

Traffic flow optimization using D-wave quantum computer

Purpose:

The aim of the project is the optimization of the traffic flow by minimizing the time for a set of cars to travel between their individual sources and destinations by minimizing total congestion in overall road segments. To do so, we minimize traffic flow by redirecting a subset of the cars to alternative routes in a way that the amount of intersecting road segments is minimized. Thus, it is essential to optimize overall cars simultaneously. In other words, any redistribution of cars that resolves the original congestion should not cause a traffic jam in other places on the map. By using a quadratic function of the number of cars traversing it in a specific time interval, the congestion on an individual segment can be determined. We use a hybrid quantum and classical approach to solve this problem.

Workflow:

1. Classical: create a random graph with the desired size using the networkx library in python.

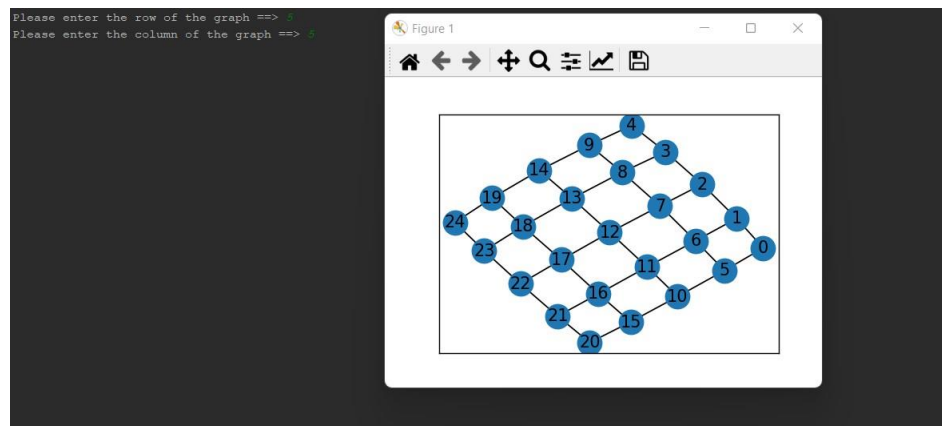


Figure 1: create a graph with the desired size (in this example is 5× 5)

