Report ML Nanodegree P4: Train a Smartcab to drive

Author: Christian Graber

Implement a basic driving agent

Implement the basic driving agent, which processes the following inputs at each time step:

- Next waypoint location, relative to its current location and heading,
- Intersection state (traffic light and presence of cars), and,
- Current deadline value (time steps remaining),

And produces some random move/action `(None, 'forward', 'left', 'right')`. Don't try to implement the correct strategy! That's exactly what your agent is supposed to learn.

Run this agent within the simulation environment with `enforce_deadline` set to `False` (see `run` function in `agent.py`), and observe how it performs. In this mode, the agent is given unlimited time to reach the destination. The current state, action taken by your agent and reward/penalty earned are shown in the simulator.

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

The agents actions are random. There is no correlation between the inputs and the action taken. Success within the extended deadline is left to chance.

With enforce_deadline set to False, the agent has an additional 100 steps on top of the regular deadline. The additional 100 steps are a hard time limit. The agent sometimes makes it to the target and sometimes it hits the hard time limit before reaching the target. When the hard time limit is hit a new trial is started. By default the simulation has 100 trials.

Example of success, the agent hits the destination by chance:

```
Environment.act(): Primary agent has reached destination!
```

Example of failure, the agent does not hit the destination:

```
Environment.step(): Primary agent hit hard time limit (-100)! Trial aborted.
```

Identify and update state

Identify a set of states that you think are appropriate for modeling the driving agent. The main source of state variables are current inputs, but not all of them may be worth representing. Also, you can choose to explicitly define states, or use some combination (vector) of inputs as an implicit state.

At each time step, process the inputs and update the current state. Run it again (and as often as you need) to observe how the reported state changes through the run.

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

The state vector should definitely include the next waypoint (abbreviated 'next' from now on). This is pointing the smartcab in the direction of the shortest path to the target. The traffic light should also be included. The smartcab will often encounter a red traffic light and needs to learn to wait (or make a legal right turn). Finally the state of other traffic at the intersection can be included. This is also a project requirement. However, in this particular setup one could argue that the traffic could be ignored in first approximation. That is because traffic hardly ever interferes with our smartcab. If there was more traffic these rules would become more important.

Another input to address is the deadline. The first question to ask is whether we want to take the deadline into account at all. Do we want the smartcab to make different decisions based on deadline, e.g. would it be OK for the smartcab to go 'rogue' and run redlights and risk accidents when the deadline is close? In real life, the answer to this is no. We still want our smartcab to drive by the rules. The next question to ask is if this could be done at all. The immediate rewards don't take deadline into account. That means the smartcab cannot learn anything about deadline from immediate rewards. That means only future rewards are relevant. The reward for reaching the destination is actually quite large, yielding 10 additional points. However, as we will see in the following discussion, future rewards can only be approximated by a constant. We don't have the global GPS that would tell us how far from the destination we are. As stated in project description: (sorry, no accurate GPS, no global location). So we can't take future rewards into account. So we won't use deadline as input.

If all state is taken into account as described above we have a state space size of:

*Full state space: light+oncoming+left+right+next = 2*4*4*4*3 = 384*

(Aside: Assuming it was possible to use future rewards, and we were to take deadline into account, this statespace would explode. The deadline is computed in steps of 5 based on distance between starting point and destination. On the given 8x6 grid we have a max distance of (7 + 5) * 5 = 60. The deadline does decrease by one for each move, giving us 60 additional states max. That means worst case we'd have a multiplier of 60 in the statespace! Altogether 384 * 60 = 23040 states max, which is impossible to learn adequately in 100 trials.)

This is a rather large statespace to learn in 100 trials, as per project requirement. We actually have some knowledge about traffic rules that allows us to reduce this space. Some states can be ignored. Specifically, oncoming car right turn and left car right turn. They will never interfere with the smartcab. What we can also take advantage of is the fact that the penalty for causing an accident and the penalty for running a red light are the same in this environment. That means this environment does make a

difference if you run a red light without causing an accident or if your run the red light and cause an accident. That means we can make the smartcab learn the simpler rule of never running a red light while ignoring any other cars that might be around.

The reduced state space can be calculated as follows:

Green light: oncoming+left+right+next = 3 * 3 * 4 * 3 = 108

(Notice that oncoming and left have option right removed because it is being ignored)

Red light: oncoming+left+right+next = 2 * 2 * 1 * 3 = 12

(Notice that for oncoming and left only options None and left matter. For right no option matters because our smartcab can only interfere with traffic from right if it runs the right light)

