Assignment 4 Stoch Opt

Helene Randi Behrens

December 2019

1 Choices in implementation

- In my interpretation of the problem, the demand and congesture is modelled by the travel times for each segment.
- I have assumed that the travel times for each direction of the same road segment are i.i.d, as this intuitively gives a more natural model of traffic in a city.
- My computer was very slow in running the program for the 100x100 city problem, so I have run the simulation for a city with 20x20 crossings, with the exit at (0,4). You can test the program for 100X100 by changing the "dim" argument from 20 to 100.
- I have written the code in Python, so I have used 0-indexing.
- I have used a normal distribution to simulate variation in the driving times. This was not explicitly given, but I though it would give a reasonable model of the traffic image.

2 Modelling of problem and Full information

The "makeAdjacency" functions returns a model of the problem. The function returns an adjacency matrix, with elements [x,y] the average time it takes to drive from x to y. These are uniformly distributed between the values $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The coordinates for each crossing (X,Y) are given the index in the adjacency matrix by the transformation

$$(X, Y) \to X \cdot \text{dim} + Y,$$

where dim is the dimension of the road network. This index is then transformed back to the coordinates by the formula

$$(X, Y) \leftarrow (\text{index}//\text{dim}, \text{index}\%\text{dim}).$$

I have used the Dijkstra algorithm [https://en.wikipedia.org/wiki/Dijkstra % 27salgorithm] for solving shortest-path problems to find the optimal route from all crossings to the exit crossing.

The findNeighbours functions is a helping function that returns the possible next crossings from a given crossing.

3 Q-learning

In the full-information case, we found the optimal route by optimizing over the average driving times for each road segment. We will now look at a more realistic learning process, where this is not known. We get information about the road segments when a car drives there. In this simulation, these driving times are sampled from a normal distribution with variance one and mean the average driving time for that segment (the same values that are used in the full-information case). This sampling is done by the "realize" function.

Q-learning was then used to learn the optimal route to the target. We used the update:

$$Q_{n+1}(x_n, a_n) = \gamma[r(x_n, a_n) + \alpha \max_a Q_n(x_{n+1}, a)] + (1 - \gamma)Q_n(x_n, a_n).$$

I used the values $\alpha = 0.95$ and $\gamma = 0.05$. The reward for taking a direction a_n in a crossing x_n is modelled as minus the travel time plus 200 if the next crossing is the target. From each crossing, the next step is chosen to be the one with the highest Q-value.

3.1 ϵ-greedy

In this method, a random direction is chosen instead of the direction with the highest Q-value with probability ϵ, at each step. I used $\epsilon = 0.05$ and initial Q-values 0. In figure 1 we see how the relative difference between the travel times of the routes found by the ϵ-greedy algorithm and the optimal route found previously decreases with the learning process.

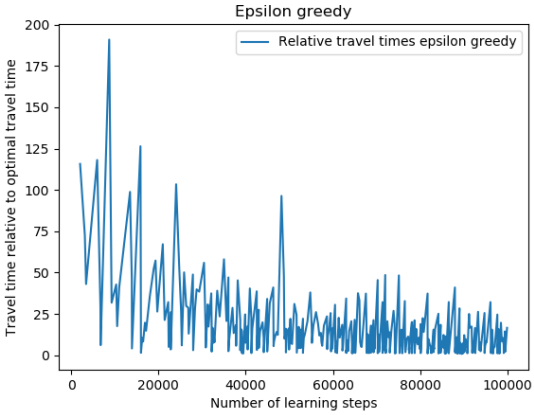

Figure 1: The travel times found by the ϵ-greedy algorithm relative to the optimal travel times, as a function of learning steps.

