REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL CONTROL: COMPARISON WITH COMMERCIAL SYSTEMS

Alvaro Cabrejas Egea
The Alan Turing Institute, MathSys CDT, University of Warwick, United Kingdom
Raymond Zhang
The Alan Turing Institute, United Kingdom; Ecole Normale Superieure de Paris-Saclay, France
Neil Walton
The Alan Turing Institute, University of Manchester, United Kingdom

ABSTRACT

In recent years, Intelligent Transportation Systems are leveraging the power of increased sensory coverage and available computing power to deliver data-intensive solutions achieving higher levels of performance than traditional systems. Within Traffic Signal Control (TSC), this has allowed the emergence of Machine Learning (ML) based systems. Among this group, Reinforcement Learning (RL) approaches have performed particularly well. Given the lack of industry standards in ML for TSC, literature exploring RL often lacks comparison against commercially available systems and straightforward formulations of how the agents operate. Here we attempt to bridge that gap. We propose three different architectures for RL based agents and compare them against currently used commercial systems MOVA, SurTrac and Cyclic controllers and provide pseudo-code for them. The agents use variations of Deep Q-Learning (Double Q Learning, Duelling Architectures and Prioritised Experience Replay) and Actor Critic agents, using states and rewards based on queue length measurements. Their performance is compared in across different map scenarios with variable demand, assessing them in terms of the global delay generated by all vehicles. We find that the RL-based systems can significantly and consistently achieve lower delays when compared with traditional and existing commercial systems.

1. INTRODUCTION

Traffic Signal Control (TSC) can be used to ensure the safe and efficient utilisation of the road network at junctions, where traffic can change directions and merge, having to manage conflicting individual priorities with the global needs of the network. Traffic congestion has a major financial impact. A study by (INRIX 2019) shows that traffic congestion in 2019 cost £6.9 billion in the UK alone, with similar patterns being observed in other developed countries.

Cities around the globe are starting to explore the deployment of smart Urban Traffic Controllers (UTCs) that use real time data to adjust their stage schedule and green time duration. Traditionally, fixed time plans have been used. Those fixed time plans can be managed by systems that try to optimise the green time splits in a deterministic manner such as TRANSYT (Robertson 1969). These types of systems require costly site-specific knowledge of the traffic lights placement and typical demand profiles to be able to provide effective control. These methods are not easily scalable and deteriorate over time as the traffic demand changes as explained by (Bell and Bretherton 1986). With the development of induction loops, real time actuated UTCs were created in two variants: those that optimize single isolated intersections with systems such as MOVA (Vincent and Peirce 1988), and those that cover multiple intersections such as SCOOT (Hunt et al. 1982). To remedy the scalability problem, other systems based on local rules that generate self-organising area traffic controllers were developed, such as SurTrac (Smith and Barlow 2013), which solves a forward implementation of Dynamic Programming.

With the recent breakthrough of Deep Reinforcement Learning (DRL) on complex problems such as Atari games or Go (Mnih et al. 2013; Silver et al. 2017; Hessel et al. 2018), attention has turned towards adapting these approaches to generate industry-grade controllers for traditionally noisy and difficult to control systems such as TSC.

The purpose of this paper is to reproduce some of the results of the main and most successful RL approaches on intersections of increasing complexity, while comparing different architectures of DRL TSC agents, since, given the complexity of their implementation, most available literature only deals with a single class.

2. STATE OF THE ART

2.1 Previous Work

Reinforcement Learning (RL) is an area of Machine Learning that tries to imitate how biological entities learn. In it, an independent agent evolves in an unknown environment and learns how to perform a task for which no prior information is given based on its interactions said environment via a set of allowed actions. The agent aims to maximise the total reward signal it receives as a feedback for each of its actions.

RL methods have been applied to TSC in experimental setups. A good review of early methods can be found here (Mannion, Duggan, and Howley 2016). More recent works (Liang et al. 2018; Genders 2018; Liu, Liu, and Chen 2018) use neural networks as function approximators to avoid the dimensionality and computing time limitations of table based methods in large state-action spaces, and show that DRL TSC can be more efficient than some earlier methods.

While there is a variety of approaches in the literature that craft successful RL-based TSC systems, most of them do no present direct comparisons against commercial systems that are the concern of this paper. Gao et al. (2017) used a Convolutional Neural Network (CNN) and discrete cell encoding with a Target Network for a value-based agent. The results were compared against a fixed time and a heuristic system (longest queue first), finding RL to perform better. In Mousavi, Schukat, and Howley (2017), raw pixels were used as input for a CNN that parametrises two agents: a policy-gradient agent and a value-based agent. The variation in the delay between actions was used as reward, and while both agents were found to have near-identical performance, they were not compared against any reference system. Later, in Wan and Hwang (2018) a DQN using discrete cell encoding as state was implemented. It used a CNN architecture and a delay-based reward. It was compared against a fixed time system, obtaining better performance. Liang et al. (2018), used the same approach and included speed information in the state, using a reward based on variations on aggregated wait time for all vehicles. It compared against two different fixed-time systems, ranking better than both and providing some early evidence of the benefits of using Double DQN, Duelling architecture and Prioritised Experience Replay. In Genders and Razavi (2018) different state spaces were evaluated using a policy-gradient algorithm, including: occupancy and speed for each incoming road, queue and a measure of density of incoming roads, and Discrete Cell Encoding, partitioning the incoming roads into cells of fixed lengths, in this case 2.5 metres. Genders found little difference in the performance of the agents as a result of the change in the magnitudes observed, but it could be argued that a discrete cell encoding would greatly benefit from a Convolutional Neural Network (CNN) architecture in the agent, which is not used.

Regarding comparisons with established systems, in Stevanovic and Martin (2008) the authors compare SCOOT with a Genetic Algorithm-based control method. It is shown that SCOOT’s performance can be surpassed by more adaptive Genetic Algorithms that, in turn, tend to be less effective at learning than RL methods.

Despite these previous works, most results are hard or impossible to reproduce given the lack of industry standards in terms of simulators, performance metrics, the lack of availability of commercial algorithms for comparison and the fierce protection of their internal workings, and the lack of open-source code of proposed RL models.

2.2 Commercial Traffic Signal Control Optimisers

MOVA (Microprocessor Optimised Vehicle Actuation, (Vincent and Peirce 1988)) is a traffic controller designed by TRL Software. It aims to reduce delay on isolated junctions. The basic functioning of MOVA involves two induction loop detectors estimating the flow of vehicles in each lane. The system makes a virtual cell representation of the lanes within MOVA, and then it computes a performance index based on the delays calculated. If the index results lower than a certain threshold, the signal is changed to the next stage, otherwise the stage it is extended.

Surtrac (Scalable URban TRaffic Control, (Smith and Barlow 2013)) is an adaptive TSC system published in 2013. A real-world deployment on 20 intersections in Pennsylvania showed a performance improvement of 20-40% in performance. Its operation is decentralised, each intersection allocates green time independently and asynchronously based on incoming flows. Each intersection is controlled by a local scheduler and communicates projected outflows to the downstream neighbouring junctions, modelling vehicles as a sequence of clusters. This communication allows for locally balancing competing flows while creating larger "green corridors" by finding an optimal sequence such that the input jobs (ordered clusters) are cleared while minimising the joint waiting time of all vehicles.

3. METHODS

3.1 Traffic Control as a Markov Decision Process

The control problem can be formulated as a Markov Decision Process (MDP) defined in terms of a 5-tuple:

- A set of possible environment states $s \in \mathcal{S}$.
- A set of available actions to the agent $a \in \mathcal{A}$.
- A stochastic transition function $\forall a \in \mathcal{A}, \mathcal{T}_{s,s'}^{a} \triangleq \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$.
- A scalar real valued reward function $R(s_t, s_{t+1}, a_t)$ providing a performance measure to the transition generated by progressing into the state s_{t+1} after taking action a_t while in state s_t
- A discount factor γ that will provide the balance between immediate exploitation and approaches that aim to maximise returns over time.