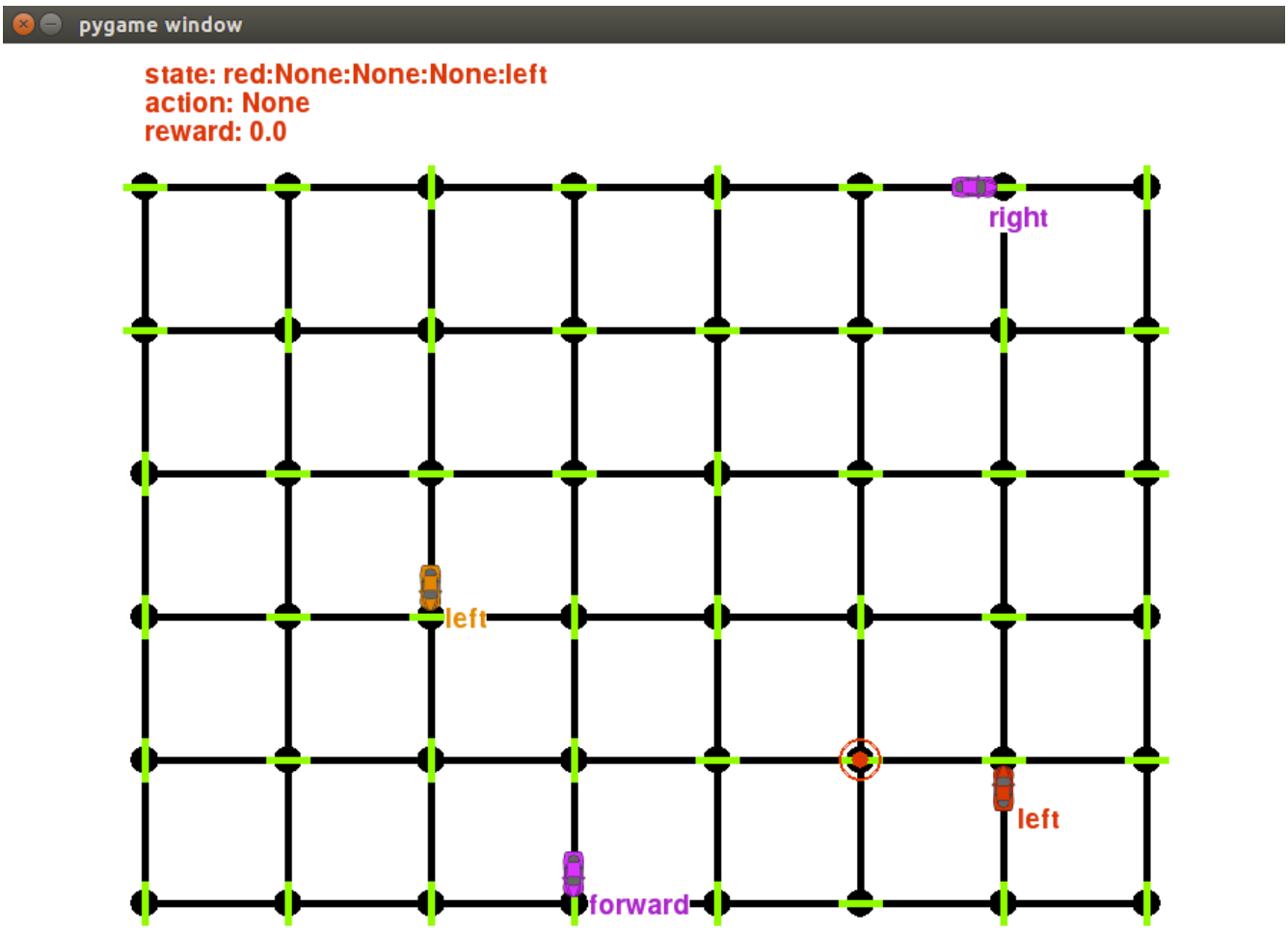
Total reduced state space: green + red = 108 + 12 = 120

This is still a very large state space. As suggested above the traffic could be ignored in first approximation. The traffic could be seen as a statistical disturbance, or noise. Q learning can handle some noise and still converge. With traffic ignored, the state space would simply be:

*Minimal state space: light+next = 2*3 = 6*

This size of space can probably be learned in full 100 trials and return a viable policy. However, requirements also have us include traffic. For comparison purposes we might want to run this space, but for the remainder of this exercise the reduced state space is used.

Snapshot of display showing state from reduced state space:


pygame window
state: red:None:None:None:left
action: None
reward: 0.0
right
left
left
forward

Implement Q-Learning

Implement the Q-Learning algorithm by initializing and updating a table/mapping of Q-values at each time step. Now, instead of randomly selecting an action, pick the best action available from the current state based on Q-values, and return that.

Each action generates a corresponding numeric reward or penalty (which may be zero). Your agent should take this into account when updating Q-values. Run it again, and observe the behavior.

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

Using the Q learning update rule it seems that future rewards are non-deterministic. Instead approximate future rewards with a constant. The constant chosen is the max possible reward in the next state.

Here is the implemented update rule:

```
nextstate = nowstate * ( 1 - alpha ) + alpha * ( reward + gamma * 2.0)
```

```
self.alpha = 0.5
self.gamma = 0.5
```

The Q matrix is initialized to all 0.0 at the beginning of each run.

In this first run the agent will always select a random action. What that means is that the agent is basically in exploration mode. The agent is in learning mode for all 100 trials.

Q matrix with random selection of action after 100 trials:

```
Number of states reached 31
                                    forward       left    right       wait
red:None:None:None:forward         0.000000  0.000000  0.50000  1.000000
green:None:None:None:left          0.500000  3.312524  0.50000  1.000000
green:None:None:None:right         0.500000  0.812653  3.00000  1.000000
red:None:None:None:left            0.000000  0.000000  0.50000  1.000000
green:None:None:None:forward       3.000001  0.500000  0.50000  1.000000
red:None:None:None:right           0.000000  0.000000  3.00000  1.000000
green:None:None:right:right        0.250000  0.000000  1.50000  0.000000
green:None:None:left:left          0.250000  1.500000  0.25000  0.750000
green:left:None:None:right         0.484375  0.375000  3.53125  0.750000
green:None:None:forward:forward    0.000000  0.250000  0.25000  0.500000
red:left:None:None:right           0.000000  0.000000  2.25000  0.984375
green:None:None:forward:right      0.250000  0.250000  0.00000  0.000000
green:None:None:left:forward       1.500000  0.250000  0.37500  0.000000
red:left:None:None:forward         0.000000  0.000000  0.25000  0.000000
green:None:None:right:left         0.250000  0.000000  0.00000  0.000000
green:None:None:forward:left       0.000000  2.625000  0.00000  0.000000
green:None:forward:None:right      0.250000  5.437500  1.50000  0.000000
red:None:forward:None:left         0.000000  0.000000  0.00000  0.000000
```

