Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 58 (2021) 638–645

Transportation Research Procedia

www.elsevier.com/locate/procedia

14th Conference on Transport Engineering: 6th – 8th July 2021

Reinforcement Learning for Traffic Signal Control: Comparison with Commercial Systems

Alvaro Cabrejas-Egea[a,b,*], Raymond Zhang[a,c], Neil Walton[a,d]

[a]*The Alan Turing Institute, 96 Euston Rd, London NW1 2DB, United Kingdom*
[b]*MathSys CDT, University of Warwick, Coventry CV4 7AL, United Kingdom*
[c]*Ecole Normale Superieure de Paris-Saclay, 4 Avenue des Sciences, 91190 Gif-sur-Yvette, France*
[d]*University of Manchester, Oxford Rd, Manchester M13 9PL, United Kingdom*

Abstract

Intelligent Transportation Systems are leveraging the power of increased sensory coverage and 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 TSC RL agents and compare them against the currently used commercial systems MOVA, SurTrac and Cyclic controllers and provide pseudo-code for them. The agents use variations of Deep Q-Learning and Actor Critic, using states and rewards based on queue lengths. Their performance is compared in across different map scenarios with variable demand, assessing them in terms of the global delay and average queue length. We find that the RL-based systems can significantly and consistently achieve lower delays when compared with existing commercial systems.

Keywords: Modelling and Simulation; Urban Traffic Control; Reinforcement Learning

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

* Corresponding author. Tel.: +44-7511-398-261
E-mail address: alvaro.cabrejas@gmail.com

10.1016/j.trpro.2021.11.084

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, with systems that optimise the green time splits in a deterministic manner, requiring costly site-specific knowledge and typical demand profiles to provide effective control. These methods are not easily scalable and deteriorate over time as the traffic demand changes (Bell and Bretherton (1986)). With the development of induction loops, real time actuated UTCs were created in two variants: those that optimise 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 are based on local rules, generating a self-organising area traffic controllers, such as SurTrac (Smith et al. (2013)). With the breakthrough of Deep Reinforcement Learning (DRL) on complex problems such as Atari games or Go (Mnih et al. (2013); Silver et al. (2017)), attention has turned towards adapting these approaches to generate industry-grade controllers for traditionally noisy and systems such as TSC. This paper aims 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

RL is an area of ML aiming to imitate how biological entities learn, where an agent evolves in an unknown environment, learning how to perform with no prior information, based on its interactions said environment. The agent aims to maximise a reward signal it receives as feedback for its actions. RL methods have been applied to TSC in experimental setups. While there is a variety of approaches in the literature that craft successful RL-based TSC systems, most do no present comparisons against commercial systems that are the concern of this paper.

Recent works (Gao et al. (2017); Wan and Hwang (2018); Mousavi et al. (2017)) use neural networks as function approximators to avoid the dimensionality and computing limitations of table-based methods in large state-action spaces, showing DRL TSC can be more efficient than some earlier methods. The first two use discreet cell encoding vectors to represent the system, which are passed to a Convolutional Neural Network (CNN), whereas the second directly uses pixels in the same manner. Gao et al. (2017) compared the results against a fixed time and longest-queue-first systems, finding RL to perform better, while Wan and Hwang (2018) found similar results comparing against a fixed time system. Liang et al (2018) used the same initial approach and compared against two different fixed-time systems, ranking better than both and providing some early evidence of the benefits of using Double DQN (Hasselt (2010)), Duelling architecture (Want et al. (2016)) and Prioritised Experience Replay (PER) (Schaul et al (2016)). Genders and Razavi (2018) evaluated different state representations, finding little difference in the performance of the agents as a result of the change in the magnitudes observed. Stevanovic and Martin (2008) compared 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 Simulators and Optimisers

PTV Vissim. Our simulations are conducted on PTV Vissim, a state-of-the-art commercial traffic simulator that can produce a wide variety of traffic demands over an array of signal controls and road traffic scenarios. We interface our RL algorithms via COM interface, using Tensorflow to construct deep learning agents.

