eMaps

Fachpraktikum Algorithms on OpenStreetMap Data 19/20

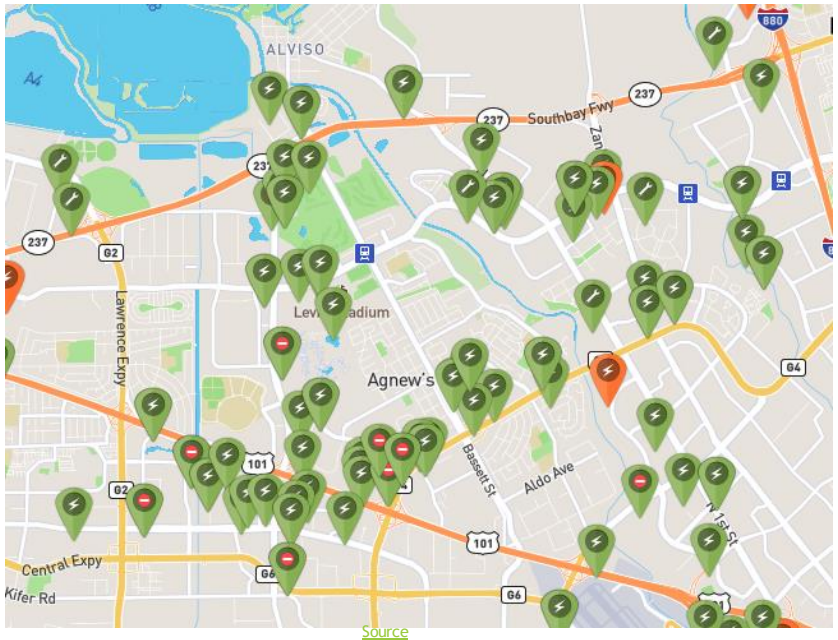
Motivation

Source

- Electrically-powered vehicles important in fight against climate change
- Unique characteristics:
  - Limited cruising range
  - Long recharge times
- May run out of power

→ Adaption of route planners required!

Idea

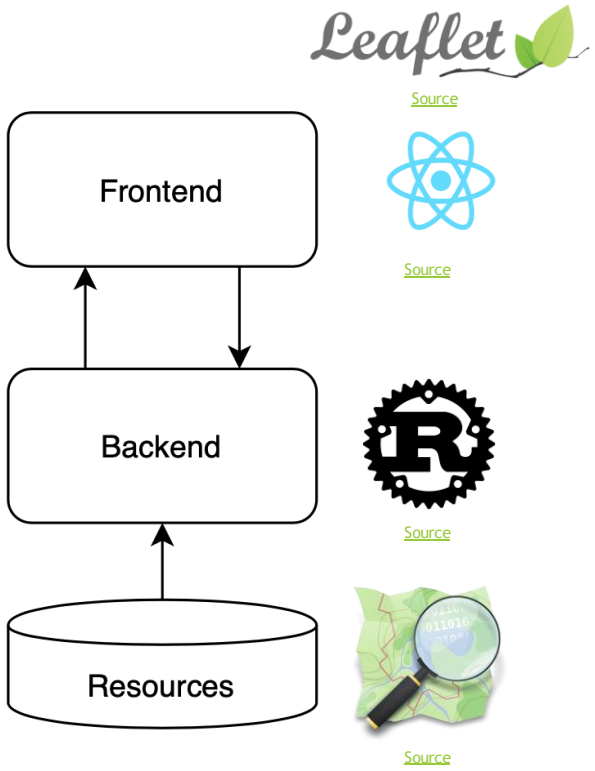


Source

- Route planner for e-Vehicles
- Route planner should consider:
  - Current and maximum range of e-Vehicles
  - Availability of charging stations
  - Never running out of power

Live Demo

Architecture



- Resources: Raw OpenStreetMap data in PBF format
- Backend: core functionality and API written in Rust
- Frontend: display map and routes using React and Leaflet

Key Concepts

- Parse amenities from OpenStreetMap data with {amenity: charging_station}
  - Parse vehicle supported by charging station, e.g. only Cars, only Bikes, or both