Each time an action is required, the agent will receive a state vector s_t from the environment. Based on this state, the agent will produce an action a_t, which will be implemented in the simulator. The environment will then advance time until a next action is required, according to its dynamics represented by $\mathcal{T}_{s,s'}^{a}$. At this point, the next state s_{t+1} will be observable. Both states will be used to generate a reward r_t to serve as feedback to the agent. The agent will receive the state observation s_{t+1} and the cycle will start again.

In the case of TSC, the MDP is modelled as partially observable, following an unknown stochastic transition function. From here on, it is assumed that the traffic environment displays the Markov property, i.e. the process is memoryless, with the next state only depending on the current state and the action taken.

3.2 Reinforcement Learning

The goal of the agents will be to maximise their future discounted return, $G_t = \sum_{t=0}^{+\infty} \gamma^t \, r(s_t, a_t)$ with $\in [0,1]$. This is done by learning a policy π, parametrised by the weights θ of the neural network performing the approximation of the reward function and mapping states to actions: $\pi: f_1(s) \rightarrow a$. The reward function maps an action given a state to a reward scalar value: $r: f_2(s, a) \rightarrow \mathbb{R}$. The action-value function or Q-value is $Q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$. It represents the total episodic return by following policy π after being in state s and taking action a.

3.3 Value-based Reinforcement Learning Methods - DQN

Tabular value-based methods, such as Q-Learning, attempt to learn an optimal policy $Q_\pi^* = \max_\pi \mathbb{E}[r_t | s_t = s, a_t = a]$ by iteratively performing Bellman updates on the Q-values of the individual state-action pairs:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\big(y_t - Q_\pi(s_{t+1}, a_{t+1})\big), \tag{1}$$

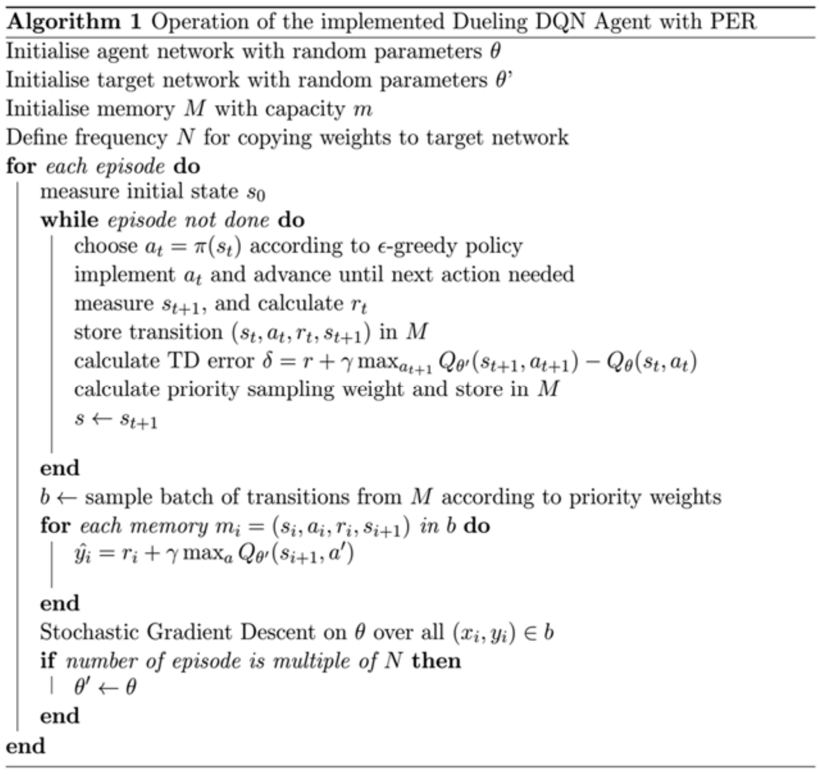
where α is the learning rate and y_t is the Temporal Difference (TD) target for the value function

$$y_t = r_t + \gamma \max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1}, \theta'). \tag{2}$$

Deep Q-Network (DQN) agents are an evolution of Q-Learning. The purpose of the agent is to find an approximation of Q_π^* by tuning the weights θ of a neural network. The agent keeps a second neural network, the target network, parametrised by the weights vector θ' which is used to generate the TD targets:

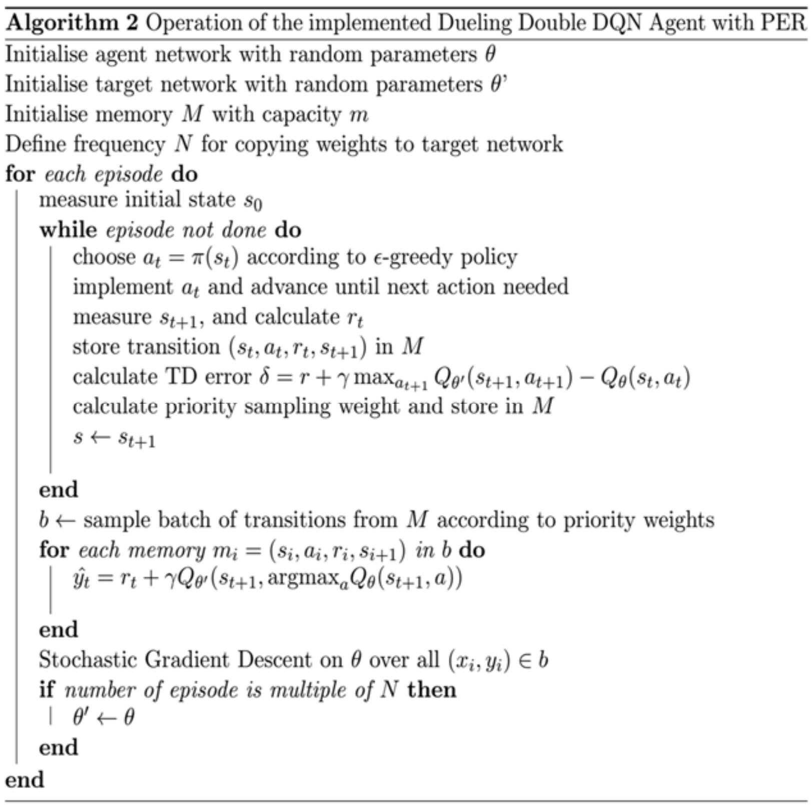
The experience replay memory is used to increase training stability, obtaining samples that cover a wider number of situations. It can also increase the data efficiency since the same transition can be used several times for gradient descent. Three additional modules have been applied to the basic agent to improve performance, Double Q Learning (Hasselt 2010), Prioritised Experience Replay (Schaul et al. 2016), and Dueling Architecture (Wang et al. 2015).

