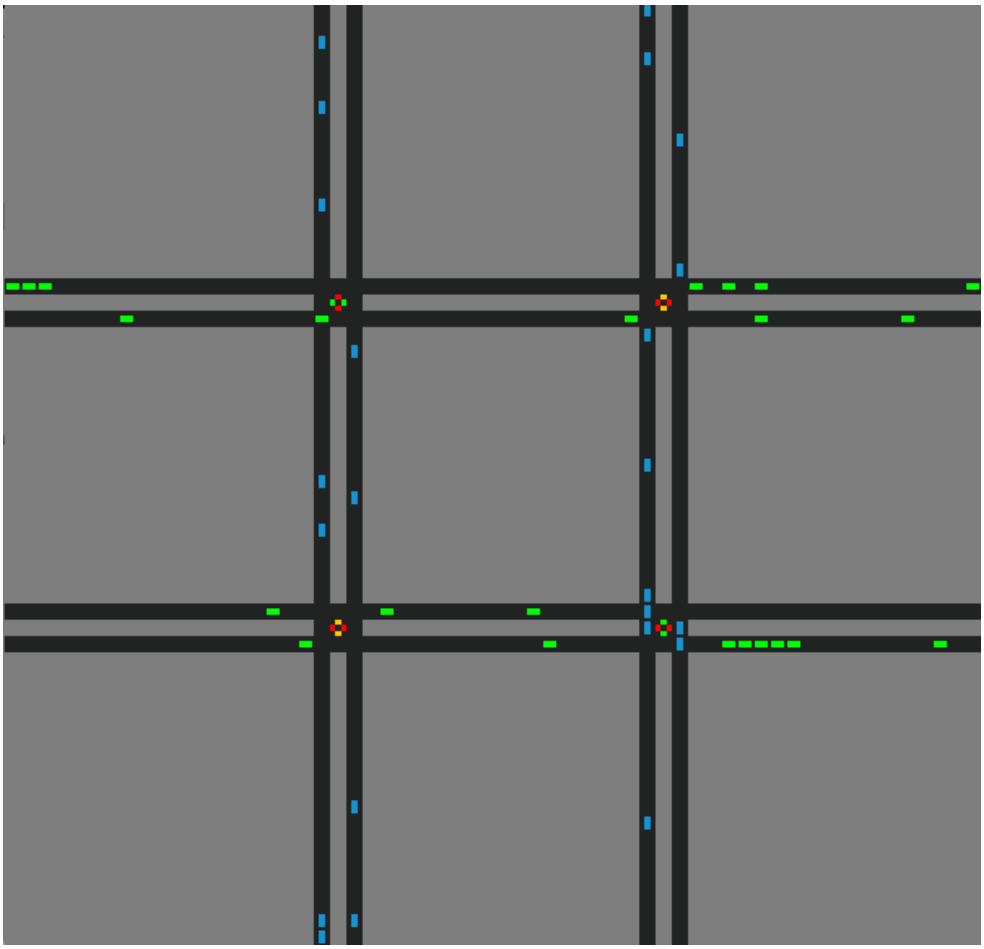
Traffic Lights Controller

Topic 10: Reinforcement Learning

COMP9417, Assignment 2

Beth Crane, Gill Morris, Nathan Wilson

Video at: https://dl.dropbox.com/u/1192047/Traffic%20Lights.mov



Introduction

Traffic lights are an important part of daily life; they control the efficiency of road travel, determining how many seconds are spent wasted in frustration at a badly designed system.

Creating an optimally functioning traffic light, which at every time step has the least number of cars stopped as possible, will be economically desirable and practically applicable. The learning done on this problem will also be relevant to scheduling problems of a similar nature.

Our task was divided into 4 parts:
1. Simulate the traffic generation, flow and traffic lights
2. Display the simulation on the screen
3. Control the lights using a fixed change time of 10 time-steps
4. Use a reinforcement learning algorithm to improve on 3.

Our project took a reinforcement learning approach, aiming to find a suitable way of defining a state and calculating a reward for it such that we could teach the machine to follow the optimal path at every step.