MOVA (Microprocessor Optimised Vehicle Actuation, Vincent and Peirce (1988)) is a traffic controller designed by TRL Software that 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, then computing a performance index based on the delays. 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 et al (2013)) is decentralised, with each intersection allocating 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 allows for locally balancing competing flows while creating "green corridors" by finding an optimal sequence such that the input jobs (ordered clusters) are cleared while minimising the joint waiting time.

3. Methods

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 $s \in S$, a set of actions of the agent $a \in A$, a stochastic transition function $\mathcal{T}_{s,s'}^{a} \triangleq P(s_{t+1} = s' | s_t = s, a_t = a)$, 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, and a discount factor γ that will provide the balance between immediate exploitation and approaches that aim to maximise returns over time. In the case of TSC, the MDP is modelled as partially observable, following an unknown stochastic transition function.

The goal of the agents is to maximise their future discounted return $G_t = \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t)$ with $\gamma \in [0,1]$ 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: $\pi: S \rightarrow A$. The reward function maps an action given a state to a reward scalar value: $r: S \times A \rightarrow R$. The action-value function or Q-value is $Q_\pi(s,a) = E_\pi[G_t | s_t = s, a_t = a]$ representing the total episodic return by following π after being in state s and taking action a.

Value Based RL Methods: Tabular value-based methods, such as Q-Learning, attempt to learn an optimal policy $Q_\pi^* = \max_\pi E\,[r_t | s_t = s, a_t = a]$ by iteratively performing Bellman updates on the Q-values of the individual state-action pairs: $Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\big(y_t - Q_\pi(s_{t+1}, a_{t+1})\big)$, with $y_t = r_t + \gamma \max_{a'} Q_\pi\,(s_{t+1}, a', \theta')$, where α is the learning rate and y_t is the Temporal Difference (TD) target for the value function.

Deep Q-Network (DQN) agents are an evolution of Q-Learning. The purpose of the agent is to find an approximation of $Q_\pi *$ 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 increases training stability, obtaining samples that cover a wider range of situations and that can be used several times for gradient descent. Three additional modules have been applied to the agent to improve performance, Double Q Learning, PER, and Dueling Architecture.

The DQN variants implemented are described on the algorithms displayed in Figs. 1a and 1b, and use the hyperparameters described in Fig. 2b.

Policy Gradient Reinforcement Learning Methods: 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 $J = \sum_{a \in A} E[Q(s_0, a)\pi(a|s)]$. The 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 $\pi_\theta = P_\theta[a_t = a | s_t = s]$, and a critic $V_\pi^\theta(s) = E_\theta[G_t | s_t = s]$ both of which are parametrised by the neural network weights vector θ. The pseudocode for the A2C agent can be found in Fig. 2a and its hyperparameters are displayed in Fig. 2c.

3.1. State, Actions and Reward of the Agents

All the experiments here presented use the same descriptions for state and reward calculation, differing 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 $s \in R^{l+\mathbb{1}}$, in which each component represents the length of the queue of vehicles measured upstream from the traffic light in metres. The last component will be current stage being implemented.

Algorithm 1 Operation of the implemented Dueling DQN Agent with PER

Initialise agent network with random parameters θ
Initialise target network with random parameters θ'
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 ϵ-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 θ over all $(x_i, y_i) \in b$
  if *number of episode is multiple of* N **then**
    $\theta' \leftarrow \theta$
  end
end

Algorithm 2 Operation of the implemented Dueling Double DQN Agent with PER

Initialise agent network with random parameters θ
Initialise target network with random parameters θ'
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 ϵ-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 θ over all $(x_i, y_i) \in b$
  if *number of episode is multiple of* N **then**
    $\theta' \leftarrow \theta$
  end
end

Fig. 1. (a) DDQN pseudocode; (b) DDDQN pseudocode.

Algorithm 3 Operation of the implemented Advantage Actor Critic Agent

Initialise actor network with random parameters θ_a,
Initialise critic network with random parameters θ_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

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}

Fig. 2. (a) A2C pseudocode; (b) DDQN/DDDQN hyperparameters (c) A2C Hyperparameters.