2. Classical: add a weight to each edge (links) of this graph (for the instance speed limit, length of the edges, traffic weight). By having this information, we could calculate travel time on this edge.

```

The avenue (0, 1) : permissible speed is 90 km/h **** length is 96 km **** traffic weight is 7 **** it takes time 91.0 minutes
The avenue (0, 5) : permissible speed is 90 km/h **** length is 48 km **** traffic weight is 17 **** it takes time 160.0 minutes
The avenue (1, 2) : permissible speed is 80 km/h **** length is 62 km **** traffic weight is 17 **** it takes time 232.0 minutes
The avenue (1, 6) : permissible speed is 80 km/h **** length is 37 km **** traffic weight is 12 **** it takes time 61.0 minutes
The avenue (2, 3) : permissible speed is 90 km/h **** length is 53 km **** traffic weight is 14 **** it takes time 100.0 minutes
The avenue (2, 7) : permissible speed is 100 km/h **** length is 36 km **** traffic weight is 17 **** it takes time 108.0 minutes
The avenue (3, 4) : permissible speed is 80 km/h **** length is 67 km **** traffic weight is 4 **** it takes time 59.0 minutes
The avenue (3, 8) : permissible speed is 110 km/h **** length is 99 km **** traffic weight is 4 **** it takes time 63.0 minutes
The avenue (4, 9) : permissible speed is 110 km/h **** length is 55 km **** traffic weight is 7 **** it takes time 42.0 minutes
The avenue (5, 6) : permissible speed is 100 km/h **** length is 70 km **** traffic weight is 16 **** it takes time 168.0 minutes
The avenue (5, 10) : permissible speed is 80 km/h **** length is 51 km **** traffic weight is 19 **** it takes time 382.0 minutes
The avenue (6, 7) : permissible speed is 80 km/h **** length is 53 km **** traffic weight is 15 **** it takes time 132.0 minutes
The avenue (6, 11) : permissible speed is 90 km/h **** length is 69 km **** traffic weight is 6 **** it takes time 61.0 minutes
The avenue (7, 8) : permissible speed is 120 km/h **** length is 21 km **** traffic weight is 11 **** it takes time 21.0 minutes
The avenue (7, 12) : permissible speed is 70 km/h **** length is 36 km **** traffic weight is 1 **** it takes time 30.0 minutes
The avenue (8, 9) : permissible speed is 40 km/h **** length is 87 km **** traffic weight is 14 **** it takes time 372.0 minutes
The avenue (8, 13) : permissible speed is 80 km/h **** length is 58 km **** traffic weight is 6 **** it takes time 58.0 minutes
The avenue (9, 14) : permissible speed is 120 km/h **** length is 39 km **** traffic weight is 2 **** it takes time 20.0 minutes
The avenue (10, 11) : permissible speed is 110 km/h **** length is 38 km **** traffic weight is 14 **** it takes time 59.0 minutes
The avenue (10, 15) : permissible speed is 90 km/h **** length is 82 km **** traffic weight is 6 **** it takes time 131.0 minutes
The avenue (11, 12) : permissible speed is 50 km/h **** length is 93 km **** traffic weight is 16 **** it takes time 446.0 minutes
The avenue (11, 16) : permissible speed is 40 km/h **** length is 42 km **** traffic weight is 2 **** it takes time 66.0 minutes
The avenue (12, 13) : permissible speed is 80 km/h **** length is 93 km **** traffic weight is 14 **** it takes time 195.0 minutes
The avenue (12, 17) : permissible speed is 120 km/h **** length is 88 km **** traffic weight is 3 **** it takes time 48.0 minutes
The avenue (13, 14) : permissible speed is 40 km/h **** length is 95 km **** traffic weight is 9 **** it takes time 237.0 minutes
The avenue (13, 18) : permissible speed is 100 km/h **** length is 40 km **** traffic weight is 1 **** it takes time 24.0 minutes
The avenue (14, 19) : permissible speed is 100 km/h **** length is 56 km **** traffic weight is 5 **** it takes time 42.0 minutes
The avenue (15, 16) : permissible speed is 90 km/h **** length is 22 km **** traffic weight is 19 **** it takes time 146.0 minutes
The avenue (15, 20) : permissible speed is 50 km/h **** length is 59 km **** traffic weight is 4 **** it takes time 83.0 minutes
The avenue (16, 17) : permissible speed is 110 km/h **** length is 63 km **** traffic weight is 9 **** it takes time 57.0 minutes
The avenue (16, 21) : permissible speed is 60 km/h **** length is 83 km **** traffic weight is 1 **** it takes time 83.0 minutes
The avenue (17, 18) : permissible speed is 110 km/h **** length is 69 km **** traffic weight is 2 **** it takes time 39.0 minutes
The avenue (17, 22) : permissible speed is 100 km/h **** length is 66 km **** traffic weight is 17 **** it takes time 198.0 minutes
The avenue (18, 19) : permissible speed is 70 km/h **** length is 54 km **** traffic weight is 1 **** it takes time 46.0 minutes
The avenue (18, 23) : permissible speed is 60 km/h **** length is 84 km **** traffic weight is 1 **** it takes time 84.0 minutes

```

Figure 2: add a weight to each edge (links) and calculate the travel time on this edge.