We were given an initial state, reward function and q-learning parameters, and once we had the system working with those we experimented with modifying them all. We tested each as we went, adding in parameters to adjust the density of the traffic in order to assist with this, and concluded that a slightly modified version of the original settings was actually the most optimal for learning.

State:
- closest car position from intersection for horizontal road (0-8, 9 if no cars)
- closest car position from intersection for vertical road (0-8, 9 if no cars)
- light setting (0 green horizontal, 1 green vertical, 2 amber)

Reward:
-1 if a car is stopped at a red light on either road, 0 otherwise.

Q-Learning Parameters:
- Discount factor: gamma = 0.9
- Learning rate: alpha = 0.7
- Epsilon-greedy exploration 0.1

Implementation

The problem of controlling a set of traffic lights is not an overly hard theoretical problem, and is reasonably simple to test. As such the bulk of our work came down to implementation.

Scalability and extensibility was a core focus of ours; we were aware of the potential for expansion, and we wanted to ensure we could build upon the project without breaking the existing functionality. We decided to use Java for the project, for modulisation, and then generalised each component as much as possible, to this end.

We prioritised getting a console view of the traffic lights working first, in order to enable us to see what was happening and debug any problems. Once we had the system set up with the recommended 10 second switching, we set to introducing the reinforcement learning.

Our learning takes place with reference to the road and a specific traffic light. We tried passing in all the cars on the map that were stopped, to see if the global effect would be an effective reward measure, but ended up performing local operations on each light. Each light then contributes to the q-values that are built up in the learner, so that the learning process is actually sped up with the addition of more lights.

At each time step an action is selected for each traffic light via the learner - picking the optimal algorithm (1-epsilon)% of the time, and a random action the rest. The traffic lights are updated, the cars are adjusted based on their velocities, new cars are spawned onto the map and ones off the edge of roads are deleted. Once this has happened the update function is called for the learner - using the reward of this new state to update the q-Value of the previous one.

The number of cars that are stopped on the road at each time step, as well as the total number on the road, are tallied for use as a performance measure.

Our q-values are stored in a hashTable to maximise memory efficiency, and we concluded early on that we had no need to store or create a ‘state’ object - rather we could reference it merely as the index of a cell. We experimented with different definitions of a state, trialling:

- ClosestCar Horizontal Roads, ClosestCar Vertical Roads, Light On/Off
- NumCarsStopped Horizontal Roads (max 9), NumCarsStopped Vertical Roads (max 9), Light On/Off
- CanCarMove Horizontal Roads, CanCarMove Vertical Roads, NumCarsStopped Horizontal Roads (max 9), NumCarsStopped Vertical Roads (max 9), Light On/Off

Each state was developed in order to test a different reward function - experimenting with -1*numCarsStopped, -20 for congested intersections, and whether to give a positive reward for cars moving through the intersection. After much testing and experimenting we concluded that the original state and reward function pair was the most effective, and reverted back to that.

The system supports multiple traffic lights, amber lights and different traffic densities.