Two variants of the DQN agent have been implemented, being described on the algorithms displayed in Figs. 1 and 2. The agents implemented used the hyperparameters described in Fig. 4.



```
Algorithm 1 Operation of the implemented Dueling DQN Agent with PER
Initialise agent network with random parameters $\theta$
Initialise target network with random parameters $\theta$'
Initialise memory $M$ with capacity $m$
Define frequency $N$ for copying weights to target network
for each episode do
    measure initial state $s_0$
    while episode not done do
        choose $a_t = \pi(s_t)$ according to $\epsilon$-greedy policy
        implement $a_t$ and advance until next action needed
        measure $s_{t+1}$, and calculate $r_t$
        store transition $(s_t, a_t, r_t, s_{t+1})$ in $M$
        calculate TD error $\delta = r + \gamma \max_{a_{t+1}} Q_{\theta'}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)$
        calculate priority sampling weight and store in $M$
        $s \leftarrow s_{t+1}$
    end
    $b \leftarrow$ sample batch of transitions from $M$ according to priority weights
    for each memory $m_i = (s_i, a_i, r_i, s_{i+1})$ in $b$ do
        $\hat{y}_i = r_i + \gamma \max_a Q_{\theta'}(s_{i+1}, a')$
    end
    Stochastic Gradient Descent on $\theta$ over all $(x_i, y_i) \in b$
    if number of episode is multiple of $N$ then
        $\theta' \leftarrow \theta$
    end
end
```

Figure 1 – DQN Agent Pseudocode.



```
Algorithm 2 Operation of the implemented Dueling Double DQN Agent with PER
Initialise agent network with random parameters $\theta$
Initialise target network with random parameters $\theta$'
Initialise memory $M$ with capacity $m$
Define frequency $N$ for copying weights to target network
for each episode do
    measure initial state $s_0$
    while episode not done do
        choose $a_t = \pi(s_t)$ according to $\epsilon$-greedy policy
        implement $a_t$ and advance until next action needed
        measure $s_{t+1}$, and calculate $r_t$
        store transition $(s_t, a_t, r_t, s_{t+1})$ in $M$
        calculate TD error $\delta = r + \gamma \max_{a_{t+1}} Q_{\theta'}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)$
        calculate priority sampling weight and store in $M$
        $s \leftarrow s_{t+1}$
    end
    $b \leftarrow$ sample batch of transitions from $M$ according to priority weights
    for each memory $m_i = (s_i, a_i, r_i, s_{i+1})$ in $b$ do
        $\hat{y}_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \text{argmax}_a Q_\theta(s_{t+1}, a))$
    end
    Stochastic Gradient Descent on $\theta$ over all $(x_i, y_i) \in b$
    if number of episode is multiple of $N$ then
        $\theta' \leftarrow \theta$
    end
end
```

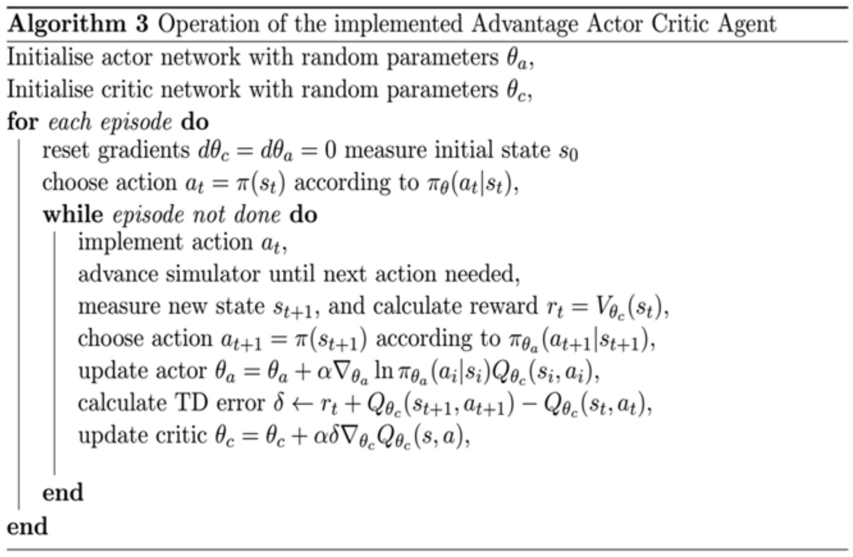
Figure 2 - DDQN Agent Pseudocode.

3.4 Policy Gradient Reinforcement Learning Methods - A2C

Policy Gradient in RL is based on the idea that obtaining a direct policy $\pi(s)$ mapping states to actions can be easier than estimating the value function or the state-action values. It has an added benefit in that it can learn stochastic policies, generating a probability distribution over the potential actions. The goal is to find the policy that maximises the reward. To do so one has to perform gradient ascent on the performance measure $J = \sum_a [Q(s_0, a)\pi(a|s)]$.
The Synchronous Advantage Actor Critic (A2C) method tries to reduce the variance in the policy method by combining the direct mapping from actions with the value-based approximation method. The goal is to learn an actor

$$\pi_\theta = \mathbb{P}_\theta[a_t = a | s_t = s], \quad \text{and a critic} \quad V_\pi^\theta(s) = \mathbb{E}_\theta[G_t | s_t = s], \tag{3}$$

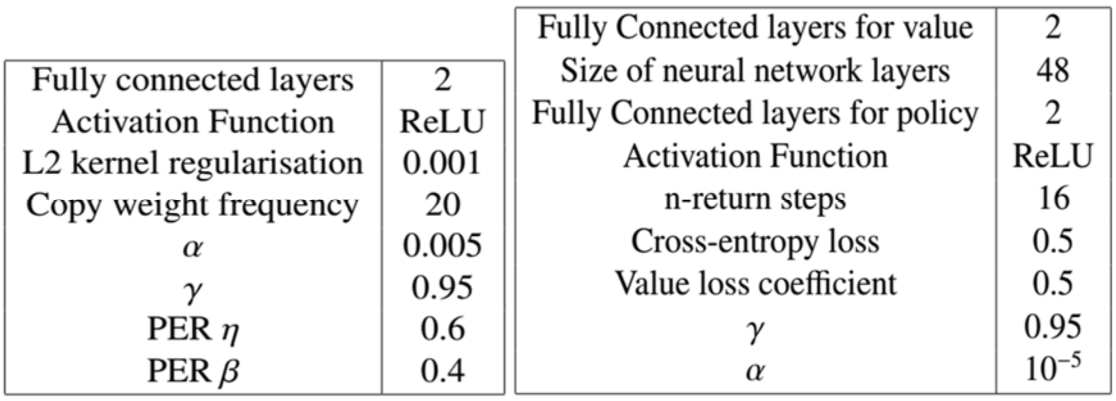
both of which are parametrised by the neural network weights vector θ.



```
Algorithm 3 Operation of the implemented Advantage Actor Critic Agent
Initialise actor network with random parameters $\theta_a$,
Initialise critic network with random parameters $\theta_c$,
for each episode do
    reset gradients $d\theta_c = d\theta_a = 0$ measure initial state $s_0$
    choose action $a_t = \pi(s_t)$ according to $\pi_\theta(a_t|s_t)$,
    while episode not done do
        implement action $a_t$,
        advance simulator until next action needed,
        measure new state $s_{t+1}$, and calculate reward $r_t = V_{\theta_c}(s_t)$,
        choose action $a_{t+1} = \pi(s_{t+1})$ according to $\pi_{\theta_a}(a_{t+1}|s_{t+1})$,
        update actor $\theta_a = \theta_a + \alpha\nabla_{\theta_a} \ln \pi_{\theta_a}(a_i|s_i) Q_{\theta_c}(s_i, a_i)$,
        calculate TD error $\delta \leftarrow r_t + Q_{\theta_c}(s_{t+1}, a_{t+1}) - Q_{\theta_c}(s_t, a_t)$,
        update critic $\theta_c = \theta_c + \alpha\delta\nabla_{\theta_c} Q_{\theta_c}(s, a)$,

    end
end
```

Figure 3 - A2C Pseudocode.