While marginal improvements in performance can be obtained by using different variables for reward (Cabrejas-Egea et al. (2020); Cabrejas-Egea and Connaughton (2020)), 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: $r_t = -\sum_l q_{l,t}$. 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. 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* using the methodology introduced in Cabrejas-Egea et al (2018) and expanded in Cabrejas-Egea and Connaughton (2019). The profile will be split on 10 segments of length 6 minutes. Each of these will correspond with a level of demand. The demand levels are obtained by setting 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 will be specified in each experiment's section. Random seeds are updated after every simulation episode, training or testing. The quantitative metrics on which the system will be evaluated are the Global Cumulative Delay (deviations from free-flow time) and the Average Queue Length generated during the evaluation.

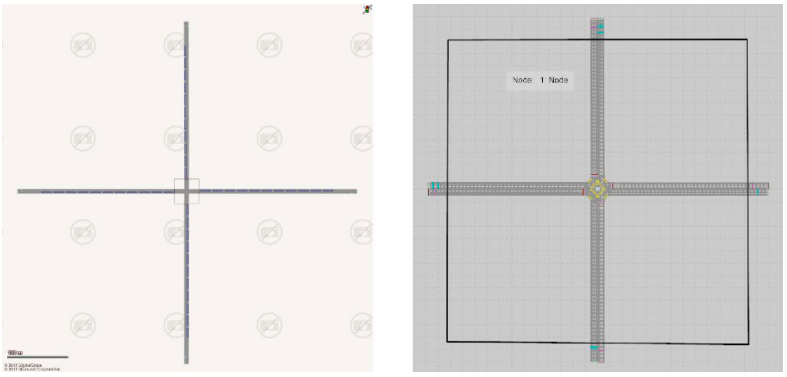

Fig. 3. (a) Cross Straight Map; (b) Cross Triple Map.

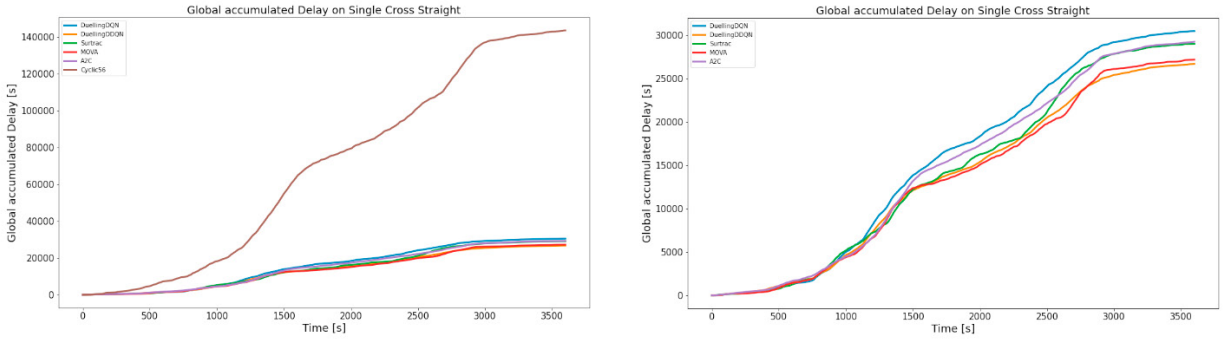

Fig. 4. Global Cumulative Delay in Single Cross for DQN variants, A2C, Surctrac, MOVA and the reference Cyclic controller..

4. Experimental Results

Experiment 1: Cross Straight. The first test is conducted on the simplest junction, shown in Fig. 3a. The junction is composed on 4 lanes distributed in 4 arms. The controller has two stages, a north-south stage and an east-west stage and turning is not allowed. The 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. MOVA was configured using loop detectors set in accordance with its manual, the implementation of Surtrac follows the work of Xie et al (2012). During evaluation, an average of 2120 vehicles are inserted in the model, with 2 peaks of demand of 3000 veh/h for 6 minutes each.