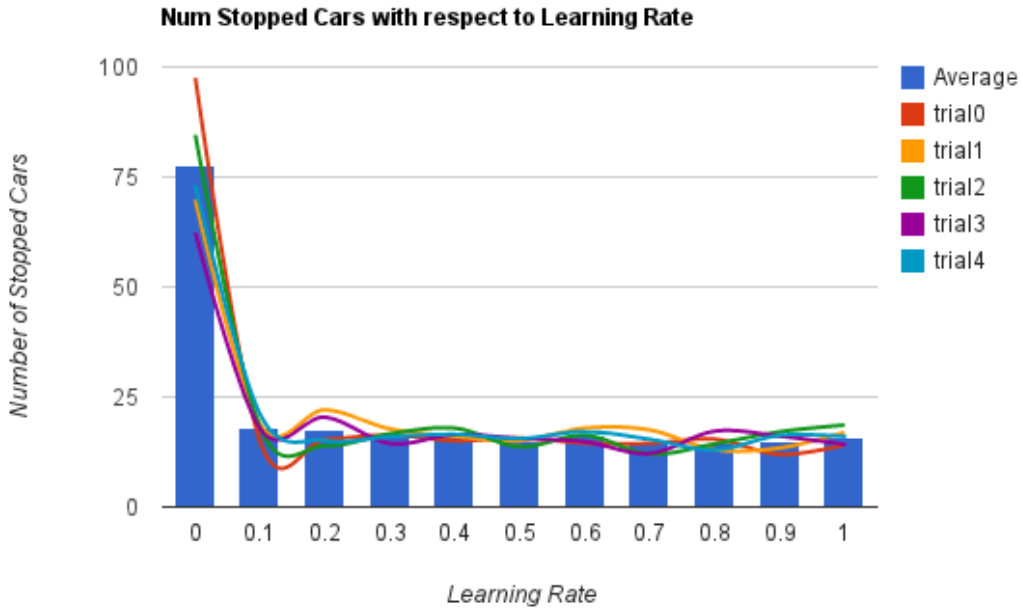
Experimentation

Parameters

We tested all our parameters with the other values set to the initial ones - learning rate: 0.1, reward discount: 0.9, epsilon-greedy exploration: 0.1

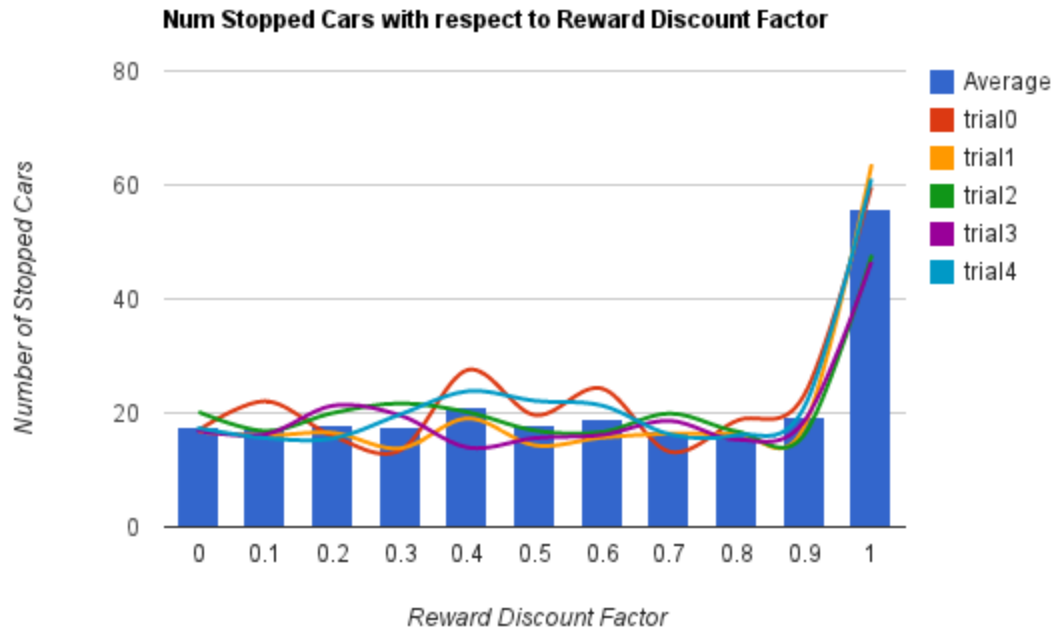
Learning Rate

We found that the learning rate was marginally more optimal at 0.7, up 0.6 from the original 0.1, but that the difference between everything aside from 0 was not statistically significant.