3. Classical: find 3 different routes from the origin node to the destination node with the lowest travel time

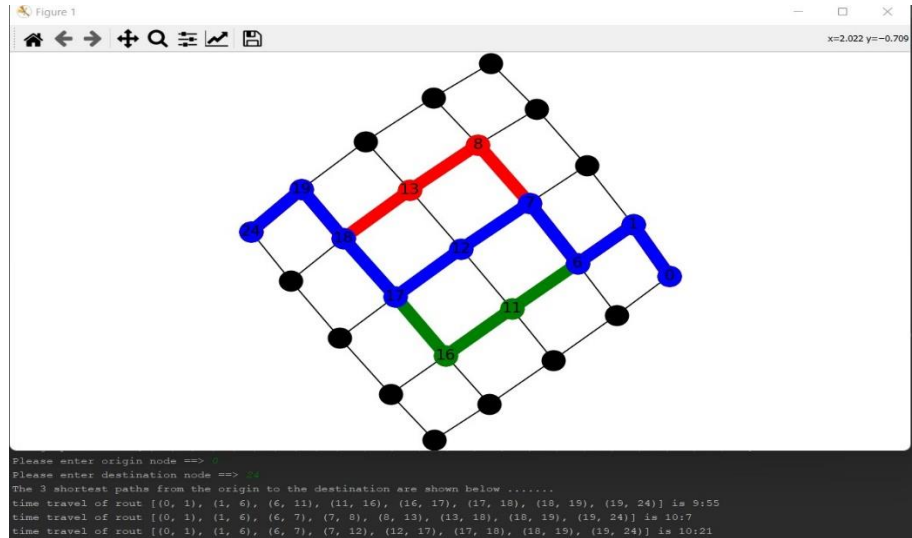


Figure 3: find 3 different routes from the origin node (0) to the destination node (24) with the lowest travel time.

4. Classical: in order to minimize congestion in road segments on overlapping routes, we formulate it as a QUBO
5. Hybrid Quantum/Classical: Find a solution that decreases congestion in three selected routes.
6. Classical: the cars are distributed based on the results in these 3 routes

```

please enter the number of cars ---> 6
time : 0.0
q11 q21 q32 q42 q53 q63 Energy: -1052.0 Occurrences: 1
6
q12 q23 q31 q42 q53 q61 Energy: -1052.0 Occurrences: 1
6
q11 q21 q32 q42 q53 q63 Energy: -1052.0 Occurrences: 1
6
q13 q22 q31 q41 q53 q62 Energy: -1052.0 Occurrences: 1
6
q13 q22 q31 q41 q52 q63 Energy: -1052.0 Occurrences: 1
6
q12 q23 q33 q42 q51 q61 Energy: -1052.0 Occurrences: 1
6
q11 q22 q33 q42 q51 q63 Energy: -1052.0 Occurrences: 1
6
q11 q22 q33 q43 q51 q62 Energy: -1052.0 Occurrences: 1
6
q13 q21 q33 q42 q51 q62 Energy: -1052.0 Occurrences: 1
6
q11 q22 q33 q43 q52 q61 Energy: -1052.0 Occurrences: 1
6
Process finished with exit code 0

```

Figure 4: Distribute cars (in this example number of cars is equal to 6) that want to travel from node 0 to node 24 based on the results in these 3 routes. In such a way that they create the least traffic for each other, in this case, we found 10 different scenarios. q11 means car number one takes route number one, q21 means car number two takes route number one, and so on.

Advantages of solving the traffic problem by using quantum computers compared to the current navigator applications:

As we know, a group of these applications uses mean-field theory to solve the traffic problem, but they do not deal with the issue precisely. Assume that some travel requests are sent to the application, simultaneously. In order to propose the best route for all the requests, they use only the traffic information by having this big data and engineering it with machine learning method, they are able to optimize the travel time. However, they do not take into account the probable traffic that each car could create for each other after navigating; due to similar routes which the algorithm may provide for some requests.

The D-Wave Systems use Quantum annealing technologies like the quantum processing units (QPUs) to solve complex combinatorial optimization problems. We can demonstrate that the continuous redistribution of position data for cars in crowded road graphs as an example of the time-critical optimization tasks are suitable candidates for quantum computing.

It is necessary to mention that this project was done by Volkswagen and D-Wave Systems¹ and we only did a simple primary model of that. The source code of this project will be sent as a separate python file.

References

1. Neukart, F.; Compostella, G.; Seidel, C.; Von Dollen, D.; Yarkoni, S.; Parney, B., Traffic flow optimization using a quantum annealer. *Frontiers in ICT* **2017**, *4*, 29.