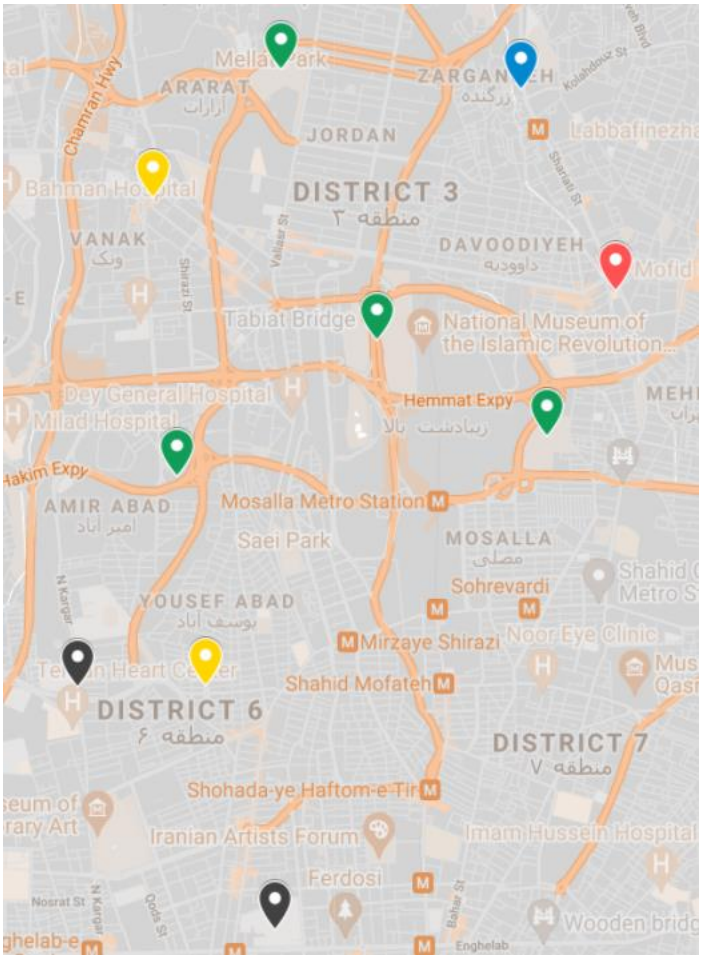
OR Final Project Report: Optimal Vehicle Routing

Mahdiyar Ali Akbar Alavi

810196513

Part 1.

In this project, the main goal is to design a real-world optimization problem (i.e., The Shortest Path Problem), then build an LP model for it and afterwards try to solve that by using Linear Programming libraries in Python. In the first part, we use Google Maps new service called “My Maps” in order to select 10 places from two districts of Tehran, i.e., district 3 and 6. The selected places are distinguished by 10 colorful pins in the following picture:



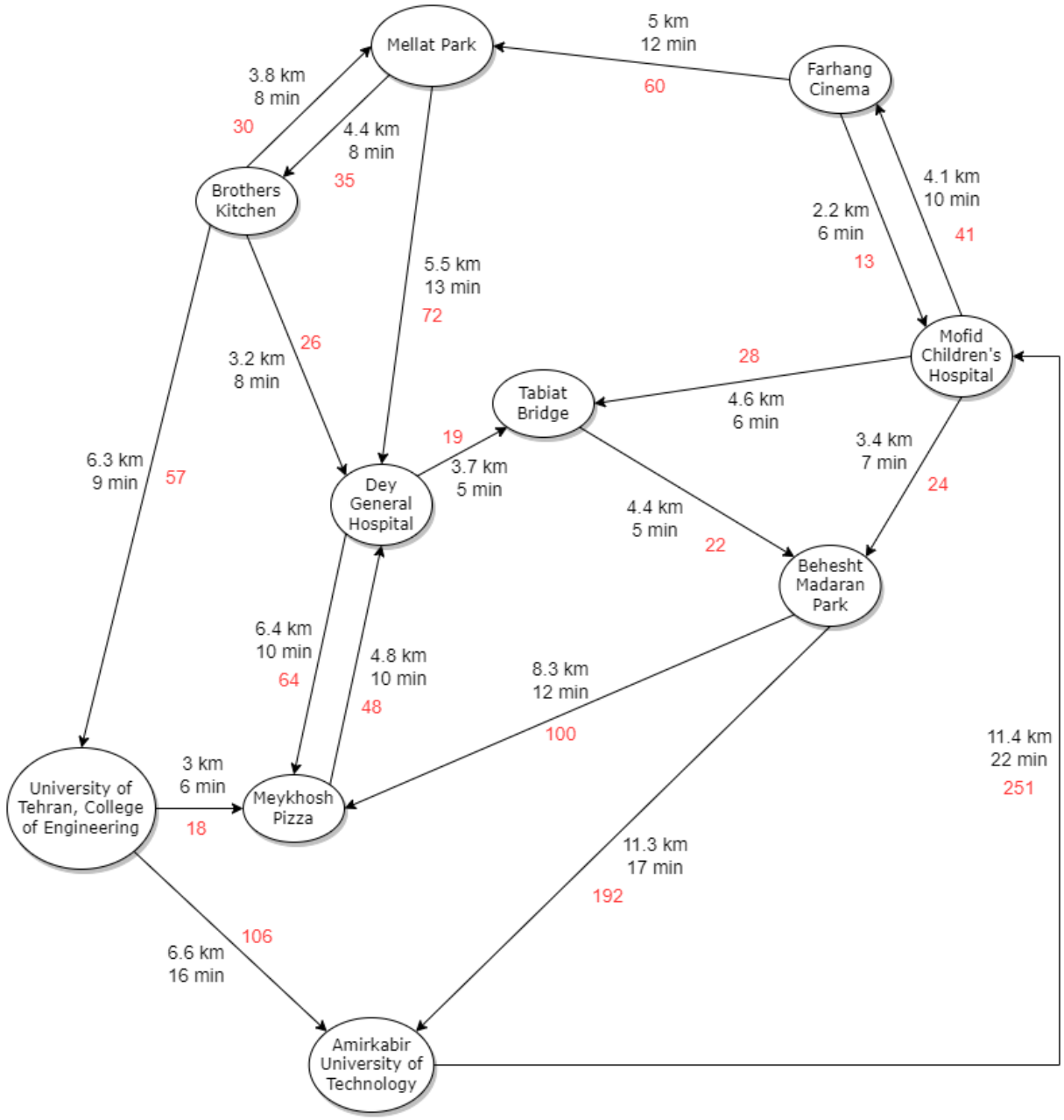
Picture 1. 10 places are shown by means of colorful pins.