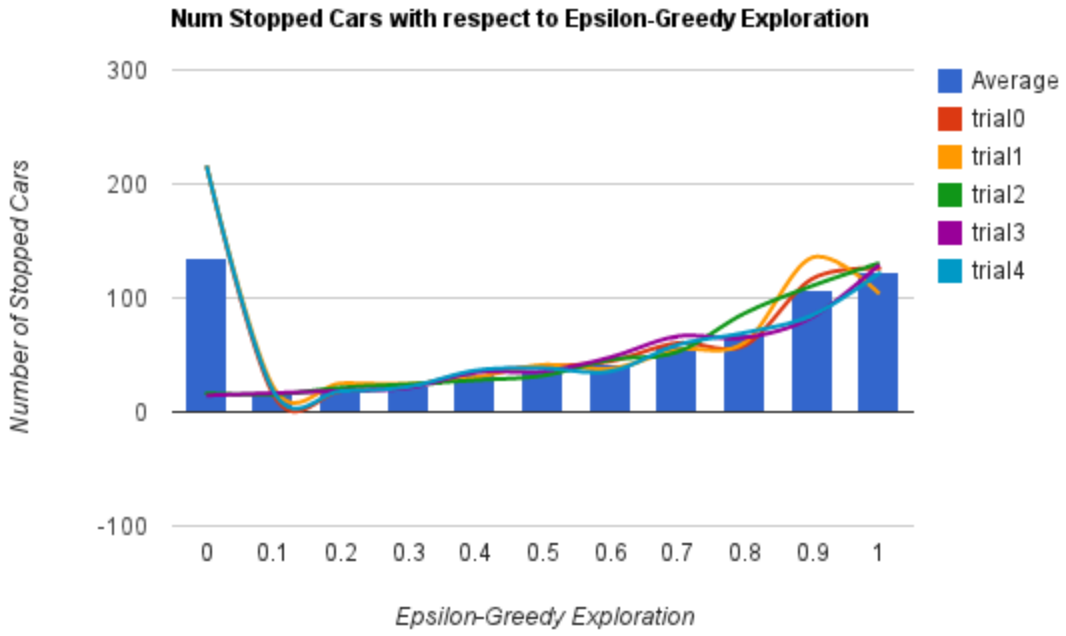
Discount Factor

We found that the reward discount factor was most optimal at 0.9 - where we originally had it.



Epsilon-Greedy Exploration

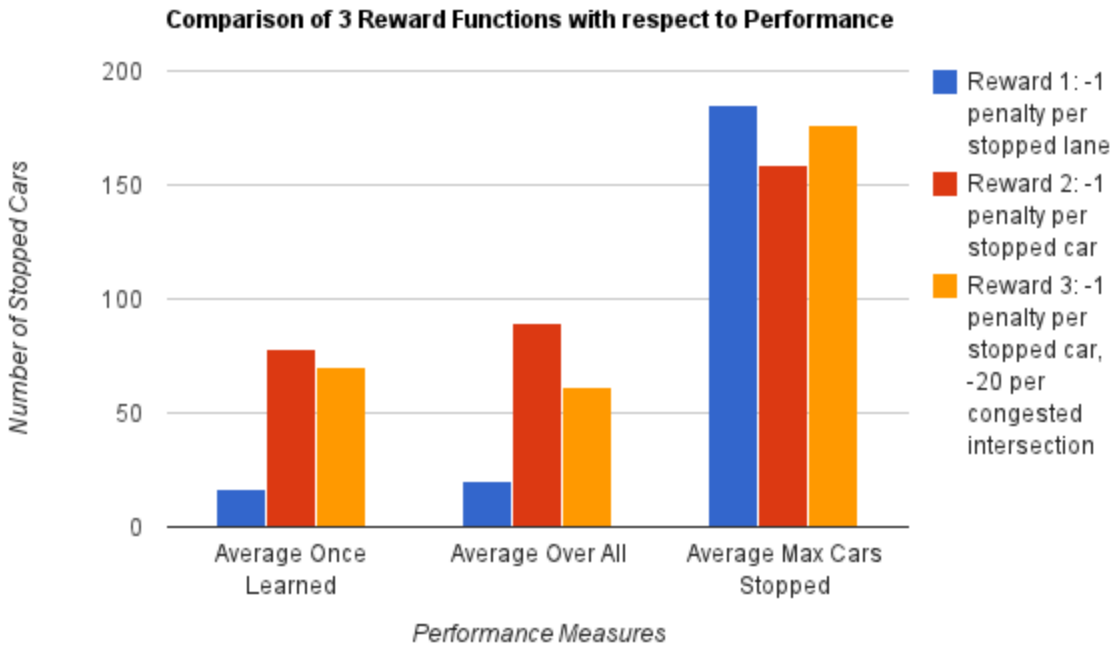
We found that the epsilon-greedy exploration percentage was most optimal at 0.1 - where we originally had it.