3.2 Optimistic Q-values

A way to improve the Q-learning process is to choose smart initial values. If these are chosen to increase exploration, i.e. somewhat higher than the expected Q-values, we can get a good relation between exploration and exploitation without the ϵ-approach. We can also use the knowledge that we want to move towards the target when choosing the initial values, by giving routes that lead towards the target a higher initial value than the routes leading away from the target. This is done in the function "qCloseToTarget", where initial values for road segments leading towards the target is set to 1 while the initial values for road segments leading away from the target is set to -5. The resulting learning process is seen in figure 2. Note that the routes found by following the initial Q-values actually seem to be better than the routes found after some time in the learning process. We then see an increase in the relative driving times, before the relative driving times again decrease at the end of the learning process. Still, the relative driving times using this method is overall quite a lot lower than the ϵ-greedy methods, as seen in figure 4.

3.3 A Smarter Heuristic; improving epsilon-greedy

As the ϵ-greedy algorithm performed the worst, we will try to find a heuristic to improve this. An intuitive solution to this, is to implement functionality such that the algorithm chooses the direction that leads us closest to the target if the Q-value is the same for two directions. This should improve the performance at least in the beginning, when most of the q-values are initialized to the same value. This is implemented in the functions "getNextPosSH" and "driveTowardsTarget". A comparison of the results are seen in figure

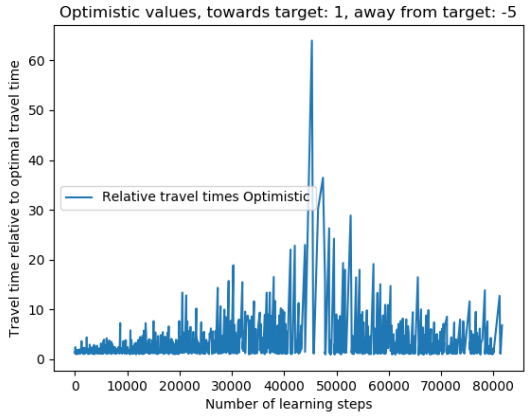

Figure 2: The travel times found by the algorithm with optimistic initial values relative to the optimal travel times, as a function of learning steps.

3. We can see that the difference is not big, but the improvement does seem to decrease the number of extreme observations.

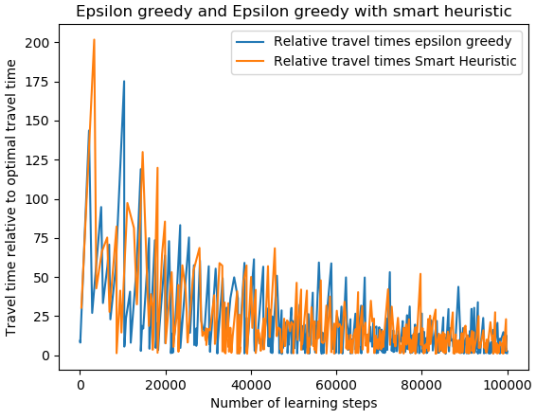

Figure 3: The travel times found by the original and improved ϵ-greedy methods relative to the optimal travel times, as a function of learning steps.

4 Right turns

To solve the problem when cars are only allowed to make right turns, we should implement the possible neighbours for each crossing differently (i.e. making the graph differently). The next possible steps from a given crossing is then either one step forward and one step right or two steps forward in each direction. The average travel times for these "new" road segments could then be found by adding the driving times for each road segments (as found in the old adjacency matrix.) Using this new adjacency matrix, the problem could be solved similarly.

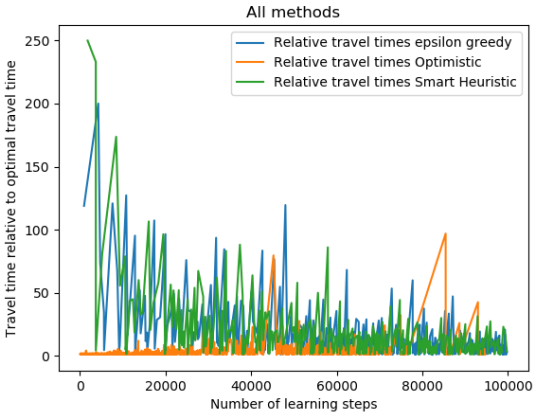

Figure 4: The travel times found by all the methods relative to the optimal travel times, as a function of learning steps.