NEA

Optimised Safe Route Finder

Last updated March 16, 2022
Pages: 8

Contents

1 Analysis **3**
1.1 Problem Area . . . 3
1.2 Client . . . 3
1.3 Similar Systems . . . 5
1.4 Features . . . 5
1.4.1 Map data . . . 5
1.4.2 Accident Data . . . 5
1.4.3 Traffic Data . . . 6
1.4.4 Finding the shortest route . . . 7
1.5 Critical Path . . . 7
1.6 Specification . . . 7

2 Design **9**
2.1 Accident Download System . . . 9
2.1.1 Pulling from the TFL API . . . 9
2.1.2 Parsing the Data . . . 9

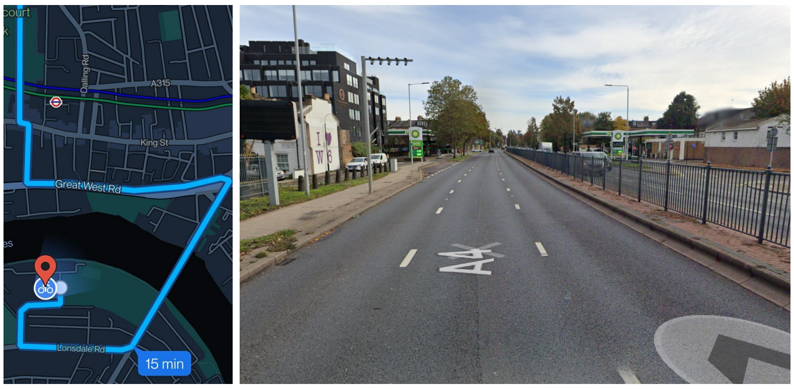

Figure 1: Route suggested by google maps

1 Analysis

1.1 Problem Area

There exist a plethora of route finding services for finding the shortest route between two points. These typically optimise for the shortest time taken to travel between two points. Provision for cyclists can be lacking, because it typically consists of an alteration to the existing code for cars, slightly modified to allow for cycle paths and different timings for cyclists.

Many of these route finding applications don't take into account safety considerations. This is shown in Figure 1 which shows the route Google Maps suggests that I cycle to school by. The route includes the A4, which is a very dangerous stretch of road for cyclists as it is a 3 lane road. My plan is to use accident statistics to work out which roads are dangerous and then avoid them. This would be done by imposing a cost for going somewhere that there had been an accident.

1.2 Client

I interviewed Yuvraj Dubey, one of my classmates, to talk about whether they would be interererested in a product that attempted to calculate safe routes to cycle.

Do you own a bicycle?
Yes
Do you cycle regularly?
Yes
Do you ever feel in danger while cycling?
Yes
When do you feel in most danger while cycling?
At a Junction near Victoria, where there is a right turn and no cycle lane to do it in. I attempted to go this way once and was almost hit by a bus.
Are you aware of your surroundings while cycling?
Not really
Do you feel aware of the places which to cycle in safely, does this change when you are in places you commonly go?
No, but especially where I don't know where I am because then I do not know where it is safe to cycle. When I cycle home I have learnt a safe route, but if asked to cycle somewhere I did not know I would most likely end up in a dangerous place.
Would you use an application that tells you safe routes to get places?

Yes
Would you be happy using a command line application for this?
Yes
If said application took more than 60 seconds to calculate a route would you lose patience?
Yes, probably
Have you found that cycling directions generated by current products are safe and efficient?
They are definitely efficient, but they often take me onto busy roads and junctions which can be less safe.

From this interview and my own experiences cycling in London I determined that there was an application for my idea, but only if it ran in a reasonable amount of time and was otherwise easy to use, as people struggle to find safe routes themselves, especially when going somewhere they had not previously been, and commercial products such as Google Maps don’t generate safe routes.

