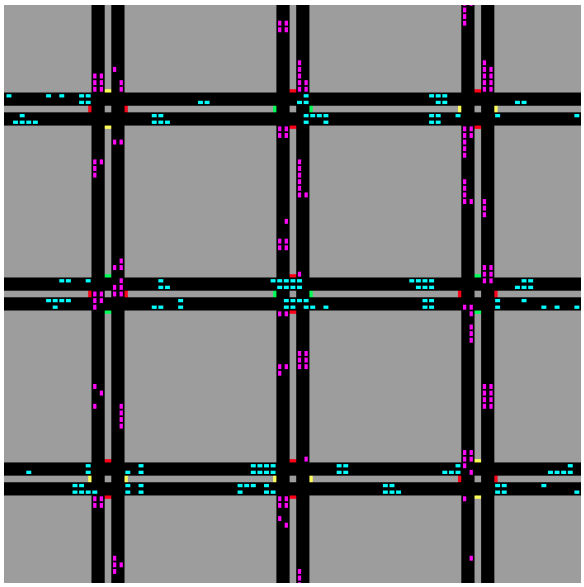
Reinforcement Learning for Intelligent Traffic Light Control

COMP9417 Machine Learning Project

Introduction

As some of the most populous cities in today's world continue to grow, they will need to either expand their existing road networks or improve road infrastructure to accommodate more traffic. One of the most effective ways to alleviate congestion and allow more road users is intelligent traffic light control. Constant phase control implementations may be effective enough when configured correctly for the given environment, but are rarely optimal. More modern implementations may incorporate the use of physical actuators, such as weight sensors or induction loops beneath road surfaces, which allow basic logic to increase the efficiency of the traffic light control. However, the addition of these physical installations would be time-consuming, expensive, and would not be able to dynamically adapt to changing environments or traffic situations. The purpose of this study is to design an intelligent traffic light control system, and evaluate its performance in a simulated traffic environment. The specific method used for learning in this study is the Q-learning algorithm with epsilon greedy exploration.



Video of Simulation at https://

Related Work

Over recent years there has been consistent ongoing research in adaptive traffic control, especially with the use of temporal difference reinforcement learning methods such as Q-learning. A study conducted by Steingrover has demonstrated the practicality of deploying reinforcement learning traffic control systems in networks of sixteen intersections which share a common knowledge base (Steingrover 2005). When each local agent is provided full perfect information about the entire network and other agents' decision processes Q-learning has been shown to perform especially well. Many other recent studies have utilised simulations to examine the effectiveness of Q-learning or other reinforcement learning algorithms in traffic light control, but it is not clear whether these theoretical concepts will be able to be deployed or tested in the real world anytime soon (Abdulhai 2003).

Mannion actually writes about the inherent difficulties associated with migrating these reinforcement learning traffic control systems from a virtual simulated world to the real physical world, and other practical differences and concerns (Mannion 2015). Providing learning agents with perfect real-time information about the world around it is relatively simple to do within a simulation, but attempting to do the same in the real world with today's technology would require a complex array of sensors and other hardware systems which would be very costly to calibrate, operate and maintain.

Implementation

The traffic control simulation software used in this study is self-coded using Python 3.5. The source code has been provided along with this study, and the only external module required to run the program is Pygame. The simulated environment involves numerous four-way intersections with traffic moving in both directions in multiple lanes. Each intersection allows traffic to flow in either the horizontal or vertical direction, but not both. Traffic can flow when the respective light signal is green, and is stopped when the light signal is red. When switching from green to red, an amber signal will be temporarily given to allow for cars that are currently travelling within the intersection to complete their trip to the other side. The number of intersections and number of lanes can both be changed in order to evaluate the performance of the reinforcement learning algorithm. The function which generates new cars for each lane is given by:

$$f(t) = True\ if\ t\ mod\ \left[rand\big([0,10)\big) + 10 - x\right] == 0$$

where x is the intensity factor ranging from 0 to 10, t is the timestep, and a new car is generated when $f(t)\ is\ True$.

