.21587,	[13.21953,	1068	Andhra Pradesh	
23	[13.21953,	[13.22305,	697	Andhra Pradesh	
24	[13.22305,	[13.22346,	305	Andhra Pradesh	
25	[13.22346,	[13.40621,	63941	Andhra Pradesh	
26	[13.40621,	[16.72519,	580713	Andhra Pradesh	
27	[16.72519,	[16.74242,	2953	Andhra Pradesh	

Finding EV vehicle route

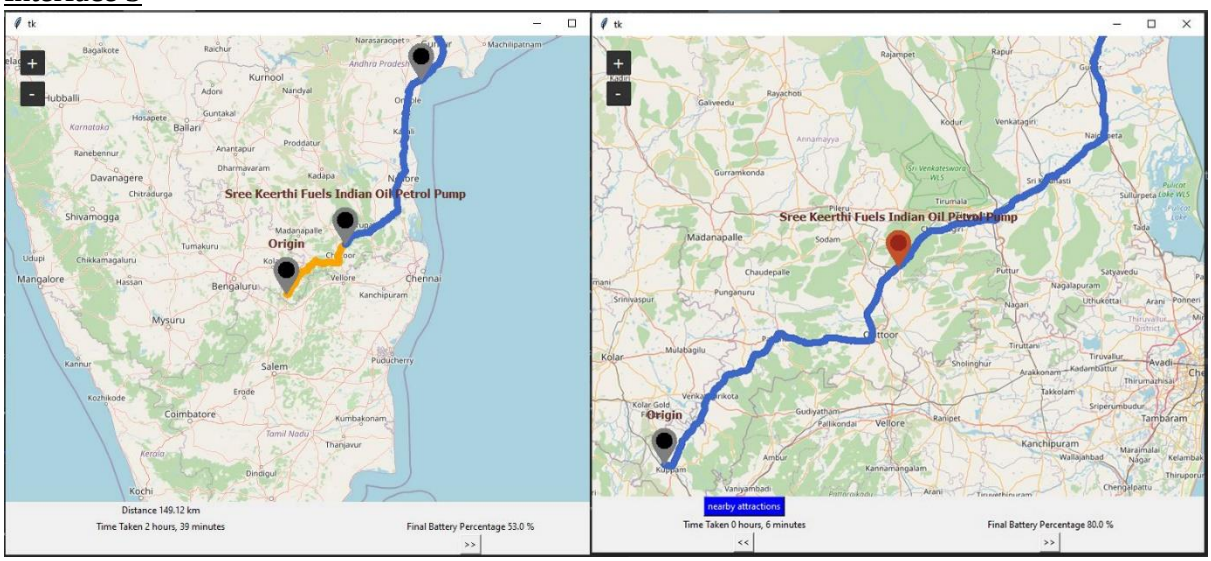
Interface 1



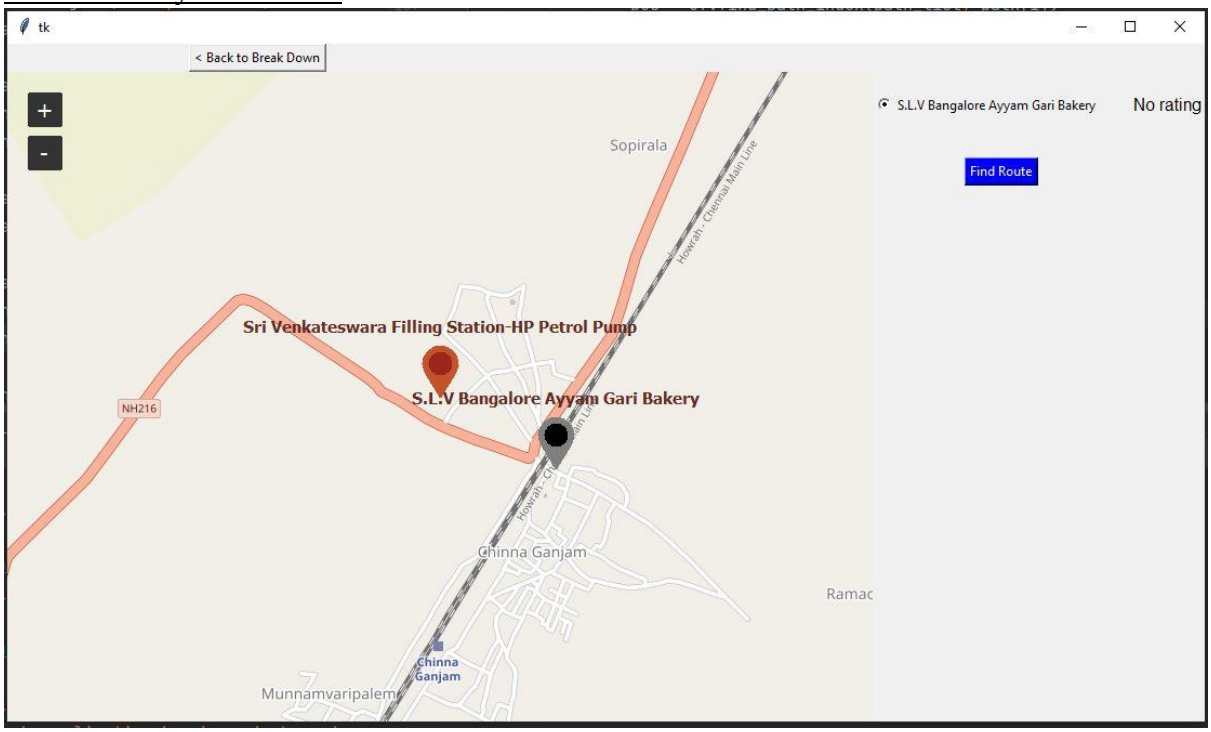
Interface 2



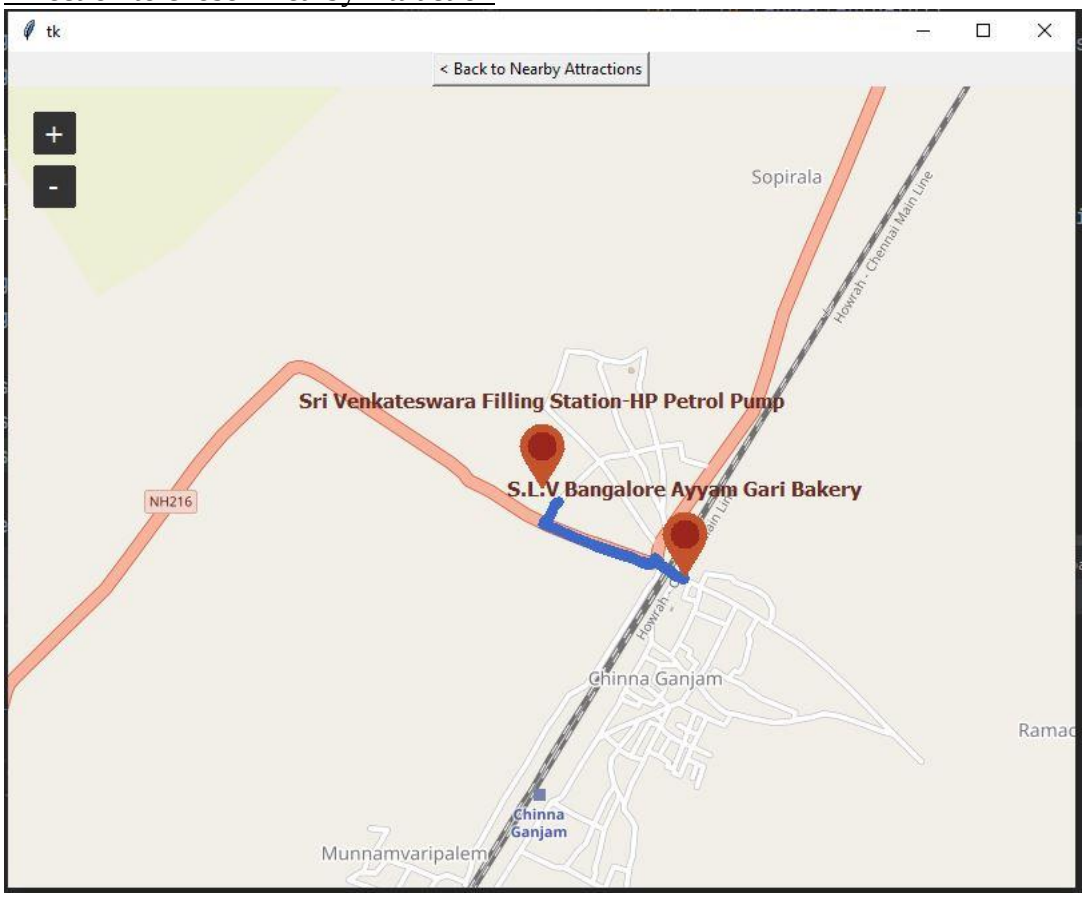
Interface 3



List of Nearby Attractions



Direction to chosen Nearby Attraction



Result and Discussion

The divide and conquer algorithm breaks down a long list of coordinates into solvable smaller segments. In this case, each district is a solvable segment. Optimal paths of districts can be combined to form the optimal path of a state. The optimal paths of each state can then be combined to form the whole country. The benefit of this approach is that it can be solved part by part, and solving the problem can resumed after a break in the implementation. Due to less complexity, this approach also allows us to use the google maps API for solving the travelling salesman problem with road distances.

However, the accuracy of this approach depends on the clustering of points. Incorrect clustering can lead to more deviation from the most optimal path. Using road distances using google maps API takes more time than obtaining straight line distances.