Figure 4 and Table 1 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. 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. 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.

Table 1. Cumulative Delay and Average Sum of Queues in Single Cross Straight.

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

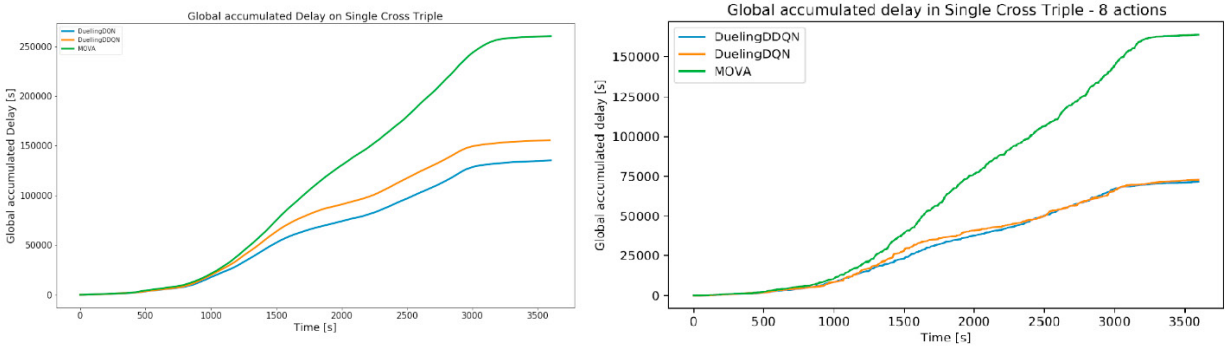

Fig. 5. (a) Global Cumulative Delay in Cross Triple - 4 actions. (b) Global Cumulative Delay in Cross Triple - 8 actions.

Experiment 2: Cross Triple - 4 actions. This junction, as shown in Fig. 3b 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. To mitigate this, the first experiment was run with agents that would take 4 queue inputs, 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. 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 it can be seen in Table 2 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. Further in Figure 5a we see that DDQN significantly outperforms MOVA over a wide range of loads and traffic scenarios. It can be seen that the agent using Dueling Q-Learning has a better performance than that Dueling Double Q-Learning.

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 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 lane changing limitations, direct comparisons with Experiment 2 must be handled with care. Both models share rough geometry, but the lanes layout is changed and so are the routing possibilities.

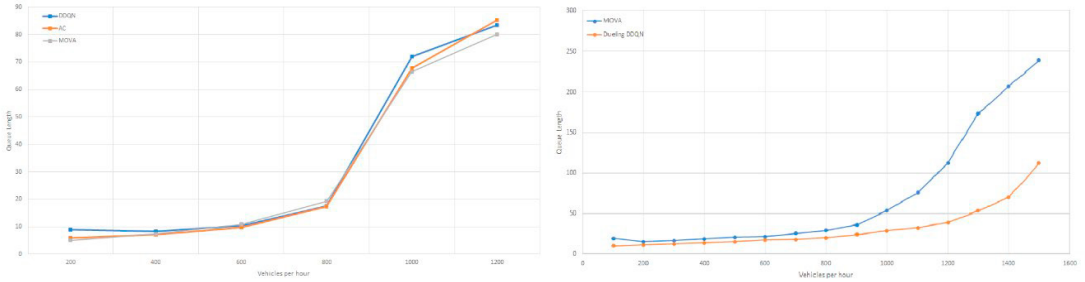

Fig. 6(a) Queue length by demand level in Cross Straight. (b) Queue length by demand level in Cross Triple.

Table 2. Cumulative Delay and Average maximum Queue length in Single Cross Triple – 4 actions.

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

The results presented below, use DQN agents taking 12 queue length inputs plus the state of the signal. There are 8 different stages. Here there are 4 additional stages available for non-conflicting cross traffic. No specific stage order is enforced, and the agents are free to change between any combination of stages.

Table 3. Cumulative Delay and Average maximum Queue length in Single Cross Triple – 8 actions.