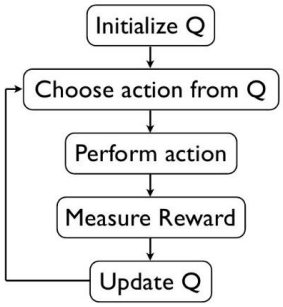
For a video of the simulation in action, please go to https://

Q-learning is a model-less temporal difference control algorithm developed by Watkins in 1989, which attempts to learn 'quality' values $Q(s,a)$ for each action a given state s within the state-space (Watkins 1989). By exploring states, actions, and rewards, it is able to estimate the true optimal action-selection function for any finite Markov decision process. It has been proven to converge, in certain specifications, to the optimal action-selection policy after each state-action pair (s,a) has been visited an infinite number of times. The one step update of the implemented Q-learning algorithm is:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

where s is the current state, s' is the future state reached by performing action a in the current state s, $Q(s,a)$ is the value of taking action a in the current state s, $\max_{a'} Q(s',a')$ is the maximum of the values of taking all possible actions a' in the future state s', α is the learning rate, γ is the discount factor, and r is the future calculated reward.

In this study, the Q-learning algorithm operates as follows. First, a table Q which represents all of the state-action values $Q(s,a)$ is initialised with all values set to 0. At each iteration, to encourage exploration, the algorithm decides with probability $\varepsilon\lambda^t$ to completely randomise the action which is chosen given the current state s. With probability $1 - \varepsilon\lambda^t$, the action with the highest corresponding Q value is chosen. If there are multiple actions with the highest Q value, we randomly choose one of them. ε is the initial probability to randomise or 'explore', λ is the probability decay factor slightly less than 1 which allows for increased exploitation of the acquired knowledge, and t is the current timestep. The chosen action is then performed, and the reward measured. We then update $Q(s,a)$ for the state we had just considered and the action we had just performed using the update formula described above. We use the new state s' as the current state and repeat this process.



Each state within the Q-learning algorithm was defined using:

- Sum of all horizontal lanes heading East where each lane is equal to closest car position from intersection (0-8, 9 if no cars)
- Sum of all vertical lanes heading South where each lane is equal to closest car position from intersection (0-8, 9 if no cars)
- Sum of all horizontal lanes heading West where each lane is equal to closest car position from intersection (0-8, 9 if no cars)
- Sum of all vertical lanes heading North where each lane is equal to closest car position from intersection (0-8, 9 if no cars)
- Light setting (0 green horizontal, 1 green vertical, 2 amber)
- Delay setting (0-(3 + time needed for cars to cross from one side of intersection to the other))

I initially considered adding a new state dimension per lane, but determined it to be infeasible without some kind of dimensionality reduction, for example using function approximation, as the resulting table would be far too large to process.