Results

For our experimentation, we chose a ‘reward’ function and a traffic density. We then spent over 100,000 iterations learning on these parameters. We repeated this process 10 times for each combination of parameters, the results are below;

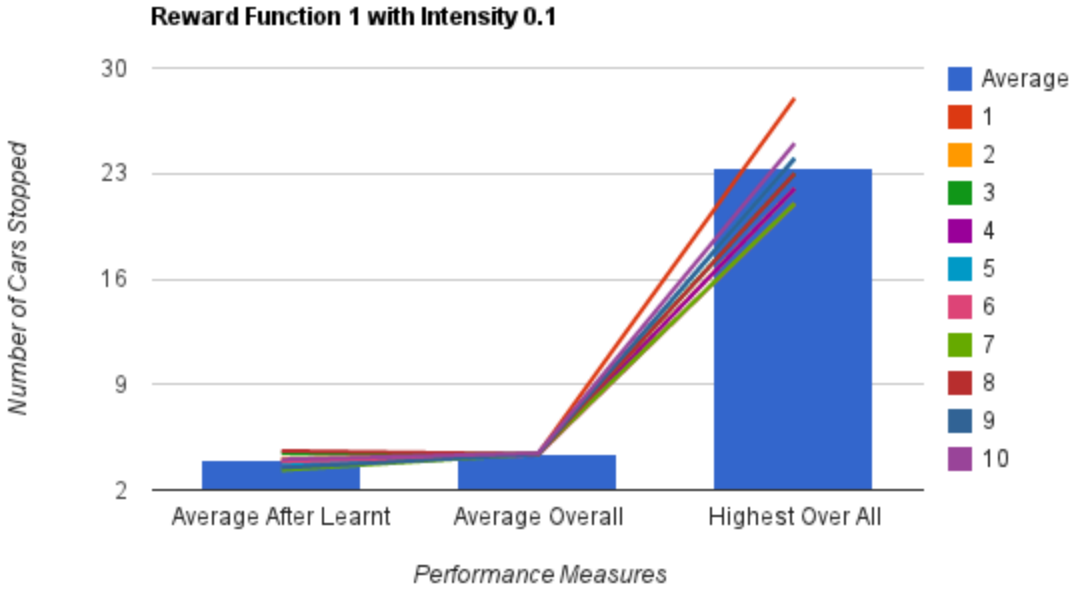
The first reward, the most simple one and the one that we originated with was the most effective by far. After experimenting on 10 iterations per reward we came to the following results, and hence proceeded with Reward 1.