red:None:forward:None:right	0.000000	0.000000	1.50000	0.500000
green:left:None:None:forward	0.000000	0.250000	0.00000	0.000000
green:None:left:None:right	0.250000	0.250000	1.50000	0.000000
red:left:None:None:left	0.000000	0.000000	0.00000	0.750000
green:left:None:None:left	0.000000	0.000000	0.25000	0.000000
green:None:forward:None:forward	1.500000	0.000000	0.00000	0.000000
green:forward:None:None:forward	0.000000	0.000000	0.00000	0.000000
green:None:None:right:forward	1.500000	0.250000	0.00000	0.500000
red:None:forward:None:forward	0.000000	0.000000	0.00000	0.000000
green:None:forward:None:left	0.000000	1.500000	0.00000	0.000000
green:None:left:None:forward	6.500000	0.000000	0.25000	0.000000
green:None:left:None:left	0.250000	1.500000	0.00000	0.500000
green:None:None:left:right	0.375000	0.000000	0.00000	0.000000

For the next trial run the action is picked from the Q matrix while it is being trained. This is called exploitation. This turns out as as a non-viable strategy. The Q matrix does not converge to Q*, but is stuck early on. So exploiting a basically untrained Q matrix does not help.

What we need is a policy that first learns and then uses the best learned action. That means we need to strike a balance between exploration and exploitation. We have to start with exploration and then switch to exploitation, gradually or abrupt. See discussion in next section.

Enhance the driving agent

Apply the reinforcement learning techniques you have learnt, and tweak the parameters (e.g. learning rate, discount factor, action selection method, etc.), to improve the performance of your agent. Your goal is to get it to a point so that within 100 trials, the agent is able to learn a feasible policy - i.e. reach the destination within the allotted time, with net reward remaining positive.

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?

As discussed, the agent has to find a balance between exploration and exploitation. Exploration means picking an action at random. Exploitation means picking the best action from the Q matrix.

The goal is to gradually switch from exploration to exploitation within our 100 trials. To achieve this, the policy implemented here will randomly select exploration or exploitation based on parameter epsilon with range [0,1]. If a random sample from range [0,1] is smaller or equal to epsilon a random action is picked. Otherwise the best action is picked. Epsilon is initialized to 1.0 and will decrease linearly until a floor of 0.05 is reached. That means the agent will explore at 5% of all moves perpetually.

Implemented action policy:

```
# Action Policy
    sample = random.random()
    if sample <= epsilon:
        action = randaction # random action
    else:
        action = qaction # best action from Q matrix

    # Slowly decrease epsilon, leave at 5% floor
    if self.epsilon > .05:
        self.epsilon -= .00075
```

The agent performs well with this policy. After about 2/3 of all trials the policy will have reached the 5% floor. At this point the agent is trained properly with the large majority of actions now being exploitative.

Adjusting Gamma:

Gamma usually represents the weight of future rewards. However, we approximated future rewards by a constant since they cannot be determined. It turns out that parameter gamma is not essential. It merely adjusts the size of the constant that was chosen to approximate future rewards. And a different size of this value will only adjust the size of values in the Q matrix, it scales all values up or down. Setting gamma to 0 will be the equivalent of leaving it out completely. However, this leads to negative values in the Q matrix. For that reason gamma is left at 0.5. I rather have positive only values in Q.

Adjusting Alpha:

Alpha is the learning rate. So far it was set to 0.5. No clear pattern emerges when trying different alpha rates. There is quite of bit of randomization in these runs. Consecutive runs with alpha 0.5 produced 20 and 25 aborted trials. Reason for the unclear pattern could be too short of a training period within 100 trials. Accordingly we will stick with alpha 0.5.

Alpha	Aborted trials out of 100
0.01	23
0.1	15
0.5	20 (25)
0.9	18
0.99	23

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?

Given the fact that traffic is very light not the entire state space is reached within 100 trials. It would be desirable to extend the training period and also increase the traffic to make sure all possible states are hit.

As can be seen in appendix A, progression of 100 trials, the agent reaches its destination with positive total reward reliably after about 50 trials.

Here is the Q matrix reached at the end of 100 trials with described policy and alpha 0.5:

```
Number of states reached 26
                                  forward       left      right       wait
red:None:None:None:right         0.000000   0.000000   3.313110   1.000000
green:None:None:None:right       0.500001   0.500005   3.000011   1.000000
green:None:None:None:forward     3.176411   0.500000   0.500000   1.000000
red:None:None:None:left          0.000000   0.000000   0.656249   1.000000
green:None:None:None:left        0.500000   9.250106   0.499969   0.999969
```