Every EV vehicle has different performance metrics. Given an origin and destination, the number of EV charging stops required may be different based on the model of the vehicle. Finding an EV vehicle route involved creating a car object, extracting coordinates from csv files, and using APIs to display a route from an origin to destination.

Summary

With petrol prices increasing on a day to day basis, people are considering to buy EV vehicles. However, high cost of EV vehicles, less availability of EV charging stations and charge time discourage many to make the transition.

A concern people have when buying an EV vehicle is the mileage and the availability of EV charging stations. Showing that a long-distance route between any origin and destination can reassure people who plan to shift to EV technologies. The availability of DC current fast charging makes it possible to travel long distance trips with EV vehicles.

A long-distance trip in an EV vehicle will be cheaper than in a petrol/diesel vehicle, however it will take more time.

References

Weblinks:

1. https://developers.google.com/maps/documentation/distance-matrix/usage-and-billing
2. https://developers.google.com/maps/documentation/distance-matrix/get-api-key
3. http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm
4. http://www.math.uwaterloo.ca/tsp/world/countries.html
5. https://en.wikiversity.org/wiki/Geographic_coordinate_conversion
6. https://www.carmagazine.co.uk/electric/how-long-does-it-take-to-charge-electric-car/#:~:text=Calculating%20the%20charge%20time%20is,up%20in%20under%20two%20hours
7. https://www.selectcarleasing.co.uk/hybrid-electric-cars/guides/miles-per-kwh

Journal:

1. B. Hu and G. R. Raidl, "Solving the Railway Traveling Salesman Problem via a Transformation into the Classical Traveling Salesman Problem," 2008 Eighth International Conference on Hybrid Intelligent Systems, 2008, pp. 73-77, doi: 10.1109/HIS.2008.30.

2. Papadimitriou, C. H. (1977). The Euclidean travelling salesman problem is NP-complete. Theoretical computer science, 4(3), 237-244.

3. Karthy, T., & Priyanka, S. (2019, June). Comparative analysis of the optimal solutions of travelling salesman problem. In AIP Conference Proceedings (Vol. 2112, No. 1, p. 020030). AIP Publishing LLC.

4. Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. Journal of the ACM (JACM), 9(1), 61-63.

5. M. H. Erol and F. Bulut, "Real-time application of travelling salesman problem using Google Maps API," 2017 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), 2017, pp. 1-5, doi: 10.1109/EBBT.2017.7956764.

Book:

1. Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2011). *The traveling salesman problem*. Princeton university press.