Fully connected layers	2
Activation Function	ReLU
L2 kernel regularisation	0.001
Copy weight frequency	20
α	0.005
γ	0.95
PER η	0.6
PER β	0.4

Fully Connected layers for value	2
Size of neural network layers	48
Fully Connected layers for policy	2
Activation Function	ReLU
n-return steps	16
Cross-entropy loss	0.5
Value loss coefficient	0.5
γ	0.95
α	10^{-5}

Figure 4 - DQN/DDQN and A2C Hyperparameters.

3.5 State, Action and Rewards of the agents

The experiments presented in the following sections all use the same descriptions for simulator state and reward calculation, although they differ in the number of actions available to them.

The state of an intersection of l lanes will be presented to the agents as a state vector $s \in \mathbb{R}^{l+1}$. Each component will contain the length of the queue of vehicles measured upstream from the traffic light in metres. The last component will be the numeric ID of the current stage that the agent is implementing.

While marginal improvements in performance can be obtained by using different variables for reward (Cabrejas Egea et al., 2020.; Cabrejas-Egea and Connaughton 2020b), as per the discussion of Heydecker (2004), queues can be a reasonable choice for states and rewards, being able to transmit useful information to the agent relative to the mean rate of delay of the system. Based on this, the reward after an action will be calculated as the negative sum of the length of the queues of all lanes immediately upstream from the intersection:

$$r_t = -\sum_l q_l^t \,. \tag{4}$$

The agent has a set of actions $\mathcal{A}$ that varies depending on the intersection being controlled. Once the agent chooses an action a, the stage corresponding with the ID of a is implemented. Green stages are set to a minimum of 6 seconds. Once this time has passed, the agent is requested a new action. If the agent chooses the same action again, the current stage is extended for a further 3 seconds. There are no inbuilt limitations as to how many times an agent can extend a stage, leaving it for the agents to learn. If the agent chooses a different action than the currently active one, a 3 seconds amber stage is implemented in the lights that were green, after which, the new stage is implemented.

3.6 Agent Benchmarking

In order to compare the agents’ performance, a testing framework was defined. For each model, a demand profile will be created, following the shape found in a typical day as described by using the methodology introduced in (Cabrejas Egea, De Ford, and Connaughton 2018) and expanded in (Cabrejas-Egea and Connaughton 2020a). The profile will be split on 10 segments of length 6 minutes. Each of these segments will correspond with a level of demand. The levels of demand are obtained by setting out what will be the maximum demand the intersection will suffer, setting that magnitude to coincide with the peaks of the distribution that could be found on said typical day and are specified in each experiment’s section.

Random seeds are changed and updated after every simulation episode, training or testing. The quantitative metrics on which the system will be evaluated are the Global Cumulative Delay (accounting for any deviations from the maximum speed) and the Average Queue Length generated by all vehicles during the execution of the evaluation.

4. EXPERIMENTAL RESULTS

4.1 Experiment 1: Cross Straight

The first test is conducted on the simplest junction, shown in Fig 5. The junction referred to as Single Cross is composed on 4 lanes distributed in 4 arms coinciding with the cardinal directions of the model. The controller for the junction has two stages, a north-south stage and an east-west stage, and turning is not allowed. The testing aim was to perform an initial performance comparison of DRL algorithms against MOVA, SUTRAC, and a cyclic controller. Here the goal for the agent was to exert fine adaptive timing control while extrapolating, rather than using complicated transitions between stages that would rarely, if ever, appear in sequence in cyclic control. MOVA was configured using loop detectors set in accordance to its manual, the implementation of Surtrac follows the work of Xie et al. (2012), and the cyclic controller was set on a 56 second cycle following both the methodology presented in (Salter 1996) and a parallel optimisation process, reaching the same result.

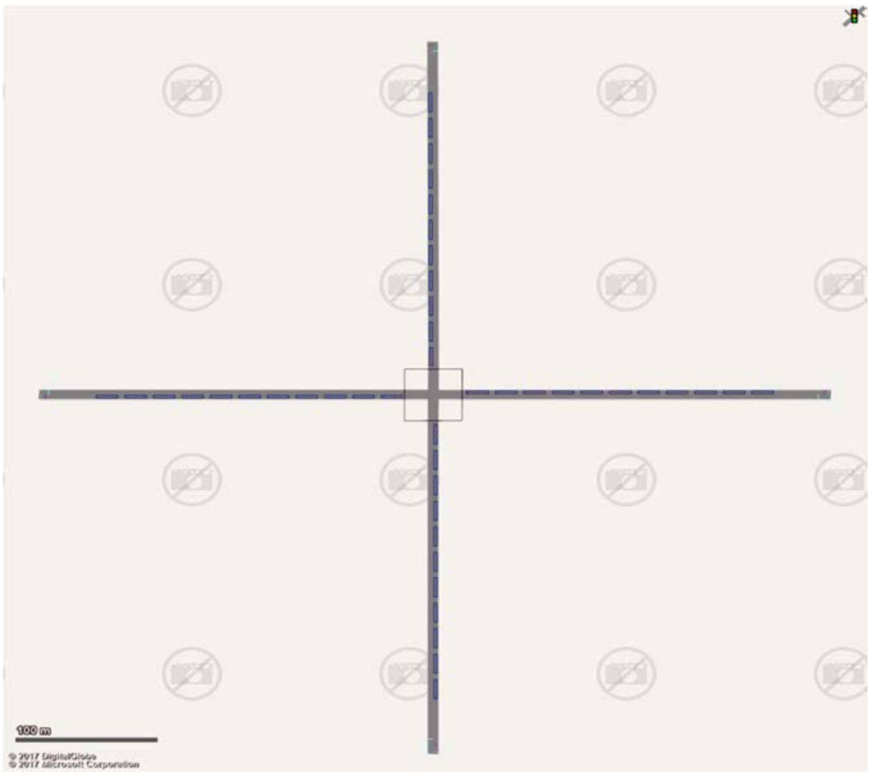


Figure 5 - Cross Straight Map.

The agents were trained using a fixed vehicle demand of 400 vehicles per hour on each of the incoming lanes. Both DQN variants were trained for 400 episodes, using an ϵ geometrically annealed from 1 to 0.001. The A2C agents were trained for 100 episodes until they converged. Best performing agents in each class were selected for benchmarking and evaluated in scenarios lasting one hour, as described in the previous section with the demands shown in Table 1. During evaluation, an average of 2120 vehicles are inserted in the model, with 2 peaks of demand of 3000 vehicles/hour for 6 minutes each. Figure 6 and 7 and Table 2 show the Global Cumulative Delay and average queue length for the network. As expected, the cyclic solution is outperformed by all adaptive controllers. The different controllers are on a par with a slight advantage for the DuelingDDQN which saves the community an average of 3000 seconds compared to MOVA on this hour of simulation, which represents on average 1-2 seconds per vehicle. RL agents also seem slightly more robust against changes in demand, producing lower slopes in the delay graphs in sections of extreme demand.

Time period [min]	**North**	**East**	**South**	**West**
0-6	200	200	200	200
6-12	400	400	400	400
12-18	900	500	900	500
18-24	1000	500	1000	500
24-30	700	500	700	500
30-36	500	700	500	700
36-42	500	1000	500	1000
42-48	500	900	500	900
48-54	400	400	400	400
54-60	200	200	200	200

Table 1 - Demand in vehicles/hour per cardinal direction over the benchmark in Cross Simple.

The cyclic controller resulted in saturated lanes during both peaks and queues in excess of 150 metres during a great part of the simulation. MOVA suffered two moments in which at least a sensor was saturated coinciding with the peaks in demand, however the queues were close to lengths of around 50 metres during the most part of the simulator. Surtrac followed a similar pattern, having a single lane saturated coinciding with the second peak in demand. RL agents as suffered no saturation in any of their lanes during the length of the evaluation. They all managed a more balanced distribution of queues in their respective lanes, displaying a higher ability to balance loads during peak times.