green:forward:None:None:right	0.000000	0.000000	0.000000	0.000000
red:None:None:None:forward	0.000000	0.000000	0.500000	1.000000
red:left:None:None:right	0.000000	0.000000	2.625000	0.500000
green:None:forward:None:left	0.000000	0.000000	0.375000	0.000000
green:None:forward:None:right	0.250000	0.250000	0.000000	0.000000
green:None:None:forward:right	0.000000	0.000000	2.250000	0.000000
green:None:None:left:forward	7.812500	0.000000	0.000000	0.000000
green:None:None:forward:left	0.000000	1.500000	0.250000	0.000000
red:left:None:None:left	0.000000	0.000000	0.375000	0.000000
green:None:left:None:left	0.000000	2.625000	0.000000	0.000000
red:None:forward:None:right	0.000000	0.000000	0.000000	0.000000
green:None:None:right:right	0.000000	0.000000	0.000000	0.750000
green:None:forward:None:forward	0.000000	0.468750	0.000000	0.000000
green:left:None:None:right	0.000000	0.375000	0.000000	0.968750
green:None:None:forward:forward	3.875000	0.000000	0.000000	0.000000
red:left:None:None:forward	0.000000	0.000000	0.000000	0.000000
green:left:None:None:forward	6.570312	0.000000	0.250000	0.000000
green:left:forward:None:left	0.000000	0.000000	0.250000	0.000000
green:None:None:right:forward	1.500000	0.000000	0.000000	0.000000
green:None:None:left:left	0.250000	0.000000	0.000000	0.000000
red:None:forward:None:forward	0.000000	0.000000	0.000000	0.000000

To get closer to the optimal policy we adjust the policy to explore only and then run 1000 trials. The result can be seen below.

Our 100 trial Q matrix resembles below matrix in all matching states for all practical purposes. That means it will execute the same actions.

Q matrix after 1000 trials:

Number of states reached 44

	forward	left	right	wait
green:None:None:None:right	0.500005	0.500000	3.000000	1.000000
green:None:None:None:left	0.500000	3.000000	0.500000	1.000000
red:None:None:None:forward	0.000000	0.000000	0.500000	1.000000
green:None:None:None:forward	3.000000	0.500000	0.500000	1.000000
red:None:None:None:right	0.000000	0.000000	3.000000	1.000000
red:None:None:None:left	0.000000	0.000000	0.500000	1.000000
red:left:None:None:left	0.000000	0.000000	0.500000	1.000000
green:None:left:None:forward	2.999999	0.500000	0.500000	0.999969
green:forward:None:None:left	0.484375	0.000000	0.437500	0.999512
green:None:None:left:left	0.500000	3.001221	0.499969	1.000000
red:left:None:None:right	0.000000	0.000000	3.000000	1.000000
green:left:None:None:right	0.500000	0.500000	3.000000	1.000000
green:left:None:None:left	0.499756	2.999908	0.499992	0.999969
green:None:None:left:right	0.500000	0.468750	3.000000	1.000000
red:None:forward:None:left	0.000000	0.000000	0.498047	0.999512
green:None:None:forward:left	0.499996	2.999997	0.656219	0.999999
green:left:None:None:forward	3.039064	0.500000	0.500000	1.000000
green:None:None:left:forward	2.999977	0.500000	0.500000	1.000000
green:forward:None:None:right	0.499985	0.000000	2.994141	0.998047

green:None:None:right:right	0.499023	0.499985	2.999817	0.999999
green:None:forward:None:left	0.499998	2.999977	0.504883	0.999999
green:None:None:forward:right	0.500000	0.499996	3.019531	1.000000
red:left:forward:None:right	0.000000	0.000000	0.000000	0.500000
green:None:left:None:right	0.500000	0.500000	3.000000	1.000000
green:None:forward:None:forward	3.000000	0.500000	0.499999	1.000000
green:None:None:right:forward	2.953125	0.499512	0.499512	1.000000
green:None:forward:None:right	0.500000	0.502441	3.000019	1.000000
green:None:left:None:left	0.500000	3.000000	0.499996	0.999996
red:None:forward:None:right	0.000000	0.000000	2.999634	0.999969
red:left:None:None:forward	0.000000	0.000000	0.500000	1.000000
green:forward:None:None:forward	2.997070	0.000000	0.496094	0.984375
green:None:None:forward:forward	2.999268	0.500000	0.499992	0.999998
red:None:forward:None:forward	0.000000	0.000000	0.468750	0.937500
green:None:None:right:left	0.499969	2.250000	0.496094	0.998047
green:left:None:forward:right	0.000000	0.250000	0.000000	0.000000
green:left:None:left:left	0.000000	0.000000	0.000000	0.500000
green:None:left:left:right	0.250000	0.000000	0.000000	0.750000
green:None:forward:left:right	0.250000	0.000000	0.000000	0.000000
green:left:None:forward:forward	0.000000	0.000000	0.250000	0.000000
green:None:left:forward:right	0.250000	0.000000	0.000000	0.750000
green:None:left:forward:forward	0.000000	0.375000	0.000000	0.500000
green:None:left:left:left	0.000000	0.000000	0.250000	0.000000
green:None:forward:left:left	0.250000	0.000000	0.000000	0.000000
green:None:left:left:forward	0.000000	0.000000	0.250000	0.000000

The optimal Q* matrix would have a complete state space of all 384 states. And the values would have converged to clean values of 3.0, 1.0, 0.5 etc.. That means the optimal Q* matrix would take the right action for every state.

Important to note is that our Q matrix after 100 trials will take the same actions as the optimal matrix for the states these matrices have in common. The first states our matrix learns are also the most relevant ones, meaning our smartcab achieves acceptable performance fast. Only some of the rarer states are not handled well by our agent vs. the optimal agent.