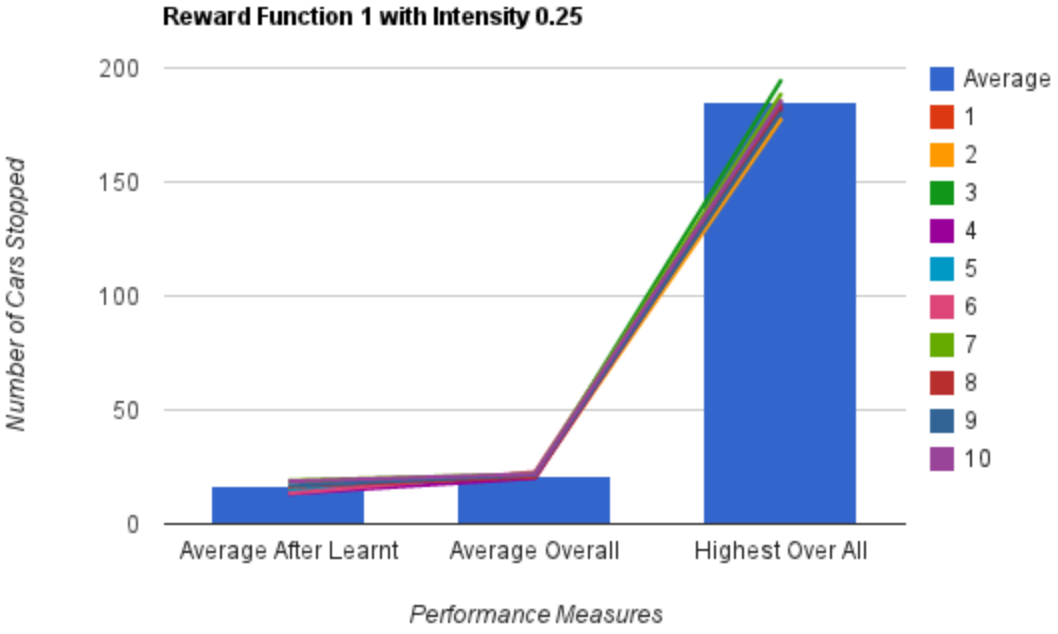


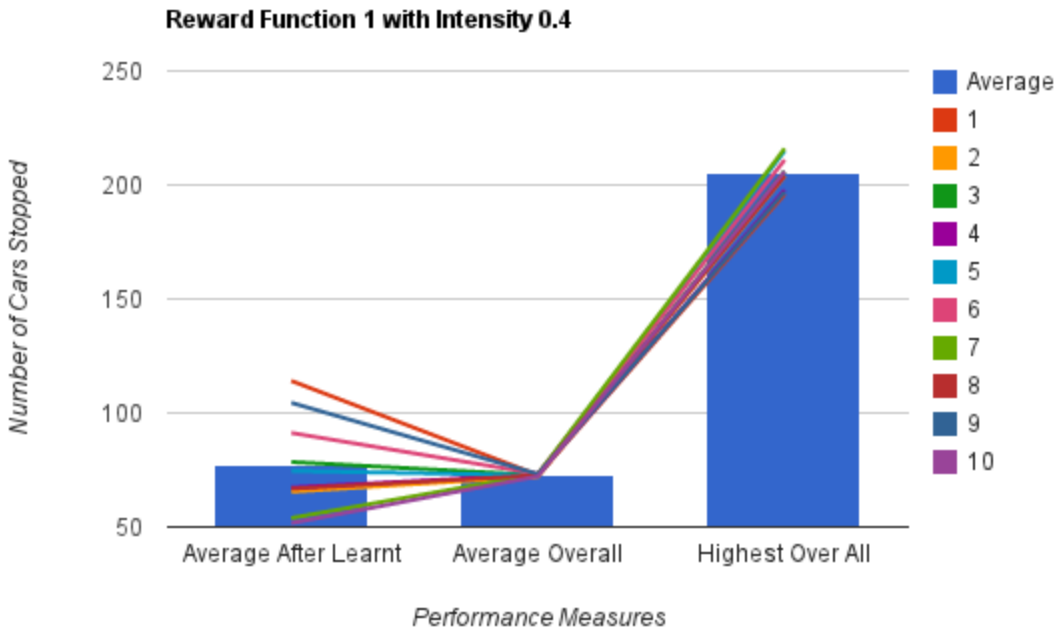
Having picked the reward function we experimented with traffic intensity for it specifically, to see how it handled it.

Our system has learnt to control traffic lights with

- an average of 4 cars stopped per second of 0.1 intensity traffic
- an average of 16 cars stopped per second of 0.25 intensity traffic
- an average of 77 cars stopped per second of 0.4 intensity traffic


Reward Function 1 with Intensity 0.1
30
23
16
9
2
Number of Cars Stopped
Average After Learnt
Average Overall
Highest Over All
Performance Measures
Average
1
2
3
4
5
6
7
8
9
10


Reward Function 1 with Intensity 0.25
200
150
100
50
0
Number of Cars Stopped
Average After Learnt
Average Overall
Highest Over All
Performance Measures
Average
1
2
3
4
5
6
7
8
9
10


Reward Function 1 with Intensity 0.4
250
200
150
100
50
Number of Cars Stopped
Average
1
2
3
4
5
6
7
8
9
10
Average After Learnt
Average Overall
Highest Over All
Performance Measures

References

Mitchell, T. "Machine Learning". McGraw Hill, 1997.

Hengst, B. Advanced Reinforcement Learning Notes. 2012.
http://www.cse.unsw.edu.au/~cs3411/12s1/lect/19_AdvancedRL4.pdf