1.3 Similar Systems

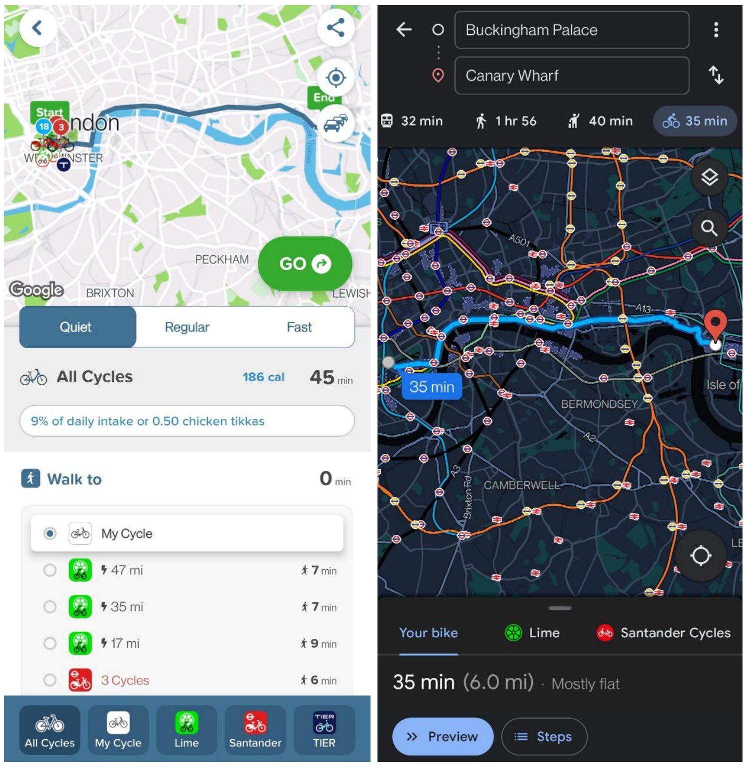

Figure 2: Citymapper and Google Maps respectively, generating the same directions

There exist similar systems for route finding such as Google Maps and Citymapper. These have advantage's over what I will be able to offer such as being able to use live traffic data, and having a good user interface. Citymapper has a option to choose between "Quiet", "Regular", and "Fast" routes, which seems to be based on the type of road you are taken down. However they don't really take accident density into account which will be my aims.

1.4 Features

1.4.1 Map data

I need a data source which can provide data on roads that are legal to cycle on as well as being freely available for me to use. I settled on Open Street Map, a project which combines data gathered by volunteers into one massive freely available map. The map is downloadable in the form of a large XML file or PBF file. A PBF file is just a binary version of the same thing. I could write code to parse this myself, but instead I think that I am going to use the library libosmium which works in C++.

1.4.2 Accident Data

Transport for London has an excellent API which you can download accident data from. The data comes in JSON files and contains information about what type of vehicle was involved in each accident, the severity, and the coordinates of the accident. I need to find a way of mapping the coordinates onto the road network so that the danger of roads can be calculated as accurately

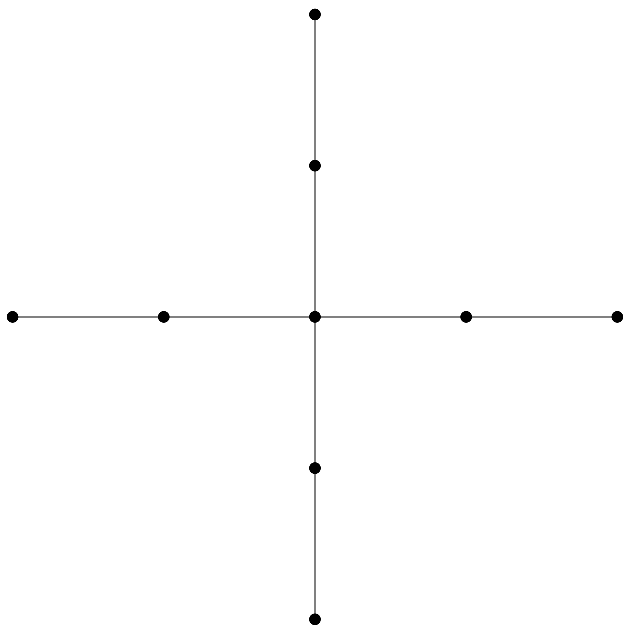

Figure 3: Typical OSM representation of an intersection

as possible. Originally I was looking at adding the accident to the nearest edge in the graph but I decided against this after considering how intersections are typically represented in OSM. Road intersections typically look something like what is represented in Figure 3. If an accident were to occur at the intersection it would end up being added to one of the segments leading into the intersection, so the danger would not be properly calculated if not passing through that segment. Instead I thought about adding the danger to the nearest node, so that the accident would always be counted when a route passes through that intersection. The other problem that I have to deal with, if adding data to the nearest node is that the density of nodes is not constant. Straighter roads will not need to use as many OSM nodes as curved roads, so i might incorrectly add cost to the wrong node. My proposed solution to this is to interpolate in nodes to a very high density to deal with this problem.

1.4.3 Traffic Data

The main problem with this is that the accident data is absolute and can thus not be used to calculate probabilities. For example, more accidents happen on King's Street than the dangerous road I showed earlier, but this doesn't mean that King's Street is more dangerous merely that more cyclists travel on it. This means that I need to get accurate cyclist traffic data for the whole of London in order to turn my accident statistics into accident probabilities. The Department for Transport and the Office for National Statistics both keep data on traffic, but it isn't applicable because cycle data is only given as a total [1] and at specific count points. This means that I will either need to work out traffic data or get it from some dataset, such as Strava's Global Heatmap. Luckily the rest of the application can work without including traffic data so my plan is to deal with this problem at a later time or hope that the avoidance of accidents alone will be enough.