Appendix A: Progression of 100 trials

```
Simulator.run(): Trial 0
Environment.reset(): Trial set up with start = (3, 6), destination = (5, 3), deadline = 25
RoutePlanner.route_to(): destination = (5, 3)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 1
Environment.reset(): Trial set up with start = (6, 2), destination = (2, 5), deadline = 35
RoutePlanner.route_to(): destination = (2, 5)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 2
Environment.reset(): Trial set up with start = (7, 1), destination = (8, 6), deadline = 30
RoutePlanner.route_to(): destination = (8, 6)
Environment.act(): Primary agent has reached destination! Total reward: -3.0
Simulator.run(): Trial 3
Environment.reset(): Trial set up with start = (4, 5), destination = (1, 1), deadline = 35
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 3.5
Simulator.run(): Trial 4
Environment.reset(): Trial set up with start = (8, 5), destination = (1, 3), deadline = 45
RoutePlanner.route_to(): destination = (1, 3)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 5
Environment.reset(): Trial set up with start = (8, 5), destination = (3, 3), deadline = 35
RoutePlanner.route_to(): destination = (3, 3)
Environment.act(): Primary agent has reached destination! Total reward: 9.5
Simulator.run(): Trial 6
Environment.reset(): Trial set up with start = (3, 6), destination = (2, 1), deadline = 30
RoutePlanner.route_to(): destination = (2, 1)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 7
Environment.reset(): Trial set up with start = (4, 2), destination = (6, 4), deadline = 20
RoutePlanner.route_to(): destination = (6, 4)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 8
Environment.reset(): Trial set up with start = (4, 1), destination = (1, 6), deadline = 40
RoutePlanner.route_to(): destination = (1, 6)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 9
Environment.reset(): Trial set up with start = (7, 6), destination = (3, 6), deadline = 20
RoutePlanner.route_to(): destination = (3, 6)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 10
Environment.reset(): Trial set up with start = (3, 6), destination = (6, 3), deadline = 30
RoutePlanner.route_to(): destination = (6, 3)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 11
Environment.reset(): Trial set up with start = (8, 5), destination = (2, 5), deadline = 30
RoutePlanner.route_to(): destination = (2, 5)
Environment.act(): Primary agent has reached destination! Total reward: 3.5
Simulator.run(): Trial 12
Environment.reset(): Trial set up with start = (8, 1), destination = (1, 2), deadline = 40
RoutePlanner.route_to(): destination = (1, 2)
Environment.act(): Primary agent has reached destination! Total reward: 5.0
Simulator.run(): Trial 13
Environment.reset(): Trial set up with start = (5, 2), destination = (2, 5), deadline = 30
RoutePlanner.route_to(): destination = (2, 5)
Environment.act(): Primary agent has reached destination! Total reward: -3.0
Simulator.run(): Trial 14
Environment.reset(): Trial set up with start = (3, 4), destination = (6, 6), deadline = 25
RoutePlanner.route_to(): destination = (6, 6)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 15
Environment.reset(): Trial set up with start = (8, 3), destination = (2, 4), deadline = 35
RoutePlanner.route_to(): destination = (2, 4)
Environment.act(): Primary agent has reached destination! Total reward: -1.0
Simulator.run(): Trial 16
Environment.reset(): Trial set up with start = (4, 4), destination = (6, 2), deadline = 20
RoutePlanner.route_to(): destination = (6, 2)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 17
Environment.reset(): Trial set up with start = (5, 1), destination = (2, 5), deadline = 35
RoutePlanner.route_to(): destination = (2, 5)
Environment.act(): Primary agent has reached destination! Total reward: 9.0
Simulator.run(): Trial 18
Environment.reset(): Trial set up with start = (4, 6), destination = (4, 2), deadline = 20
RoutePlanner.route_to(): destination = (4, 2)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 19
Environment.reset(): Trial set up with start = (3, 3), destination = (1, 1), deadline = 20
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 2.5
Simulator.run(): Trial 20
Environment.reset(): Trial set up with start = (4, 5), destination = (8, 3), deadline = 30
RoutePlanner.route_to(): destination = (8, 3)
Environment.act(): Primary agent has reached destination! Total reward: 6.5
Simulator.run(): Trial 21
Environment.reset(): Trial set up with start = (1, 5), destination = (7, 2), deadline = 45
RoutePlanner.route_to(): destination = (7, 2)
Environment.act(): Primary agent has reached destination! Total reward: 1.5
Simulator.run(): Trial 22
Environment.reset(): Trial set up with start = (6, 3), destination = (8, 1), deadline = 20
```

```
RoutePlanner.route_to(): destination = (8, 1)
Environment.act(): Primary agent has reached destination! Total reward: 5.5
Simulator.run(): Trial 23
Environment.reset(): Trial set up with start = (3, 1), destination = (5, 3), deadline = 20
RoutePlanner.route_to(): destination = (5, 3)
Environment.act(): Primary agent has reached destination! Total reward: 5.0
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 24
Environment.reset(): Trial set up with start = (3, 3), destination = (6, 4), deadline = 20
RoutePlanner.route_to(): destination = (6, 4)
Environment.act(): Primary agent has reached destination! Total reward: 7.5
Simulator.run(): Trial 25
Environment.reset(): Trial set up with start = (1, 5), destination = (7, 2), deadline = 45