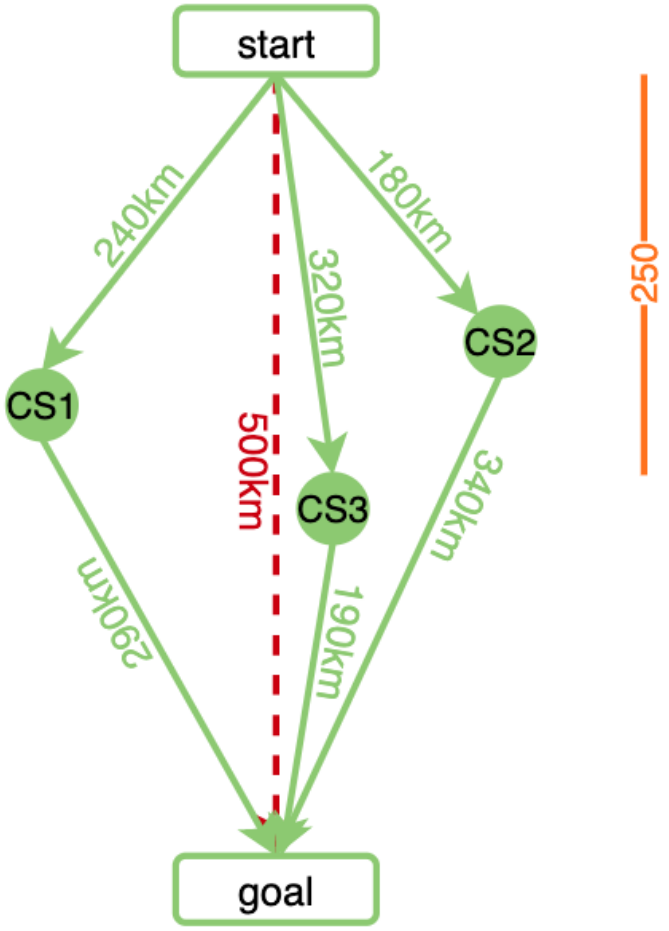
```
{
    "amenity": "charging_station",
    "authentication:none": "no",
    "car": "yes",
    "bicycle": "yes",
    "capacity": "2",
    "operator": "HellensteinStrom"
    ...
}
```

- Extend graph with charging station nodes

Key Concepts

1. Backend receives routing request from Frontend including current and maximum range of electric vehicle
2. Initial Dijkstra calculation to check if charging is required at all
   - No: return calculated dijkstra
   - Yes: go to **3**
3. Identify "best" charging station related to start and goal and current range
4. Calculate Dijkstra from start to identified charging station
5. Set current range = maximum range and start = charging station
6. Calculate Dijkstra from charging station to goal to check if further charging is required:
   - No: concatenate route and return
   - Yes: continue with **3**

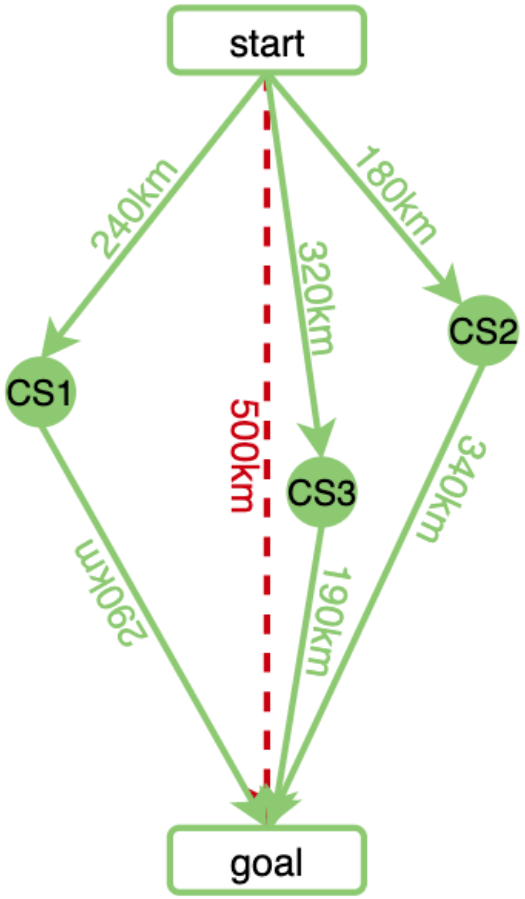
Key Concepts



For each charging station:

1. Check if haversine distance from start to charging station is within current range with treshold
   - Yes: go to 2
   - No: go to next charging station
2. Check if charging station utilizes at least 50% of current range
   1. Yes: go to 3
   2. No: go to next charging station
3. Calculate haversine distance from charging station to goal
4. Check if sum of distances is smaller than current best sum
   1. Yes: update currently best charging station
   2. No: go to next charging station

Key Concepts



250

- CS1:
  1. Is in current range: 240 < 250 ➡ true
  2. Utilizes at least 50% of range: 240 => 250*0.5 ➡ true
  3. Sum of distances smaller than current best sum: 290 + 240 < MAX ➡ true ➡ set as currently best charging station
- CS2:
  1. Is in current range: 320 < 250 ➡ false ➡ go to CS3
- CS3:
  1. Is in current range: 180 < 250 ➡ true
  2. Utilizes at least 50% of range: 180 => 250*0.5 ➡ true
  3. Sum of distances smaller than current best sum: 180 + 340 < 530 ➡ true ➡ set as currently best charging station

Other Features

- Search Cities, Places, POIs, … via Nominatim API
- Show map of all charging stations
- Time/distance routing
- Routing for eBike and eCar

Limitations & Future Work

- Determining charging station/route not optimal
  - Calculate dijkstra for each charging station ➡ inefficient
  - Extend edges by a weight representing the energy consumption
  - Consider elevation profile to determine more energy efficient routes

- Thank you!