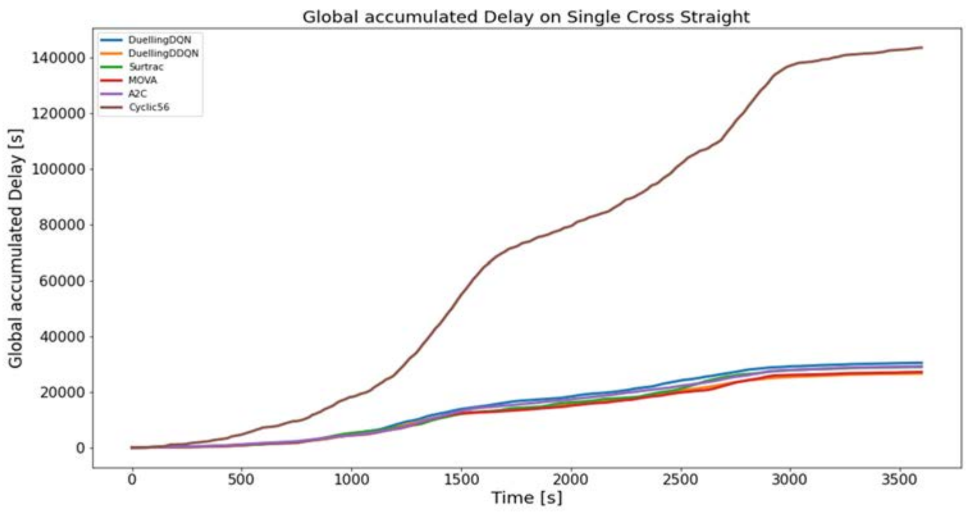


Figure 6: Global Cumulative Delay in Single Cross for DQN variants, A2C, Surtrac, MOVA and the reference Cyclic controller.

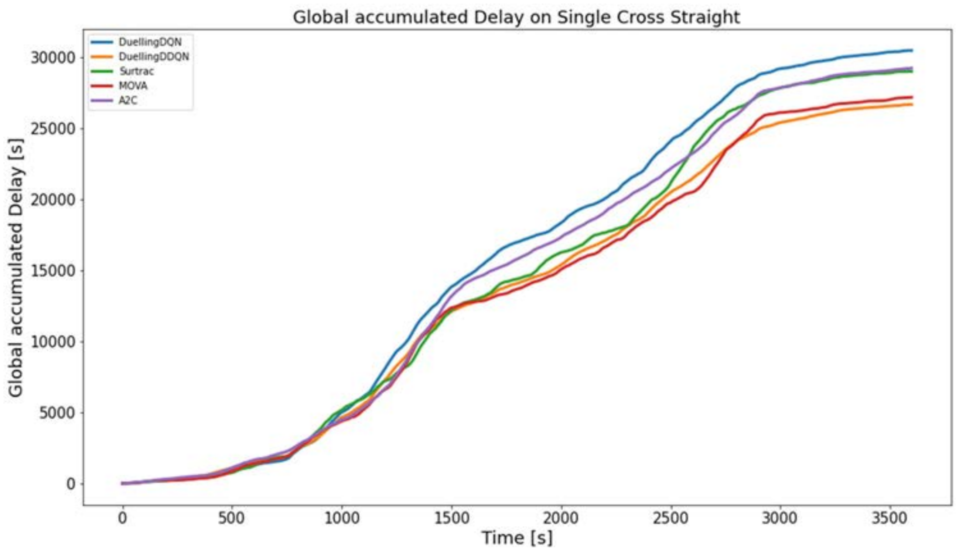


Figure 7: Global Cumulative Delay in Single Cross for DQN variants, A2C, Surtrac, MOVA and the reference Cyclic controller.

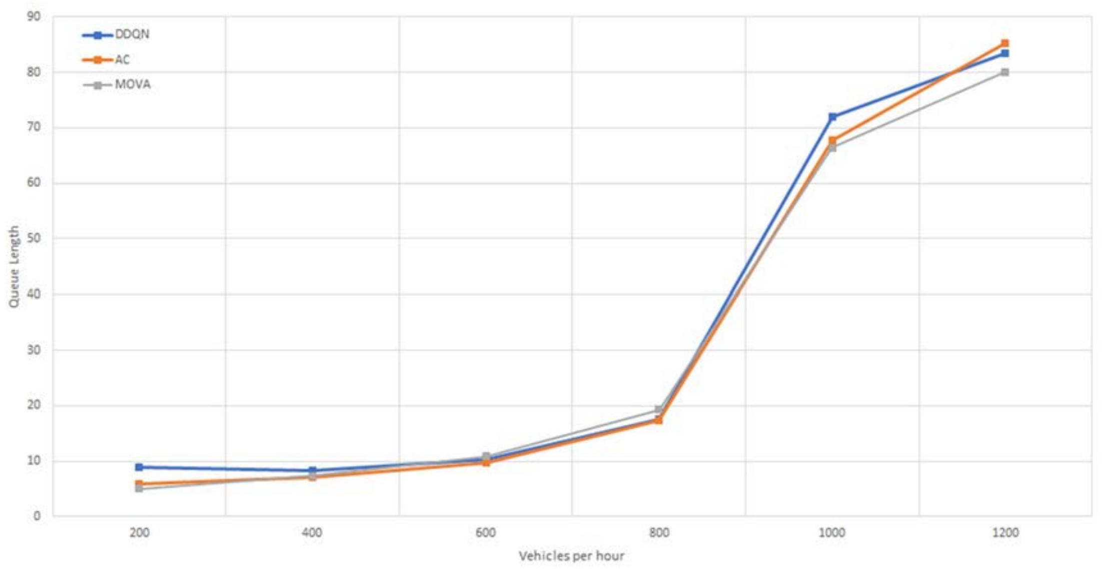


Figure 8: Queue length by demand level in Single Cross

When comparing the systems that obtain the best performance, RL agents and MOVA, across different constant demand profiles, we observe that MOVA obtains a small advantage at the extreme low and high demand levels, but this difference is overcome by the RL based systems in the middle demand levels. This gives another indication that while MOVA can obtain really good results, especially at constant demand, it is not optimal, having room for improvement in the control exerted while in situations of varying demand.

Controller	Cumulative Delay [s]	Average Sum of Queues [m]
Cyclic	143660.50	132.37
MOVA	27187.53	60.59
SURTRAC	29008.36	72.41
A2C	26382.14	56.07
DDQN	28303.94	50.11
DDDQN	21286.86	49.42

Table 2 - Cumulative Delay and Cumulative Stop Delay in seconds across Controllers on Single Cross Straight.

Because of the simplicity of this 2 actions intersection, there is not a lot of delay difference between adaptive UTCs. As it will be appreciated shortly, these results will change when we consider more complex junctions.

Given the difference in performance between the adaptive and cyclic controllers, which is expected to become greater on more complex intersections, and the increasing difficulty in setting them in large intersections, the cyclic controller will be omitted for the next examples. Given that the A2C agent has been clearly outperformed in this experiment by those based on the DQN architecture, the following experiments will focus on the performance of this last architecture compared with commercial systems.

4.2 Experiment 2: Cross Triple - 4 actions

This junction, as shown in Fig. 9, displays a much higher complexity than the intersection presented in the previous section. It is composed of 4 incoming links of 3 lanes each. In each incoming link, the left lane serves a dedicated nearside turning lane, the central allows for forward travel and the right lane allows for both offside turning and going straight. Due to limitations in how Vissim internally treats the queues, it is not possible to obtain a straightforward measurement of the lane queues in links that have more than one lane. To mitigate this, the first experiment was run with agents that would take 4 queue inputs (one for each incoming link), plus the state of the traffic signal as state input. The action set was consequently limited to 4 different actions, being allowed only those that set to green the 3 traffic lights serving the lanes of the same incoming link. This allows for turning vehicles but prevents more sophisticated stages from happening.