RoutePlanner.route_to(): destination = (7, 2)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 26
Environment.reset(): Trial set up with start = (8, 6), destination = (6, 1), deadline = 35
RoutePlanner.route_to(): destination = (6, 1)
Environment.act(): Primary agent has reached destination! Total reward: 13.5
Simulator.run(): Trial 27
Environment.reset(): Trial set up with start = (5, 3), destination = (3, 6), deadline = 25
RoutePlanner.route_to(): destination = (3, 6)
Environment.act(): Primary agent has reached destination! Total reward: 8.5
Simulator.run(): Trial 28
Environment.reset(): Trial set up with start = (1, 6), destination = (1, 1), deadline = 25
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 1.5
Simulator.run(): Trial 29
Environment.reset(): Trial set up with start = (5, 5), destination = (8, 6), deadline = 20
RoutePlanner.route_to(): destination = (8, 6)
Environment.act(): Primary agent has reached destination! Total reward: 5.5
Simulator.run(): Trial 30
Environment.reset(): Trial set up with start = (6, 5), destination = (2, 3), deadline = 30
RoutePlanner.route_to(): destination = (2, 3)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 31
Environment.reset(): Trial set up with start = (1, 5), destination = (6, 5), deadline = 25
RoutePlanner.route_to(): destination = (6, 5)
Environment.act(): Primary agent has reached destination! Total reward: 6.5
Simulator.run(): Trial 32
Environment.reset(): Trial set up with start = (7, 6), destination = (5, 1), deadline = 35
RoutePlanner.route_to(): destination = (5, 1)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 33
Environment.reset(): Trial set up with start = (5, 1), destination = (1, 4), deadline = 35
RoutePlanner.route_to(): destination = (1, 4)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 34
Environment.reset(): Trial set up with start = (2, 5), destination = (7, 5), deadline = 25
RoutePlanner.route_to(): destination = (7, 5)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 35
Environment.reset(): Trial set up with start = (5, 5), destination = (1, 1), deadline = 40
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 16.5
Simulator.run(): Trial 36
Environment.reset(): Trial set up with start = (4, 2), destination = (8, 6), deadline = 40
RoutePlanner.route_to(): destination = (8, 6)
Environment.act(): Primary agent has reached destination! Total reward: 23.5
Simulator.run(): Trial 37
Environment.reset(): Trial set up with start = (3, 5), destination = (7, 6), deadline = 25
RoutePlanner.route_to(): destination = (7, 6)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 38
Environment.reset(): Trial set up with start = (4, 5), destination = (3, 2), deadline = 20
RoutePlanner.route_to(): destination = (3, 2)
Environment.act(): Primary agent has reached destination! Total reward: 11.0
Simulator.run(): Trial 39
Environment.reset(): Trial set up with start = (3, 2), destination = (6, 1), deadline = 20
RoutePlanner.route_to(): destination = (6, 1)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 40
Environment.reset(): Trial set up with start = (4, 6), destination = (6, 1), deadline = 35
RoutePlanner.route_to(): destination = (6, 1)
Environment.act(): Primary agent has reached destination! Total reward: 4.0
Simulator.run(): Trial 41
Environment.reset(): Trial set up with start = (6, 6), destination = (4, 4), deadline = 20
RoutePlanner.route_to(): destination = (4, 4)
Environment.act(): Primary agent has reached destination! Total reward: 3.5
Simulator.run(): Trial 42
Environment.reset(): Trial set up with start = (1, 3), destination = (3, 1), deadline = 20
RoutePlanner.route_to(): destination = (3, 1)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 43
Environment.reset(): Trial set up with start = (7, 4), destination = (1, 3), deadline = 35
RoutePlanner.route_to(): destination = (1, 3)
Environment.act(): Primary agent has reached destination! Total reward: 13.0
Simulator.run(): Trial 44
Environment.reset(): Trial set up with start = (8, 5), destination = (2, 2), deadline = 45
RoutePlanner.route_to(): destination = (2, 2)
Environment.act(): Primary agent has reached destination! Total reward: 11.0
Simulator.run(): Trial 45
Environment.reset(): Trial set up with start = (5, 6), destination = (1, 3), deadline = 35
RoutePlanner.route_to(): destination = (1, 3)
```

```
Environment.act(): Primary agent has reached destination! Total reward: 9.0
Simulator.run(): Trial 46
Environment.reset(): Trial set up with start = (7, 2), destination = (3, 2), deadline = 20
RoutePlanner.route_to(): destination = (3, 2)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 47
Environment.reset(): Trial set up with start = (5, 1), destination = (1, 6), deadline = 45
RoutePlanner.route_to(): destination = (1, 6)
Environment.act(): Primary agent has reached destination! Total reward: 3.5
Simulator.run(): Trial 48
Environment.reset(): Trial set up with start = (7, 2), destination = (1, 3), deadline = 35
RoutePlanner.route_to(): destination = (1, 3)
Environment.act(): Primary agent has reached destination! Total reward: 13.5
Simulator.run(): Trial 49
Environment.reset(): Trial set up with start = (7, 1), destination = (2, 1), deadline = 25
RoutePlanner.route_to(): destination = (2, 1)
Environment.act(): Primary agent has reached destination! Total reward: 8.0
Simulator.run(): Trial 50
Environment.reset(): Trial set up with start = (3, 3), destination = (8, 2), deadline = 30
RoutePlanner.route_to(): destination = (8, 2)
