University of Tehran

Department of Electrical and Computer Engineering

Optimal Routing

Operations Research – Final Project

Erfan Bagheri Soula

Table of content

Dataset Generation and Visualization

Data Extraction

For generating the dataset, 10 locations in district 3 and district 6 were chosen, aiming to cover all the area properly. Then, all properties mentioned in the project manual were extracted for each of these locations. The weight of an edge between a node (location) and its neighbor is simply the time (in minutes) estimated by Google Maps to drive from that node to the corresponding neighbor with a car.

All the extracted data is stored in a JSON file in the same directory as python script.

Model Graph

The following is a graph representing the model we extracted from the map.

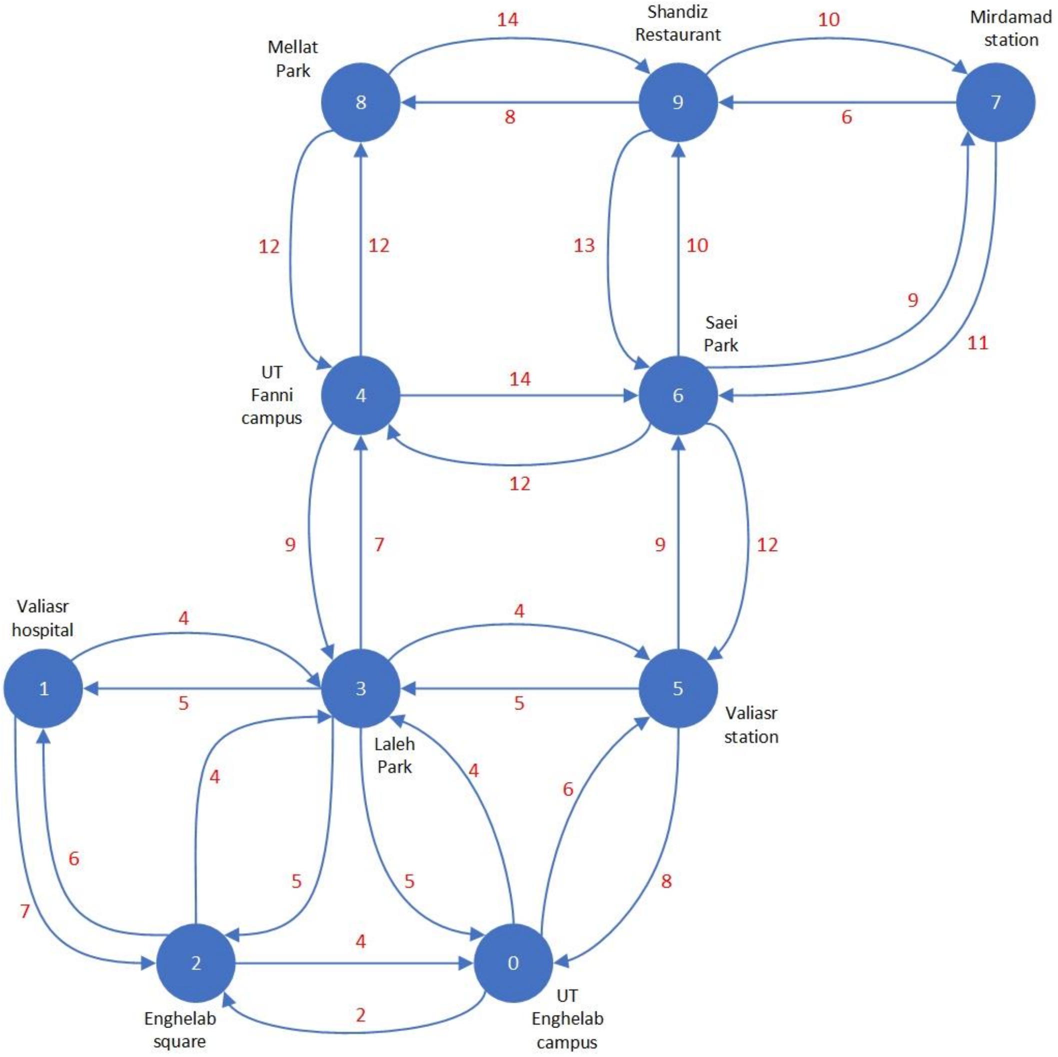


Figure 1: A graph representing the modeled map

Minimum Cost Routing

To find the shortest path between two nodes of this model graph, we just need to run a UCS (uniform cost search) algorithm with starting point as the root and the destination as the goal. Below, we use a technique to transform this search to a linear programming problem and then simply use the tools we have to quickly solve this kind of problems.