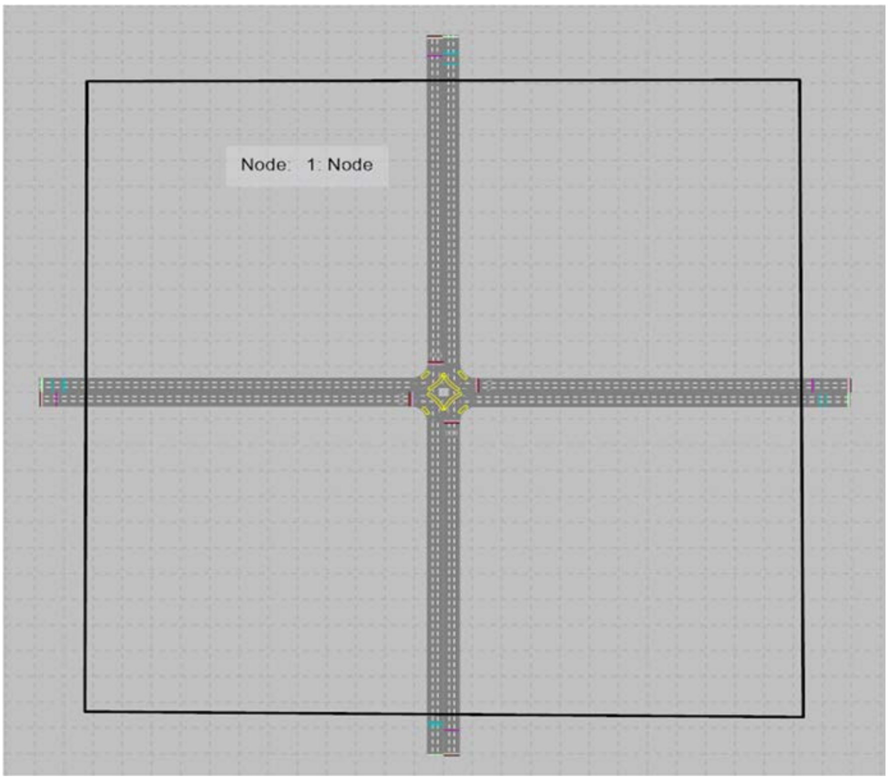


Figure 9 - Cross Triple Map.

The agents were trained on a fixed demand of 400 vehicles per hours on each incoming link. The DQNs were trained for 400 episodes of one hour of simulated time, with ϵ annealed geometrically between 1 and 0.001. A2C agents were trained for 100 episodes until they converged. During the hour of evaluation, the demand profile from the last experiment was used with a scaling factor of 1.5, an average of 3180 vehicles were introduced to the model, with 2 peaks of demand of 4500 vehicles/hour for 6 minutes each, as shown in Table 3.

As it can be seen in Figure 10 and Table 4 the UTC using MOVA performs poorly compared to the DQN-based agents. During this hour of simulation RL agents halve the cumulative delay, saving over 27 hours of travel time for all vehicles involved, an average of over 32 seconds of per vehicle. The length of the queues in those intersections controlled by RL agents during the test scenario were lower than the ones controlled by MOVA. Additionally, it can be seen that the agent using Dueling Q-Learning has a better performance than that Dueling Double Q-Learning.

Time period [min]	North	East	South	West
0-6	300	300	300	300
6-12	600	600	600	600
12-18	1350	750	1350	750
18-24	1500	750	1500	750
24-30	1050	750	1050	750
30-36	750	1050	750	1050
36-42	750	1500	750	1500
42-48	750	1350	500	1350
48-54	600	600	600	600
54-60	300	300	300	300

Table 3 - Demand in vehicles/hour per cardinal direction over the benchmark in Cross Triple.

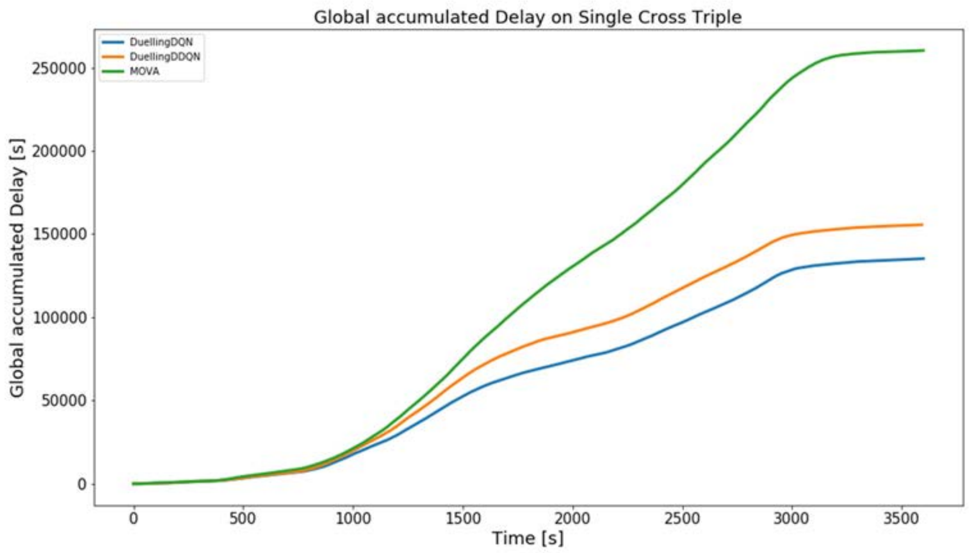


Figure 10 - Global Cumulative Delay in Single Cross Triple - 4 actions.

Controller	Cumulative Delay [s]	Average Max Queue Length [m]
MOVA	260257.65	179.27
DDQN	135220.91	153.58
DDDQN	155563.22	128.20

Table 4 - Cumulative Delay in seconds and Average Sum of Max Queue length in meters on Single Cross Triple - 4 actions.

4.3 Experiment 3: Cross Triple - 8 actions

In order to allow the use of a higher variety of stages in the controllers the map was reworked. All original lanes were partitioned into their own independent links, allowing extra space for lane changes. While these modifications allowed using information from all lanes in an akin manner to what modern sensors would achieve, due the limitations to lane changing, direct comparisons with Experiment 2 must be handled with care. Both models share name and rough geometry, but the layout of lanes is changed and so are the routing possibilities open to the vehicles.

The results presented below, use DQN agents taking 12 queue length inputs plus the state of the signal. Here, 8 different stages are available as represented in Fig. 11. No specific stage order is enforced, and the agents are free to change between any combination of stages.

