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

The car wanders around aimlessly. It does occasionally make it to the end, but usually does not.

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

I am storing the light state, oncoming traffic, traffic to the left, and the direction of the next waypoint direction. The first three determine what are legal moves to make. The fourth is a hint as to where the car should go next. Traffic to the right doesn't matter as long as everyone is obeying the traffic laws. The deadline was also not stored, as it would explode the number of legal states and make it impossible for the agent to learn within a single run (as each state in that run would necessarily be a different state. It may be possible to bin the deadlines to allow it to regard the deadline as two states (plenty of time, not much time) and thus allow it to be more aggressive when it is running out of time. However that would still double the number of states it would have to learn, and the ones it would have the most trouble learning (due to a lack of examples) would probably be the running out of time states, which are the ones we are probably most concerned with. If we do want it to change behavior when it gets low on time, probably a better approach would be to adjust the explore probability based on the proximity to the deadline

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

It is now tending to go towards the target more often than not, and seems to be obeying traffic rules (at least as often as drivers around here do…)

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 started with alpha, gamma, and explore probability all set to 0.5. With those values, it was able to successfully navigate to the end 50% of the time in the final 10% of the runs. I first decreased the explore probability to make it more likely to use the learned route (there are a small number of states and plenty of time for it to learn something, so exploration isn't as important). I increased alpha as the game is pretty deterministic with regard to the transitions, so it should be able to trust that it is learning the right thing. Gamma was reduced based on the observation that the next state was usually independent on the action the agent takes at the current light. With alpha, gamma, and explore set to 0.75, 0.25, and 0.25 respectively and running 10 trials, it was able to get to the end 85% of the time in the final 10% of the runs. Reducing them further to .85, .15, and .15 resulted in 94% completion of the final 10% (using 10 trials). Reducing them to .95, .05, and .05 had it shrink to a 99% completion rate. Further optimization at this point seems impractical using this method as we don't have the precision to reliably see an improvement. Ideally we could run a sort of gradient descent to find the optimal parameters, though that would require additional modification of the base code to allow it to be called more programmatically.

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?

The optimal policy should be to follow all waypoints and obey all traffic laws. The car does seem to be usually finding the correct policy, but seems to have issues at the edge of the world. This is likely because the world seems to be doughnut shaped so if it goes off the edge it ends up on the other side. Yet the waypoints seem to be calculated as if the world has fixed edges, so if it takes an experimental turn off the edge, even though it is only one square further away it thinks it has to go all across the world. And even at the end of the simulation, the car does occasionally make exploration decisions, so there is always some chance it will make a "mistake" and try something it shouldn't.