Controller	Cumulative Delay [s]	Average Max Queue [m]
MOVA	165456.44	339.41
MOVA	72642.59	123.52
DDDQN	71245.61	119.86

The RL agents display a similar 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 have to 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 neural depth, width and activations weren't optimised, meaning that the RL agents can still be improved upon.

5. Discussion

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. 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 experienced during training.

In Experiment 1 MOVA and the RL agent following a DuelingDDQN architecture obtained very similar results, with a slight advantage for the RL agent. Experiment 2 implies less granularity in the data and makes the control

task more challenging. The results followed the same pattern with a DuelingDDQN agent obtaining the best performance over an array of loadings, despite the lower resolution in the input data. Experiment 3 introduced a much more complex intersection. Once again RL agents obtained significantly better results than MOVA, with the DuelingDDQN agent obtaining the lowest global and stop delay. 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.

We find that Reinforcement Learning applied to UTC can significantly outperform current adaptive traffic controllers in realistic commercial simulation software. These experiments provide credible evidence that Reinforcement Learning based UTC will be part of the next generation of traffic signal controllers.

Acknowledgements

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

References

Bell, M.C., Bretherton, R.D., 1986. Ageing of fixed-time traffic signal plans, in: International conference on road traffic control.

Cabrejas Egea, A., De Ford, P., and Connaughton, C. "Estimating Baseline Travel Times for the UK Strategic Road Network.". In IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC (pp. 531–536).2018.

Cabrejas-Egea, A. Connaughton C., 2020. Wavelet Augmented Regression Profiling (WARP): improved long-term estimation of travel time series with recurrent congestion, in: IEEE Conference on Intelligent Transportation Systems.

Cabrejas Egea, A., Howell, S., Knutins, M., Connaughton, C. (2020). 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.

Cabrejas-Egea, A., & Connaughton, C. (2020). Assessment of Reward Functions in Reinforcement Learning for Multi-Modal Urban Traffic Control under Real-World limitations. arXiv preprint arXiv:2010.08819.

Gao, J., Shen, Y., Liu, J., Ito, M., & Shiratori, N. (2017). Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. arXiv preprint arXiv:1705.02755.

Genders, W., & Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. Procedia computer science, 130, 26–33.

Hasselt, H. (2010). Double Q-learning. In Advances in Neural Information Processing Systems (pp. 2613–2621).

Heydecker, B. (2004). Objectives, stimulus and feedback in signal control of road traffic. Journal of Intelligent Transportation Systems, 8(2) 63.

Hunt, P. B., et al. "The SCOOT on-line traffic signal optimisation technique." Traffic Engineering & Control 23.4 (1982).

INRIX. (2019). Scorecard.

Liang, Xiaoyuan, et al. "Deep reinforcement learning for traffic light control in vehicular networks." arXiv preprint arXiv:1803.11115 (2018). Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. Autonomic road transport support systems, 47-66.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mousavi, S. S., Schukat, M., & Howley, E. (2017). Traffic light control using deep policy-gradient and value-function-based reinforcement learning. IET Intelligent Transport Systems, 11(7), 417-423.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.

Smith, S. F., Barlow, G., Xie, X. F., & Rubinstein, Z. B. (2013). Surtrac: Scalable urban traffic control.

Stevanovic, A., & Martin, P. T. (2008). Split-cycle offset optimization technique and coordinated actuated traffic control evaluated through microsimulation. Transportation Research Record, 2080(1), 48-56.

Vincent, R. A., & Peirce, J. R. (1988). 'MOVA': Traffic Responsive, Self-optimising Signal Control for Isolated Intersections (No. 70).

Wan, C. H., & Hwang, M. C. (2018). Value-based deep reinforcement learning for adaptive isolated intersection signal control. IET Intelligent Transport Systems, 12(9), 1005-1010.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In International conference on machine learning (pp. 1995-2003). PMLR.

Xie, X. F., Smith, S. F., Lu, L., & Barlow, G. J. (2012). Schedule-driven intersection control. Transportation Research Part C: Emerging Technologies, 24, 168-189.