Environment.act(): Primary agent has reached destination! Total reward: 9.0
Simulator.run(): Trial 51
Environment.reset(): Trial set up with start = (7, 6), destination = (4, 1), deadline = 40
RoutePlanner.route_to(): destination = (4, 1)
Environment.act(): Primary agent has reached destination! Total reward: 8.5
Simulator.run(): Trial 52
Environment.reset(): Trial set up with start = (4, 2), destination = (5, 6), deadline = 25
RoutePlanner.route_to(): destination = (5, 6)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 53
Environment.reset(): Trial set up with start = (6, 6), destination = (4, 3), deadline = 25
RoutePlanner.route_to(): destination = (4, 3)
Environment.act(): Primary agent has reached destination! Total reward: 7.0
Simulator.run(): Trial 54
Environment.reset(): Trial set up with start = (1, 5), destination = (7, 6), deadline = 35
RoutePlanner.route_to(): destination = (7, 6)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 55
Environment.reset(): Trial set up with start = (1, 1), destination = (5, 5), deadline = 40
RoutePlanner.route_to(): destination = (5, 5)
Environment.act(): Primary agent has reached destination! Total reward: 4.0
Simulator.run(): Trial 56
Environment.reset(): Trial set up with start = (1, 6), destination = (5, 4), deadline = 30
RoutePlanner.route_to(): destination = (5, 4)
Environment.act(): Primary agent has reached destination! Total reward: 8.0
Simulator.run(): Trial 57
Environment.reset(): Trial set up with start = (2, 6), destination = (1, 2), deadline = 25
RoutePlanner.route_to(): destination = (1, 2)
Environment.act(): Primary agent has reached destination! Total reward: 4.0
Simulator.run(): Trial 58
Environment.reset(): Trial set up with start = (4, 1), destination = (2, 3), deadline = 20
RoutePlanner.route_to(): destination = (2, 3)
Environment.act(): Primary agent has reached destination! Total reward: 7.5
Simulator.run(): Trial 59
Environment.reset(): Trial set up with start = (6, 1), destination = (7, 4), deadline = 20
RoutePlanner.route_to(): destination = (7, 4)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 60
Environment.reset(): Trial set up with start = (1, 6), destination = (4, 5), deadline = 20
RoutePlanner.route_to(): destination = (4, 5)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 61
Environment.reset(): Trial set up with start = (5, 2), destination = (2, 4), deadline = 25
RoutePlanner.route_to(): destination = (2, 4)
Environment.act(): Primary agent has reached destination! Total reward: 8.0
Simulator.run(): Trial 62
Environment.reset(): Trial set up with start = (3, 3), destination = (2, 6), deadline = 20
RoutePlanner.route_to(): destination = (2, 6)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 63
Environment.reset(): Trial set up with start = (8, 5), destination = (5, 2), deadline = 30
RoutePlanner.route_to(): destination = (5, 2)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 64
Environment.reset(): Trial set up with start = (7, 6), destination = (4, 5), deadline = 20
RoutePlanner.route_to(): destination = (4, 5)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 65
Environment.reset(): Trial set up with start = (5, 3), destination = (1, 5), deadline = 30
RoutePlanner.route_to(): destination = (1, 5)
Environment.act(): Primary agent has reached destination! Total reward: 10.0
Simulator.run(): Trial 66
Environment.reset(): Trial set up with start = (3, 3), destination = (5, 1), deadline = 20
RoutePlanner.route_to(): destination = (5, 1)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 67
Environment.reset(): Trial set up with start = (6, 2), destination = (4, 4), deadline = 20
RoutePlanner.route_to(): destination = (4, 4)
Environment.act(): Primary agent has reached destination! Total reward: 3.5
Simulator.run(): Trial 68
Environment.reset(): Trial set up with start = (5, 5), destination = (1, 5), deadline = 20
RoutePlanner.route_to(): destination = (1, 5)
Environment.act(): Primary agent has reached destination! Total reward: 5.0
Simulator.run(): Trial 69
```

```
Environment.reset(): Trial set up with start = (8, 4), destination = (1, 1), deadline = 50
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 28.5
Simulator.run(): Trial 70
Environment.reset(): Trial set up with start = (2, 6), destination = (5, 2), deadline = 35
RoutePlanner.route_to(): destination = (5, 2)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 71
Environment.reset(): Trial set up with start = (2, 5), destination = (3, 2), deadline = 20
RoutePlanner.route_to(): destination = (3, 2)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 72
Environment.reset(): Trial set up with start = (4, 1), destination = (4, 6), deadline = 25
RoutePlanner.route_to(): destination = (4, 6)
Environment.act(): Primary agent has reached destination! Total reward: 7.0
Simulator.run(): Trial 73
Environment.reset(): Trial set up with start = (3, 2), destination = (7, 1), deadline = 25
RoutePlanner.route_to(): destination = (7, 1)
Environment.act(): Primary agent has reached destination! Total reward: 8.0
Simulator.run(): Trial 74
Environment.reset(): Trial set up with start = (8, 5), destination = (6, 1), deadline = 30
RoutePlanner.route_to(): destination = (6, 1)
Environment.act(): Primary agent has reached destination! Total reward: 13.0
Simulator.run(): Trial 75
Environment.reset(): Trial set up with start = (3, 1), destination = (7, 1), deadline = 20
RoutePlanner.route_to(): destination = (7, 1)
Environment.step(): Primary agent ran out of time! Trial aborted.
Simulator.run(): Trial 76