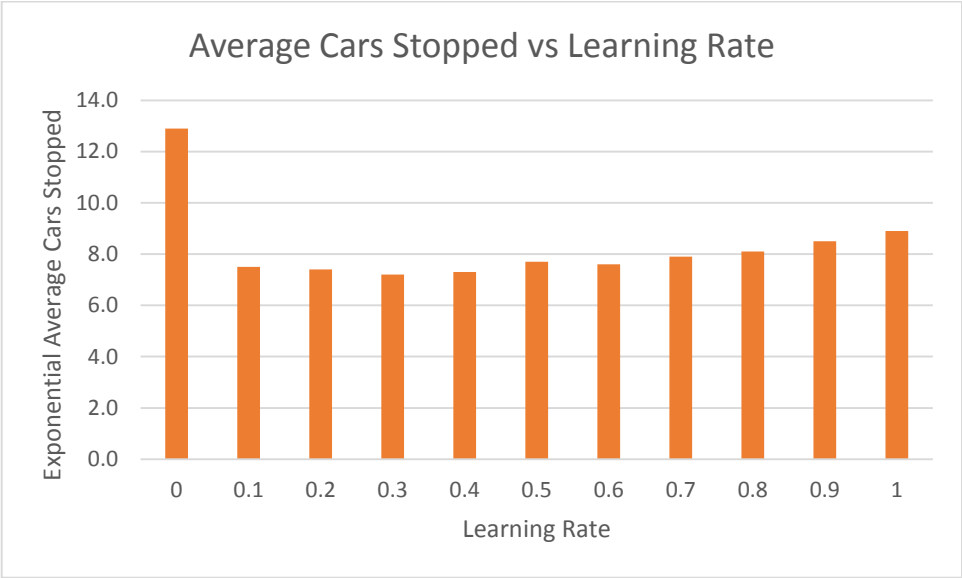
Results

The performance measure for all of my experiments is the sum of the number of queued (stopped) cars in all lanes within each timestep. I then further evaluate this performance measure as an exponential moving average with a smoothing factor $\alpha = 1 - e^{-\frac{1}{\tau}}$ where τ is the scaling constant and is equal to 300 iterations. The exponential moving average is calculated using the following equation:

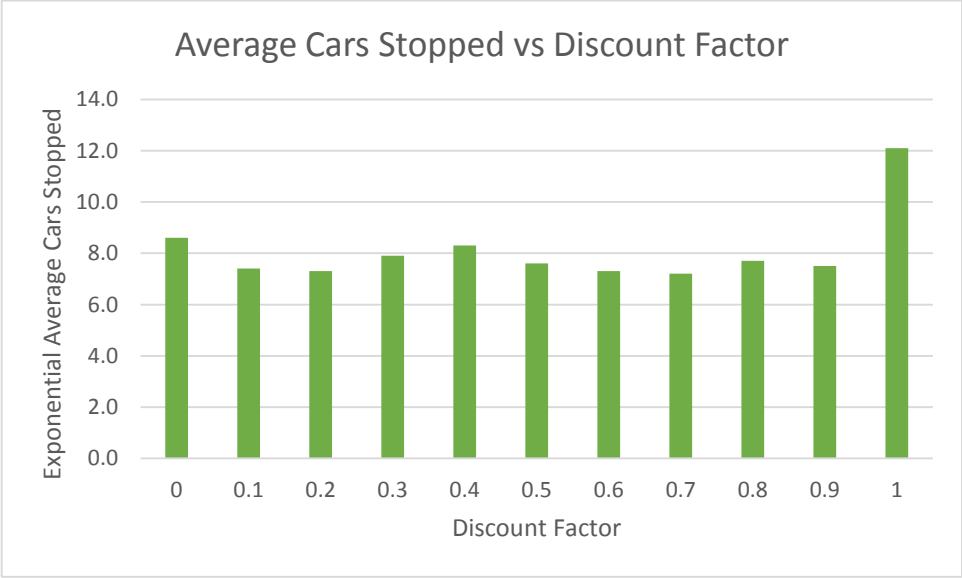
$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}$$

where s_t is the new exponential moving average at timestep t, x_t is the number of stopped cars during this timestep, and s_{t-1} is the exponential moving average at the previous timestep $t - 1$.

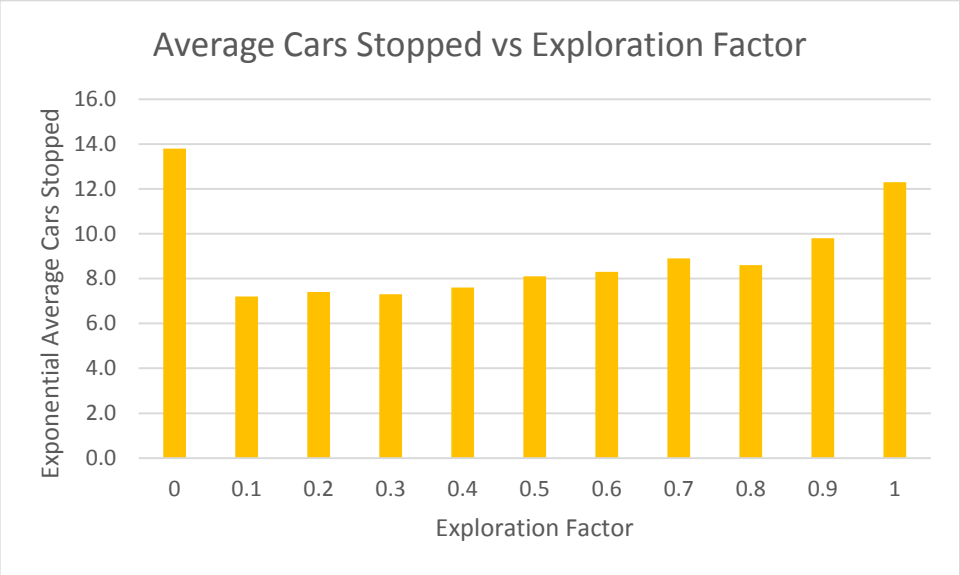
I began my experimentation with the Q-learning agent by playing around with the learning parameters. By first finding the optimal learning parameters for this particular use case, namely traffic signal control, we are able to generalise better overall results in subsequence experiments by sticking to this particular set of parameters. In order to test these parameters in the simulation, I decided to use 2 lanes to and from each intersection in every direction, and a total of 4 intersections (2 in each row and column), and averaged all results between 5 runs. The generating function intensity is 5, and the reward function chosen is the negative of the total number of cars stopped on roads leading to the intersection in either direction or lane. The results below followed from running the simulation for 10000 iterations.