[1] 0.6 billion miles per year in London

1.4.4 Finding the shortest route

Algorithms for finding the shortest path in a Graph are abundant. The most well known is Dijkstra's algorithm, and it's variant A*. In large Graphs, both Dijkstra's algorithm and A* can be very slow. I will most likely use preprocessing based algorithms such as ALT*/Contraction Hierachies to make my program run a lot faster. This preprocessing will take a long time, but when completed it will significantly reduce the time taken to complete searches.

1.5 Critical Path

- Download accidents from the TFL API, parse for accidents that are of interest, and store it in a format that can be easily parsed later
- Import OSM map of London as a Graph which can be easily queried.
- Write code that imports the accident data and connects it to given Graph nodes.
- Write a simple Dijkstra's Algorithm method of finding the shortest path.
- Write a preprocessing based method of finding the shortest path.
- Write a frontend for inputing data about the route.

1.6 Specification

1. Accident Download System
   - 1.1 The System must be capable of interfacing with the TFL API to download accidents
   - 1.2 The System must be capable of parsing accident data to determine which accidents are relevant
   - 1.3 The System must be capable of storing accident data in a easily accessible file for the route finding algorithm to parse
2. Route Finding Systems
   - 2.1 The System should be capable of processing an Open Street Map map file into a suitable data structure
   - 2.2 The System should be capable of processing the accident data from the Accident Download System
   - 2.3 The System should be capable of attaching the data from the Accident Download System to the Suitable Data Structure from specification point **2.1**
   - 2.4 The System should be capable of using Accident Data combined with other suitable cost estimation functions to find a directed route between two points that are places on roads in London
   - 2.5 The System should always suggest a route that can be followed while observing all currently known laws of physics
   - 2.6 The System should always suggest a route that can be followed while observing all international, national, and local laws
   - 2.7 The System should suggest a safe route wherever possible
   - 2.8 The System should have guards in place for certain types of roads which are deemed too dangerous to consider

2.9 The System should be able to generate this route quickly

2.10 The System should have a mode for preprocessing to make the route generation even quicker

2.11 When using the same parameters, the route generated using a preprocessed graph and associated algorithms should be the same as that generated by the non-preprocessing based approach

2.12 The System should be easy to use

2.13 The System should not crash, and should give appropriate non crashing errors if user data is determined to be bad

2.14 The parameters for the cost estimation functions should be user definable, but sensible defaults should also be defined

```
[{
    "id": 0,
    "lat": 0.0,
    "lon": 0.0,
    "location": "string",
    "date": "string",
    "severity": "string",
    "borough": "string",
    "casualties": [{
        "age": 0,
        "class": "string",
        "severity": "string",
        "mode": "string",
        "ageBand": "string"
    }],
    "vehicles": [{
        "type": "string"
    }]
}]

```

Figure 4: The Default output from the API

2 Design

2.1 Accident Download System

As outlined in the Specification, this system should be capable of interfacing with the TFL API, downloading the accidents, deciding which ones are relevant, and storing this in a useful format.

2.1.1 Pulling from the TFL API

The Transport For London API is an excellent API which can provide information on many different aspects of the Transport For London network. The specific API which I used is the AccidentStats API. This API is very simple, you simply request a given year and a JSON object is returned which contains all of the accidents that happened in london that year. These consist of all the accidents that were reported tocdepth the police as happening in that year. Of course, there will be many more accidents than are on the API, but these will mostly be more minor accidents. The Data is returned as a list of accidents, formatted as shown in Figure 4.
As the data about accidents that happened in the past is not going to change any time soon, and TFL only updates this API every year, it is simplest just to download the files once, parse it once, and then use that result in the route finder.
TFL started gathering this data in 2005, and the most recent update was in 2019, so I wrote a simple python script to download all the data from 2005 to 2019 and save it in a subfolder called `accidents`. The program loops through all the years between 2005 and 2019 and fills a JSON file for that year.

2.1.2 Parsing the Data

The next step was to go through the data from all the years, and save all the accidents that were pertinent to my project. As seen in Figure 4, each accident listing has information on casualties, of which there may be many. As this is a cycling application, I only wanted data on accidents where

```python
import json
accidents = []
for year in range(2005,2020):
    print(len(accidents), year)
    with open(f"accidents/{year}.json","r") as outfile:
        data = json.load(outfile)
    for x in data:
        if len([person for person in x["casualties"] if person["mode"] == "PedalCycle"
    ]) >= 1:
            accidents.append([x["lat"],x["lon"],x["severity"]])
with open("output.json","w") as outfile:
    json.dump(accidents,outfile)

```

Figure 5: The script I wrote to parse the data

at least one of the casualties was a cyclist. The API provides a lot of data, but all that I decided was relevant was the latitude and longitude, as well as the severity of the incident. All of this data was ultimately saved to a file called `output.json`. The code used for parsing is shown in Figure 5.