Environment.reset(): Trial set up with start = (5, 3), destination = (3, 1), deadline = 20
RoutePlanner.route_to(): destination = (3, 1)
Environment.act(): Primary agent has reached destination! Total reward: 2.5
Simulator.run(): Trial 77
Environment.reset(): Trial set up with start = (6, 6), destination = (4, 3), deadline = 25
RoutePlanner.route_to(): destination = (4, 3)
Environment.act(): Primary agent has reached destination! Total reward: 5.5
Simulator.run(): Trial 78
Environment.reset(): Trial set up with start = (8, 6), destination = (1, 2), deadline = 55
RoutePlanner.route_to(): destination = (1, 2)
Environment.act(): Primary agent has reached destination! Total reward: 13.5
Simulator.run(): Trial 79
Environment.reset(): Trial set up with start = (4, 6), destination = (2, 1), deadline = 35
RoutePlanner.route_to(): destination = (2, 1)
Environment.act(): Primary agent has reached destination! Total reward: 4.0
Simulator.run(): Trial 80
Environment.reset(): Trial set up with start = (5, 5), destination = (2, 6), deadline = 20
RoutePlanner.route_to(): destination = (2, 6)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 81
Environment.reset(): Trial set up with start = (1, 5), destination = (3, 1), deadline = 30
RoutePlanner.route_to(): destination = (3, 1)
Environment.act(): Primary agent has reached destination! Total reward: 10.0
Simulator.run(): Trial 82
Environment.reset(): Trial set up with start = (5, 1), destination = (1, 3), deadline = 30
RoutePlanner.route_to(): destination = (1, 3)
Environment.act(): Primary agent has reached destination! Total reward: 7.5
Simulator.run(): Trial 83
Environment.reset(): Trial set up with start = (6, 4), destination = (2, 1), deadline = 35
RoutePlanner.route_to(): destination = (2, 1)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 84
Environment.reset(): Trial set up with start = (3, 2), destination = (7, 4), deadline = 30
RoutePlanner.route_to(): destination = (7, 4)
Environment.act(): Primary agent has reached destination! Total reward: 10.0
Simulator.run(): Trial 85
Environment.reset(): Trial set up with start = (8, 2), destination = (3, 5), deadline = 40
RoutePlanner.route_to(): destination = (3, 5)
Environment.act(): Primary agent has reached destination! Total reward: 14.0
Simulator.run(): Trial 86
Environment.reset(): Trial set up with start = (7, 2), destination = (2, 4), deadline = 35
RoutePlanner.route_to(): destination = (2, 4)
Environment.act(): Primary agent has reached destination! Total reward: 13.5
Simulator.run(): Trial 87
Environment.reset(): Trial set up with start = (8, 5), destination = (3, 1), deadline = 45
RoutePlanner.route_to(): destination = (3, 1)
Environment.act(): Primary agent has reached destination! Total reward: 19.0
Simulator.run(): Trial 88
Environment.reset(): Trial set up with start = (1, 5), destination = (7, 2), deadline = 45
RoutePlanner.route_to(): destination = (7, 2)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 89
Environment.reset(): Trial set up with start = (1, 3), destination = (4, 5), deadline = 25
RoutePlanner.route_to(): destination = (4, 5)
Environment.act(): Primary agent has reached destination! Total reward: 12.0
Simulator.run(): Trial 90
Environment.reset(): Trial set up with start = (3, 4), destination = (6, 5), deadline = 20
RoutePlanner.route_to(): destination = (6, 5)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 91
Environment.reset(): Trial set up with start = (8, 3), destination = (3, 6), deadline = 40
RoutePlanner.route_to(): destination = (3, 6)
Environment.act(): Primary agent has reached destination! Total reward: 14.0
Simulator.run(): Trial 92
Environment.reset(): Trial set up with start = (4, 4), destination = (1, 6), deadline = 25
RoutePlanner.route_to(): destination = (1, 6)
```

```
Environment.act(): Primary agent has reached destination! Total reward: 8.0
Simulator.run(): Trial 93
Environment.reset(): Trial set up with start = (1, 6), destination = (8, 1), deadline = 60
RoutePlanner.route_to(): destination = (8, 1)
Environment.act(): Primary agent has reached destination! Total reward: 22.0
Simulator.run(): Trial 94
Environment.reset(): Trial set up with start = (7, 5), destination = (1, 1), deadline = 50
RoutePlanner.route_to(): destination = (1, 1)
Environment.act(): Primary agent has reached destination! Total reward: 18.0
Simulator.run(): Trial 95
Environment.reset(): Trial set up with start = (3, 5), destination = (6, 2), deadline = 30
RoutePlanner.route_to(): destination = (6, 2)
Environment.act(): Primary agent has reached destination! Total reward: 9.0
Simulator.run(): Trial 96
Environment.reset(): Trial set up with start = (4, 4), destination = (7, 2), deadline = 25
RoutePlanner.route_to(): destination = (7, 2)
Environment.act(): Primary agent has reached destination! Total reward: 5.5
Simulator.run(): Trial 97
Environment.reset(): Trial set up with start = (3, 1), destination = (8, 6), deadline = 50
RoutePlanner.route_to(): destination = (8, 6)
Environment.act(): Primary agent has reached destination! Total reward: 10.0
Simulator.run(): Trial 98
Environment.reset(): Trial set up with start = (3, 4), destination = (4, 1), deadline = 20
RoutePlanner.route_to(): destination = (4, 1)
Environment.act(): Primary agent has reached destination! Total reward: 6.0
Simulator.run(): Trial 99
Environment.reset(): Trial set up with start = (1, 4), destination = (7, 2), deadline = 40
RoutePlanner.route_to(): destination = (7, 2)
Environment.act(): Primary agent has reached destination! Total reward: 14.0
```