The learning rate alpha determines how important new information is compared to older information. A learning rate of 0 results in the agent not being able to learn anything, and it is clear from the results that it does not perform too well in this case. As we approach a learning rate of 1, the agent considers recent information more and more. From the results above, it is obvious that a learning rate of 0.3 gives the best outcome on average.



The discount factor gamma is responsible for adjusting the significance of future rewards. A discount factor of 0 causes the learning agent to only consider the current rewards, and ignore all future rewards, and the opposite is true with a discount factor of 1, which results in a rather suboptimal performance. The discount factor which leads to the best performance is 0.7.

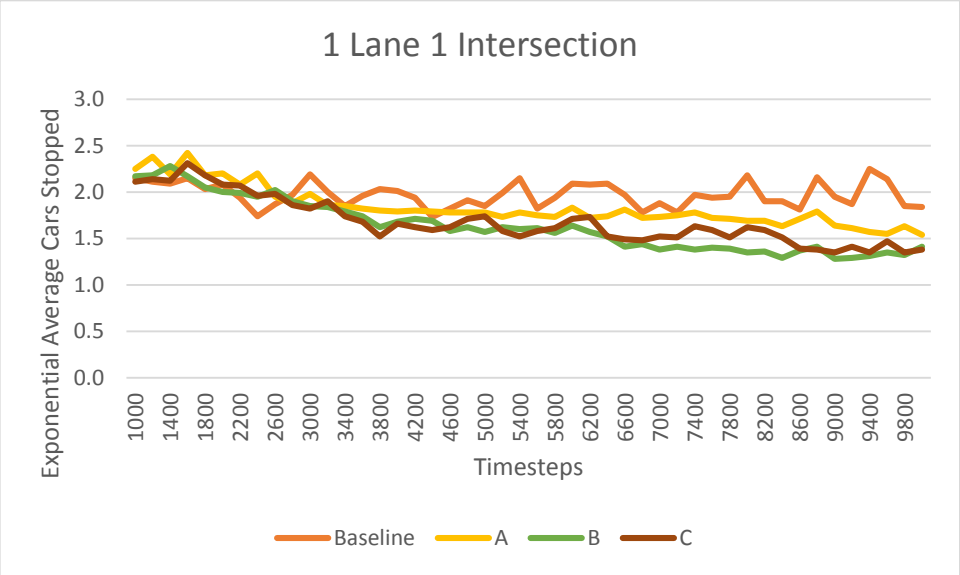


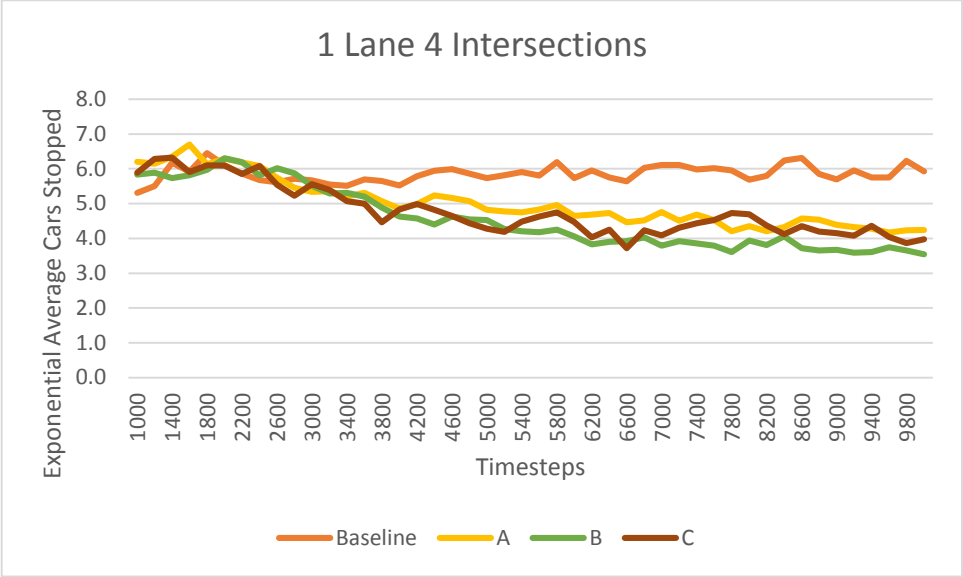
The exploration factor epsilon enables state exploration by sometimes allowing the learning agent to randomly choose a next action to take on instead of the highest valued one. An exploration factor of 0 means that there is no exploration conducted, which means that decent action