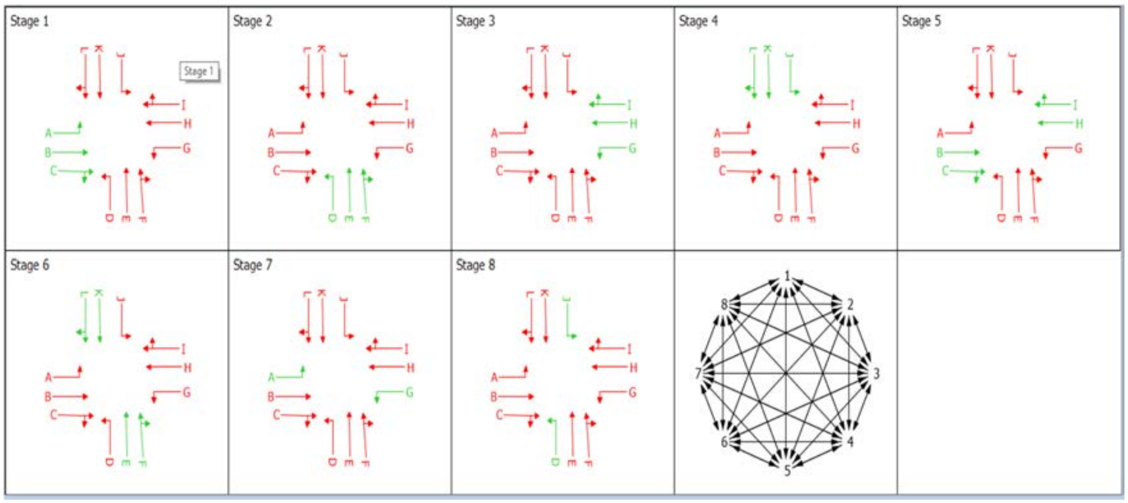


Fig 11 - Allowed stages of the Single Cross Triple model and allowed transitions between stages.

The RL agents display a similar, yet wider gap in performance with MOVA as in the previous experiment, with both classes benefiting from the increased actions pool. RL agents manage to generate about a third of the delay produced by MOVA. While this appears to be a great success, these results must be put into context. MOVA has a lot of internal parameters meant to be fine-tuned by a traffic engineer with site-specific knowledge. Our settings did produce a successful control loop, operating in line with what was expected of the configuration process. None of the RL agents has been fine tuned to the level that would be expected during commercial operation. The layers and neuron distribution weren't optimised, nor were the activation functions, meaning that the RL agents can still be improved upon.

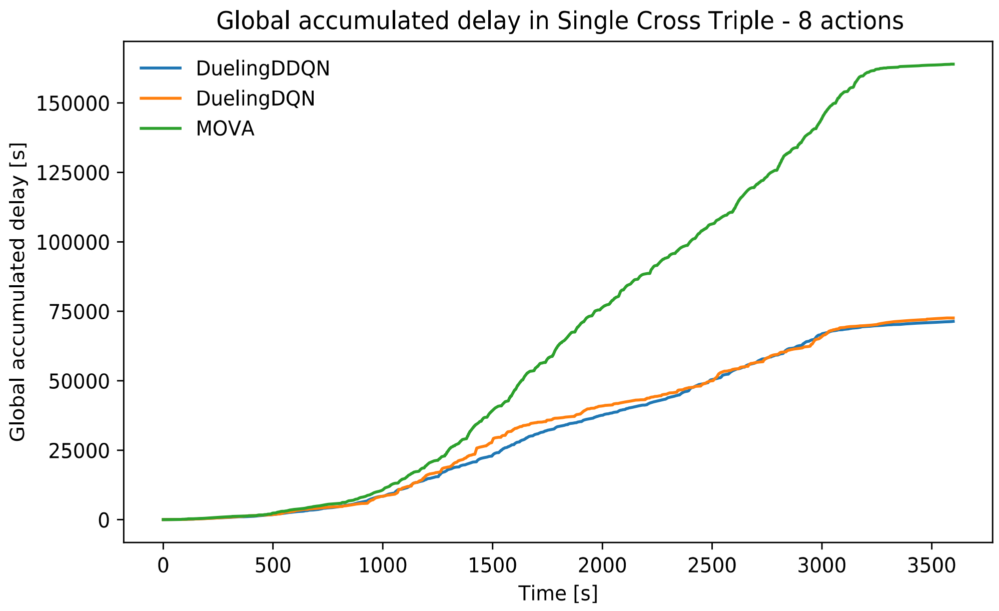


Figure 12 - Global Cumulative Delay in Cross Triple - 8 actions.

When we look at how the performance of the different systems scales as the demand increases, we can observe in Fig. 13 that in the complex intersection where the RL agents have the ability to freely switch between stages, they obtain better performance than MOVA across the board. In the scenarios covering low demand levels, where in simple intersections MOVA was obtaining marginally better performance, now the situation is reversed, mostly due to the previously mentioned ability to switch to the most suitable stages in absence of a predefined order, which was not the case in the intersection with 2 stages.

As the demand increases, the performance gap becomes wider, reaching a very significant advantage for the RL agent at high demand levels. This is consistent with the results presented in Fig.12 and Table 5.

Controller	Cumulative Delay [s]	Average Sum of Queues [m]
MOVA	165456.44	339.41
DDQN	72642.59	123.52
DDDQN	71245.61	119.86

Table 5 Cumulative Delay and Cumulative Stop Delay in seconds across Controllers on Single Cross Triple - 8 actions.

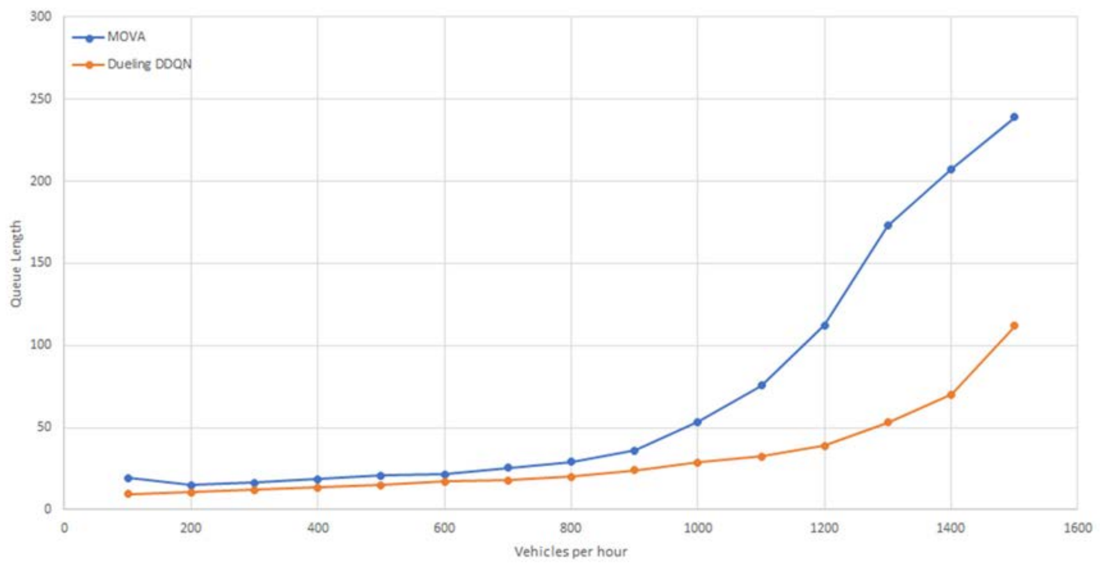


Figure 13 - Queue length by demand level in Cross Triple - 8 actions.

5. DISCUSSION AND FUTURE WORK

Several neural network architectures for RL controllers were tested. The agents did not require extensive or complex configurations to adequately control traffic junctions, outperforming the commercial controllers in terms of average queue length and delay, calculated as deviation from free-flow times. RL agents showed great stability and robustness to control situations within their training envelope as well as outside of it. Additionally, agents trained on relatively low uniform demand showed they can perform better than commercial systems during evaluation tests that included variable demand 5 times higher than anything experienced during training.

Experiment 1 provided evidence that fixed time systems perform worse than adaptive and RL UTCs in simple intersections. In this case, MOVA and the RL agent following a DuelingDDQN architecture obtained very similar results, with a slight advantage for the RL agent.

