In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?

Yes it does, but it takes a long time as at any moment the likelihood of the agent selecting the string of correct actions to reach the destination is n^4, where n is the number of actions needed to reach the destination (this is more than distance as it may not be safe to go at certain time steps). There are 4 actions, none, right, left and forward.

It's movement is of course random at this stage, and doesn't take into account rewards, right of way or the planner.

Justify why you picked these set of states, and how they model the agent and its environment.

The state should represent all information needed to choose the next move, modeling all the factors involved, so in this scenario I have decided to include the next waypoint, as well as the inputs apart from "right". This means that the state-action pair should give a deterministic reward in this model so therefore will be easier for the agent to learn.

The next waypoint gives the smart cab the intended direction, right, left or forward, while the inputs define if it is safe for the smart cab to take this direction, or if it should wait (do none). I believe this covers the factors my agent needs to be aware of to make it's decision, as it does not matter what the previous steps are and no global information is available (such as further steps beyond this one or where other agents are other than immediately next to my agent). Deadline will detailed later.

Specifically light instructs if the car can drive initially, green is go, red is stop. Additionally oncoming and left define additional right of way rules on top of this. For example if next_waypoint is "left" and there is another agent oncoming going forward or right then despite a green light our agent cannot move. Or if our next_waypoint is "right" and there isn't another agent to the left going forward then our agent can in fact move.

"Right" is omitted from defining the state as due to US right-of-way rules there is never a time when a car to the right would affect ability to go if light is already known.

There is a possibility that the inputs could help define a direction, for example if the agent works out that it can go right then up instead of up then right, if up is currently unsafe, to reach the same destination, however I believe in this scenario there isn't the right information for the agent to figure this out.

I have decided against including the deadline, as this would give too many states for the agent to learn from, the equivalent of too many features in supervised learning and so would require far more data / iterations. In addition the deadline would only provide impetus for the agent to speed up and I feel this is already rolled into the reward of reaching the next destination / waypoint.

Perhaps there would be a case if the next waypoint was blocked, and the deadline tight enough to need an alternative route to be taken as outlined above, that including deadline might help. However I believe this benefit to be too minimal and to intricate to be worth the increase in iterations needed and to be of enough benefit.

What changes do you notice in the agent's behavior?

After running a number of iterations it starts to develop a noticeable preference for heading towards the destination, and starts succeeding increasingly. In addition the penalties it receives reduces and the overall reward starts becoming positive. It would often succeed around 60/100 trials.

This change in behaviour comes as the algorithm updates the Q-Values it begins to prefer the actions that have previously given positive reward either immediately or after later actions (positive utility).

Occasionally though the algorithm would get stuck at a local optima, that was far from the global optima. This led to the smart cab succeeding much less, around 0-5 times out of 100 trials.

This probably comes as it makes sub-optimal choices but the update remains positive meaning the Q-Value increases. This means the algorithm is going to keep choosing this option without exploring if other actions would yield a better utility and value to the agent overall.

Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

I implemented a "optimism in the face of uncertainty" by initiating Q-values at values increasing from 0 to try ensure that new state-actions were explored an optimal amount, and more than under my previous algorithm. The results are outlined below in terms of a penalty score received in the last ten iterations (of 100 total). This was computed as for all rewards under 1 (suboptimal steps), sum (1 - reward value) for all rewards received. The results are totals over 20 runs.

Initial Q Value	0(20)	1(20)	3(20)	5(20)	7(20)	9 (20)	11(20)	13(20)	15(20)
Score	-39.5	-21.5	-39.5	-69.5	-51.5	-26.5	-55.5	-70.5	-47.0

This showed 1 to be optimal, and I feel this is due to the fact any reward of 1 or more is optimal so this eliminates the local optima described above. It is also perhaps the value closest to the "true" utilities, meaning less training time is needed to converge.

After a few iterations it starts to learn the way to travel very quickly, and after a few trials starts heading towards the destination almost straight away. In addition the penalty picked up for wrong actions at intersections after these first few trials is minimal.

Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties?

An optimal policy would be one that behaves as is described in the question, and so for any given state the Q-Value that would do this would be the highest value. This would be the Q-Value that is to move in the direction of the next waypoint when it is safe, and to stay still, or take action “none”, when it is not.

The smart cab appears to reach close to the optimal policy, minimising penalties and almost always reaching the destination (80%+ out 100 iterations).