paths previously taken are likely to be repeatedly taken again and again even if they are not part of the optimal route. As shown by the results, this clearly does not lead to optimal results. Neither does an exploration factor of 1, which causes a random exploration action to be selected every time. It is evident from the results that an exploration factor of 0.1 should be chosen for our further experiments. Note that in our simulation this exploration factor is actually decayed by a factor of 0.9998 to the power of the number of timesteps undertaken so far. This factor corresponds to lambda in the code, and the reason for its use is explained in the implementation section above.

Using these learning parameters as the default parameters, I proceeded to evaluate the performance of different reward functions. The reward (punishment) functions tested were:

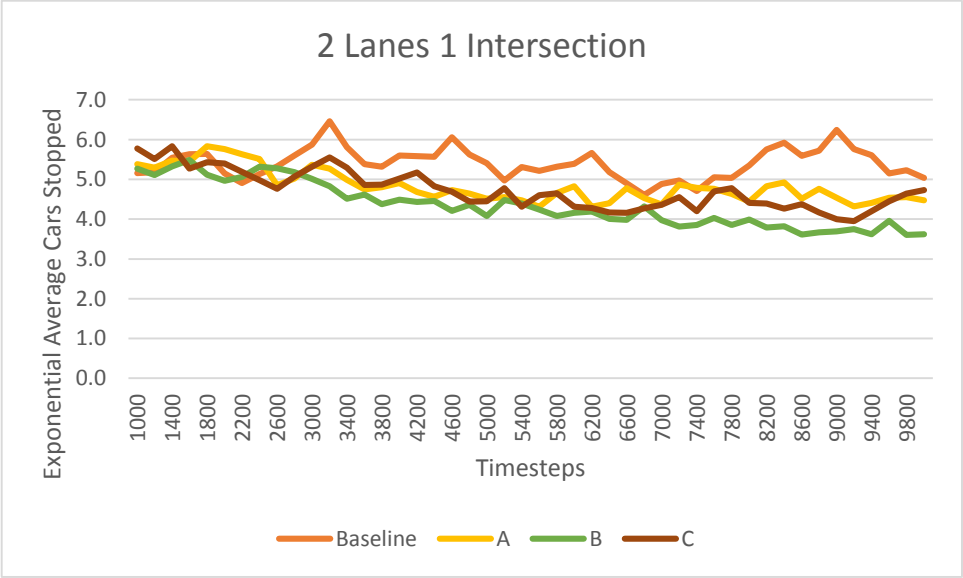
A: -1 for any stopped car on roads leading to intersection in either direction or lane
B: -1 * total number of cars stopped on roads leading to the intersection in either direction or lane
C: -1 * sum of squared number of cars stopped in each lane of roads leading to the intersection