We might also want to take a look at these places in detail:

0. Mofid Children's Hospital
1. Behesht Madaran Park
2. Tabiat Bridge
3. Brothers Kitchen
4. Mellat Park
5. Amirkabir University of Technology
6. Meykhosh Pizza
7. University of Tehran, College of Engineering
8. Dey General Hospital
9. Farhang Cinema

A .csv file is also attached to this report in the uploaded files in which one can figure out the latitudes and longitudes of the aforementioned places. It is somehow obvious that the pins are covering districts 3 and 6 nicely and the required conditions are satisfied.

At this point, we have to arbitrarily pick neighbors for each and every place (i.e., node) to reach a neat digraph as shown in the upcoming picture:



Picture 2. A neat digraph with precise costs determined by multiplying distance by the elapsed time to travel from one place to another

As you can see, each place is modeled with a node in the digraph and every node has some incoming and outgoing edges which indicate its paths to/from other places (i.e., nodes). Each path is also equipped with an integer which is the distance of that path multiplied by the elapsed time to travel from its origin to its destination; we call this integer the path's cost. Our main goal in the following part is to find a set of paths with the minimum sum of costs in order to travel from one node to another.

Part 2.

In this part, we aim to solve the shortest path problem by using LP libraries in Python. Our final goal is to write a simple script which takes two arbitrary places (i.e., node indices) and reads a .csv file containing the adequate info to build the digraph of picture 2. Then it uses the PuLP library in Python to solve the problem and output this info:

1. Details on the origin node and the destination one
2. The indices of all nodes in the optimal path
3. The total cost of traveling from the origin node to the destination one

A simple shortest path problem could be modeled with the following objective function and constraints:

$$minimize \sum_{u \to v} l_{u \to v} \cdot x_{u \to v}$$

$$subject\ to \sum_{u} x_{u \to s} - \sum_{w} x_{s \to w} = -1$$

$$\sum_{u} x_{u \to t} - \sum_{w} x_{t \to w} = 1$$

$$\sum_{u} x_{u \to v} - \sum_{w} x_{v \to w} = 0; \forall\ node \neq s, t$$

$$x_{u \to v} \in \{0,1\}; \forall\ edge\ u \to v$$

in which the origin node is indicated by s, the destination one is shown by t and a path going from a random node v to w is mentioned as $v \rightarrow w$.

In the aforementioned model, each x variable can be either 0 or 1; actually, if $x_{v \rightarrow w}$ is equal to 0, it means that the path $v \rightarrow w$ is not lied on the optimal path from s to t. Vividly, if $x_{v \rightarrow w} = 1$, the corresponding path (i.e., $x_{v \rightarrow w}$) is used in order to reach the node t from the node s. Speaking of the x variables, the objective function is built based upon them. As a matter of fact, we aim to minimize the summation of the costs of the used paths (i.e., their corresponding x variable is equal to 1), thus we take the vector multiplication $L^T X$ as our objective function in which the L vector contains the cost of each corresponding path and the X vector contains all of the x variables; this could also be shown as $\sum_{u \rightarrow v} l_{u \rightarrow v} \cdot x_{u \rightarrow v}$.

The constraints of this problem are also defined as follows:

1. For each node, except the origin and the destination ones, the total incoming edges count must meet the total outgoing edges count, as we cannot go to one node and stay there, or even more oddly leave a node without even entering it! So, we have the 3rd constraint to hold this condition.
2. As we will leave the s node finally, the number of its outgoing edges must be less than the number of its incoming edges by one. So, we have the 1st constraint as well.
3. The last but not least, is the $\sum_u x_{u \rightarrow t} - \sum_w x_{t \rightarrow w} = 1$ constraint which holds the condition of reaching t and staying there forever.

Other constraints are intuitively declared as we know we cannot use each path more than once or less than never.

P.S. Script file is attached to this report and will be uploaded on ELearn.