Experiment 2 provided similar evidence about a smaller number of controllers, in a situation where queues were measured on a per-link basis rather than per-lane. This implies less granularity in the data and makes the control task more challenging. The results followed the same pattern with a DuelingDQN agent obtaining the best performance, despite the lower quality of the input data.

Experiment 3 required modifications of the map in order to obtain said per-lane queues. This model saw the introduction of a much more complex intersection, with a multitude of actions available to the agent, some of them serving the same lanes in different ways, and letting the agent decide on the sequence of actions. Once again RL agents obtained better results than MOVA, with the DuelingDDQN agent obtaining the lowest global delay and average queue length. The gap between the performance of MOVA and RL agents is increased here with respect to the last experiment. Most likely reasons are higher granularity in the data and extra actions being available to the agent, allowing it to display more complex sequences of actions.

Reinforcement Learning applied to UTC keeps demonstrating that it can be a real-life solution to improve traffic conditions in urban environments, even though the sensors required are more sophisticated than simple induction loops. These experiments provide further evidence that Reinforcement Learning based UTCs could be the next generation solution for reducing traffic congestion.

ACKNOWLEDGEMENTS

This work was part funded by EPSRC Grant EP/L015374 and part funded by The Alan Turing Institute and the Toyota Mobility Foundation. The authors thank Dr. W. Chernicoff for the initial discussions and drive that made this project possible.

REFERENCES

BELL, M C, AND R D BRETHERTON. 1986. “Ageing of fixed-time traffic signal plans.” In International Conference on Road Traffic Control.

CABREJAS EGEA, ALVARO, PETER DE FORD, AND COLM CONNAUGHTON. 2018. “Estimating Baseline Travel Times for the UK Strategic Road Network.” In IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2018-Novem:531–36. IEEE. https://doi.org/10.1109/ITSC.2018.8569924.

CABREJAS EGEA, ALVARO, SHAUN HOWELL, MAKSIS KNUTINS, AND COLM CONNAUGHTON. n.d. “Assessment of Reward Functions for Reinforcement Learning Traffic Signal Control under Real-World Limitations.” In IEEE International Conference on Systems, Man and Cybernetics, Proceedings, SMC. https://doi.org/10.1109/smc42975.2020.9283498.

CABREJAS-EGEA, ALVARO AND CONNAUGHTON COLM. 2020a. “Wavelet Augmented Regression Profiling (WARP): improved long-term estimation of travel time series with recurrent congestion.” In IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC. https://doi.org/10.1109/itsc45102.2020.9294318.

CABREJAS-EGEA, ALVARO, AND COLM CONNAUGHTON. 2020b. “Assessment of Reward Functions in Reinforcement Learning for Multi-Modal Urban Traffic Control under Real-World limitations.” arXiv Preprint arXiv:2010.08819. https://arxiv.org/abs/arXiv:2010.08819.

GAO, JUNTAO, YULONG SHEN, JIA LIU, MINORU ITO, AND NORIO SHIRATORI. 2017. “Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network.” arXiv Preprint arXiv:1705.02755.

GENDERS, WADE. 2018. “Deep reinforcement learning adaptive traffic signal control.” PhD thesis.

GENDERS, WADE, AND SAIEDEH RAZAVI. 2018. “Evaluating reinforcement learning state representations for adaptive traffic signal control.” Procedia Computer Science 130: 26–33.

HASSELT, HADO V. 2010. “Double Q-learning.” In Advances in Neural Information Processing Systems, 2613–21.

HESSEL, MATTEO, JOSEPH MODAYIL, HADO VAN HASSELT, TOM SCHAUL, GEORG OSTROVSKI, WILL DABNEY, DAN HORGAN, BILAL PIOT, MOHAMMAD AZAR, AND DAVID SILVER. 2018. “Rainbow: Combining improvements in deep reinforcement learning.” In Thirty-Second AAAI Conference on Artificial Intelligence.

HEYDECKER, BENJAMIN G. 2004. “Objectives, stimulus and feedback in signal control of road traffic.” Journal of Intelligent Transportation Systems 8 (2): 63–76.

HUNT, P B, D I ROBERTSON, R D BRETHERTON, AND M CR ROYLE. 1982. “The SCOOT on-line traffic signal optimisation technique.” Traffic Engineering & Control 23 (4).

INRIX. 2019. “Scorecard.” http://inrix.com/scorecard/.

LIANG, XIAOYUAN, XUNSHENG DU, GUILING WANG, AND ZHU HAN. 2018. “Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks” XX (Xx): 1–11. http://arxiv.org/abs/1803.11115.

LIU, YING, LEI LIU, AND WEI PENG CHEN. 2018. “Intelligent traffic light control using distributed multi-agent Q learning.” IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC 2018-March: 1–8. https://doi.org/10.1109/ITSC.2017.8317730.

MANNION, PATRICK, JIM DUGGAN, AND ENDA HOWLEY. 2016. “An experimental review of reinforcement learning algorithms for adaptive traffic signal control.” In Autonomic Road Transport Support Systems, 47–66. Springer.

MNIH, VOLODYMYR, KORAY KAVUKCUOGLU, DAVID SILVER, ALEX GRAVES, IOANNIS ANTONOGLOU, DAAN WIERSTRA, AND MARTIN RIEDMILLER. 2013. “Playing atari with deep reinforcement learning.” arXiv Preprint arXiv:1312.5602.

MOUSAVI, SEYED SAJAD, MICHAEL SCHUKAT, AND ENDA HOWLEY. 2017. “Traffic light control using deep policy-gradient and value-function-based reinforcement learning.” IET Intelligent Transport Systems 11 (7): 417–23.

ROBERTSON, D. I. 1969. “A traffic network study tool.” RRL Report 253.

SALTER, R J. 1996. “Optimum cycle times for an intersection.” In Highway Traffic Analysis and Design, 304–10. Springer.

SCHAUL, TOM, JOHN QUAN, IOANNIS ANTONOGLOU, AND DAVID SILVER. 2016. “Prioritized experience replay.” 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings. https://arxiv.org/abs/1511.05952.

SILVER, DAVID, JULIAN SCHRITTWIESER, KAREN SIMONYAN, IOANNIS ANTONOGLOU, AJA HUANG, ARTHUR GUEZ, THOMAS HUBERT, ET AL. 2017. “Mastering the game of go without human knowledge.” Nature 550 (7676): 354–59.

SMITH, STEPHEN F, AND GREGORY J BARLOW. 2013. “SURTRAC : Scalable Urban Traffic Control.” Transport Research Board.

STEVANOVIC, ALEKSANDAR, AND PETER T MARTIN. 2008. “Split-cycle offset optimization technique and coordinated actuated traffic control evaluated through microsimulation.” Transportation Research Record 2080 (1): 48–56.

WAN, CHIA-HAO, AND MING-CHORNG HWANG. 2018. “Value-based deep reinforcement learning for adaptive isolated intersection signal control.” IET Intelligent Transport Systems 12 (9): 1005–10.

XIE, XIAO FENG, STEPHEN F. SMITH, LIANG LU, AND GREGORY J. BARLOW. 2012. “Schedule-driven intersection control.” Transportation Research Part C: Emerging Technologies 24: 168–89. https://doi.org/10.1016/j.trc.2012.03.004.

VINCENT, R A, AND J R PEIRCE. 1988. ’MOVA’: Traffic Responsive, Self-optimising Signal Control for Isolated Intersections. Traffic Management Division, Traffic Group, Transport; Road Research.

WANG, ZIYU, TOM SCHAUL, MATTEO HESSEL, HADO VAN HASSELT, MARC LANCTOT, AND NANDO DE FREITAS. 2015. “Dueling Network Architectures for Deep Reinforcement Learning.” ICML, November.