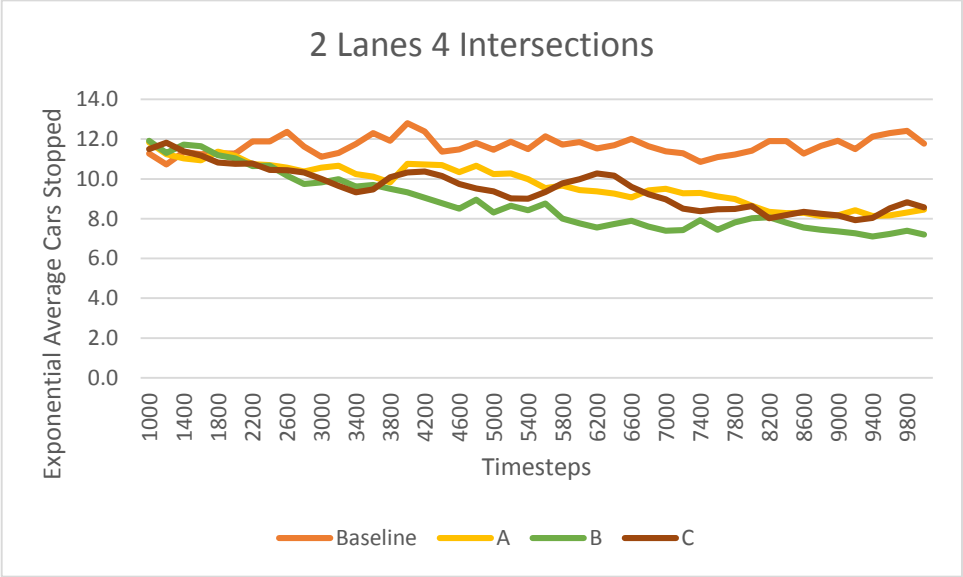
I plotted these reward functions in comparison to a baseline which consisted of a non-adaptive basic algorithm which alternated light signals every 10 timesteps. The generating function intensity was kept at 5 during all of the runs, and each simulation run lasted 10000 iterations.





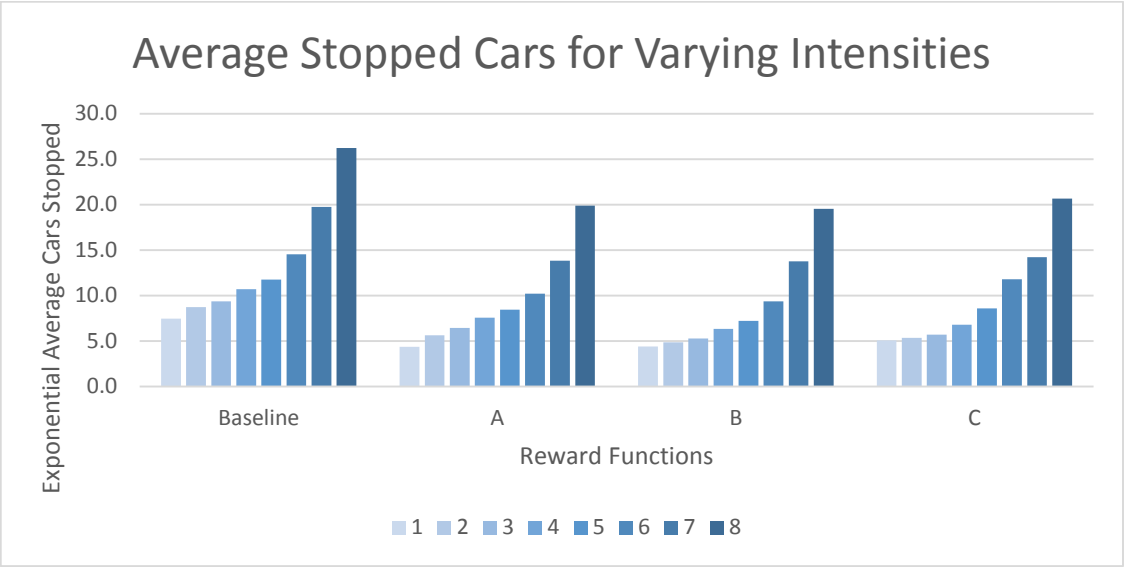
From the 1 lane results, it is clear that all of the Q-learning algorithms eventually performed better than the naïve constant light switching approach. Reward function B resulted in a consistently lower number of cars stopped than A, and was for the most part also better than C. Using a reinforcement learning algorithm to adaptively control traffic lights appears to have a valuable effect.