Transforming to a Linear Programming Problem

Given a directed graph (V, A) with starting node s, target node t, and cost $W_{i,j}$ for each edge (i, j) in A, we can use the natural linear programming formulation written below for the shortest path problem in this graph and transform it to a discrete optimization problem.

Consider a program with binary variables $X_{i,j}$

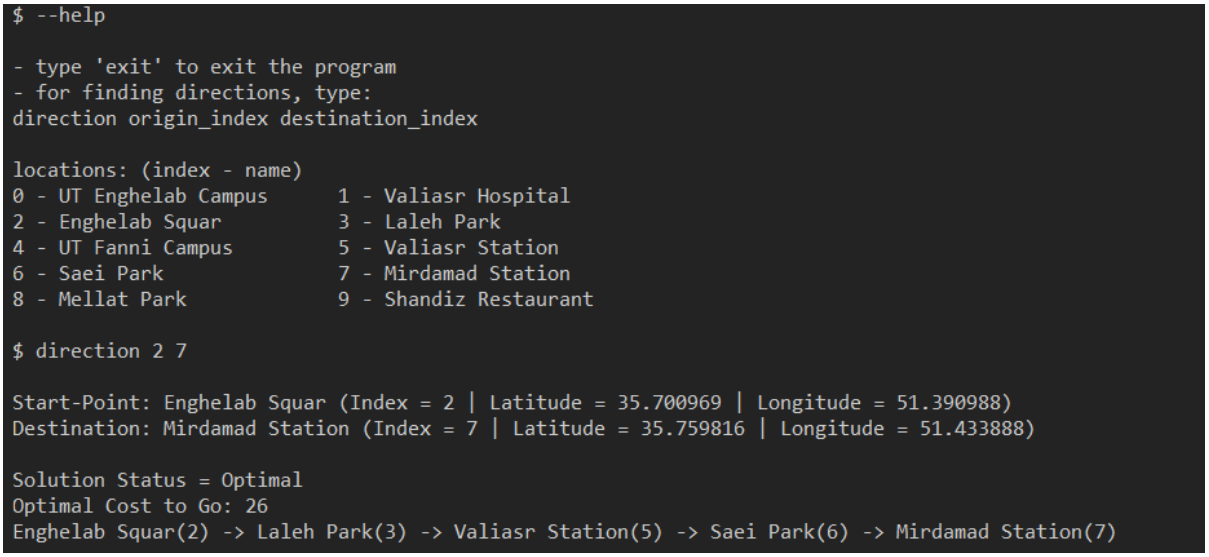
$$Minimize \sum_{i,j \epsilon A} W_{i,j} X_{i,j}$$

$$Subject\ to: \forall i \ \rightarrow \sum_{j} X_{i,j} - \sum_{j} X_{j,i} = \begin{cases} 1\ (if\ i = s) \\ -1\ (if\ i = t) \\ 0\ (otherwise) \end{cases}$$

The intuition behind this is that $X_{i,j}$ is an indicator variable for whether edge (i, j) is part of the shortest path (when it is 1) or not (when it is 0).

Computer Program

To automate this process, a python program is written that uses PULP package for solving the linear optimization problem. A simple console interface is also provided. The program reports the optimal cost and solution after it solves the problem in the console environment. One can type “--help” command to get more information on how to use the console interface.



```
$ --help

- type 'exit' to exit the program
- for finding directions, type:
direction origin_index destination_index

locations: (index - name)
0 - UT Enghelab Campus      1 - Valiasr Hospital
2 - Enghelab Squar          3 - Laleh Park
4 - UT Fanni Campus         5 - Valiasr Station
6 - Saei Park               7 - Mirdamad Station
8 - Mellat Park             9 - Shandiz Restaurant

$ direction 2 7

Start-Point: Enghelab Squar (Index = 2 | Latitude = 35.700969 | Longitude = 51.390988)
Destination: Mirdamad Station (Index = 7 | Latitude = 35.759816 | Longitude = 51.433888)

Solution Status = Optimal
Optimal Cost to Go: 26
Enghelab Squar(2) -> Laleh Park(3) -> Valiasr Station(5) -> Saei Park(6) -> Mirdamad Station(7)
```

Figure 2: Example on how to use the program