The 2 lane results tell a similar story, with B consistently outperforming A and C, and by extension the baseline too. It can be seen that incorporating 2 lanes instead of 1 has also made the graphs more jittery, reflecting additional volatility in the number of cars

stopped. It is now fairly obvious that using a temporal difference reinforcement learning algorithm such as Q-learning is definitely valuable in reducing traffic in a road network, at the very least in a simulated environment.



After running the simulation with varying intensities of the generating function using the various reward functions compared to the baseline, it becomes clear that the Q-learning traffic control system scales with increased traffic in a similar fashion to a non-adaptive implementation. This is perhaps indicative of the fact that even if a learning system manages to approach the optimal action function, the fundamental limit may still be an issue. This means that no matter how intelligent a traffic system is, if there is an extremely large amount of traffic on the roads, then extended delays will be unavoidable. This seems to suggest that intelligent traffic signals will only be a temporary solution to the real world problem of ever increasing traffic on our roads.

Conclusion

This study was completed in order to determine the feasibility and value of applying a model-less temporal difference reinforcement learning algorithm to traffic light control. The traffic lights at each intersection within the coded simulation were controlled by an independent Q-learning agent, which was able to work its way towards an optimal light switching action policy. This study further confirms that Q-learning shows promise for solving traffic light issues and improving overall efficiency in the real world. However, such an application is extremely sensitive to parameter changes, and together with inherent difficulties with moving such an algorithm out of a simulation into the real world makes it a non-trivial and costly solution to implement. And even after it has been eventually deployed in the real world, such a solution does not solve the traffic problem indefinitely, as there is still a fundamental limit to how much traffic can be alleviated.

Future Work

This study has tested a generic reinforcement learning algorithm on a simulation which can be expanded in a number of areas. After further work in the future, the simulation should be able to model lane changes and turns at intersections, as well as modelling of variable speeds and car spacing, perhaps at a continuous or non-discrete level. More stochastic processes should be implemented, including that of pedestrians crossing, weather effects, visibility changes, and/or modelling of accident aftermaths.

Various other temporal difference reinforcement learning algorithms, such as SARSA (Rummery 1994), should be experimented with, together with the use of eligibility traces to speed up the learning process. In order to process more complex, higher dimensional or even continuous state-spaces, one would later require the use of function approximation methods such as k-means clustering or neural networks to allow computation to be feasible.

Conducting experiments within a simulation can also be fairly limiting. Future work will need to be done in controlled real world experiments before we can safely deploy such algorithms out in the field. However, the physical instruments which would be required to allow intersections to gain access to the real time data required to operate a machine learning traffic light system would be rather costly and difficult to set up, thus making it uncertain whether we will be able to evaluate such systems in the near future.

References

Abdulhai, B., Pringle, R., Karakoulas, G.J. 2003, 'Reinforcement learning for true adaptive traffic signal control', *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278-285.

Mannion, P., Duggan, J., Howley, E. 2015a, 'An experimental review of reinforcement learning algorithms for adaptive traffic control', *Autonomic Road Transport Support Systems, Autonomic Systems*, Birkhauser/Springer.

Rummery, G.A., Niranjan, M. 1994, 'Online Q-learning using connectionist systems', Cambridge University Engineering Department.

Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B. 2005, 'Reinforcement learning of traffic light controllers adapting to traffic congestion', *BNAIC*, pp. 216-223.

Watkins, C.J.C.H. 1989, 'Learning from delayed rewards', *PhD thesis*, University of Cambridge England.