Assessment of Reward Functions for Reinforcement Learning Traffic Signal Control under Real-World Limitations

Alvaro Cabrejas Egea*†, Shaun Howell†, Maksis Knutins† and Colm Connaughton§
*MathSys Centre for Doctoral Training
University of Warwick, Coventry CV4 7AL, United Kingdom
a.cabrejas-egea@warwick.ac.uk
†Vivacity Labs
London NW5 3AQ, United Kingdom
shaun@vivacitylabs.com, maksis@vivacitylabs.com
‡Warwick Mathematics Institute
University of Warwick, Coventry CV4 7AL, United Kingdom
c.p.connaughton@warwick.ac.uk

***Abstract*—Adaptive traffic signal control is one key avenue for mitigating the growing consequences of traffic congestion. Incumbent solutions such as SCOOT and SCATS require regular and time-consuming calibration, can't optimise well for multiple road use modalities, and require the manual curation of many implementation plans.**

Vision-based traffic sensors, such as those from Vivacity Labs, as well as advances in GPU technology, have paved the way for real-time machine learning approaches in this field, with deep reinforcement learning showing promising results. This uses deep neural networks to choose the best action in any given state, but is sensitive to the choice of reward function. Several authors have surveyed the reward functions used in the literature, but attributing outcome differences to reward function choice across works is problematic as there are many uncontrolled differences, as well as different outcome metrics.

This paper compares reward functions in a simulation of a junction in Greater Manchester, UK, across various demand profiles, subject to real world constraints: realistic sensor inputs, controllers, calibrated demand, intergreen times and stage sequencing. The reward metrics considered are based on the time spent stopped, lost time, change in lost time, average speed, queue length, junction throughput and variations of these magnitudes. The performance of these reward functions is compared in terms of total waiting time. We find that speed maximisation resulted in the lowest average waiting times across all demand levels, displaying significantly better performance than other rewards previously introduced in the literature.

***Index Terms*—Reinforcement Learning, Urban Traffic Control, Smart Cities, Agent-Based Modeling**

I. Introduction

Improving the operation of urban road networks can significantly reduce traffic congestion, as well as promote active travel and public transport within a city and reduce emissions. Optimising the timings of adaptive traffic signals can play a large role in achieving this by making effective use of green lights in response to the demand levels on each approach to a junction and by promoting progression across multiple junctions. In the UK, the main incumbent algorithms are 'Microprocessor Optimised Vehicle Actuation' (MOVA) [1] for isolated junctions, and 'Split, Cycle and Offset Optimisation technique' (SCOOT) [2] for regions of up to 30 signalised junctions. However, these algorithms have been improved only incrementally since their initial development in the 1980s, and so do not take advantage of either modern traffic data sources, nor modern computational resources. Modern traffic data sources, such as vision-based above-ground detectors, are able to provide much richer information about the road network, for example by providing live occupancy, flow, class of vehicle (e.g. car, van, bus) and individual vehicles' positions and speeds.

Reinforcement learning (RL) has been investigated as a potential next step in urban traffic control (UTC) systems, demonstrating the potential to outperform even well-calibrated systems currently in use [3]. However, most of the work to date is not intended to be directly applied to the real world. As such, all observed works in the literature overlook operational limitations of this application of RL. Further, there is a gap in the literature regarding the choice of reward function for such an RL system, which is a critical aspect. This paper provides a robust comparison of reward functions for RL, in the context of a junction in Greater Manchester, UK, a simulation of which has been calibrated using extensive data from Vivacity Labs vision-based sensors. This research is directly translatable to real-world applications of the technology, and has since been deployed to manage real traffic.

The paper is structured as follows: Section II reviews earlier work in the field and enumerates different reward functions used in the literature. Section III states the mathematical problem and the Reinforcement Learning theoretical background.

*This work was part funded by InnovateUK grant 104219 and part funded by EPSRC Grant EP/L015374. We are also grateful to the Alan Turing Institute for support in the initial stages.

Section IV describes the implementation and characteristics of the agents and environment. Section V describes and provides the analytic expressions of the different reward functions being tested. Section VI gives detail on the training, selection and evaluation of the agents. Section VII shows the results of the experiments in terms of average waiting time of the vehicles.

II. Related Work

Previous studies have considered RL for UTC without focusing specifically on the choice of reward functions. Initial research was centered around proof-of-concept, with studies such as [4] and [5] advocating for its potential use, and the ability of Q-Learning to perform better [6] [7] than traditional UTC methods such as MOVA [1], SCOOT [2] and SCATS [8]. Later research looked into neural networks as a function approximator to estimate the value of state-action pairs whilst addressing discretisation issues raised previously [9] [10]. Recent research makes use of deep RL to estimate the state-action values for each state [11] [12] [13] [14] [15] [16] [17] or to learn a policy directly that maps states to actions [9] [18] [19] [20] [21].

A number of publications have compiled RL methods. In [22], early table based methods are summarised; in [11] reward functions in a multi-junction network (delay between actions, difference in delay between actions, minimisation and balancing of queues, and minimising stops) are compared, and in [19] three different state representations are compared, finding similar performance with each. As the outcome was found not to be sensitive to state representation, the present work keeps the state representation constant. More recent studies focus on different RL approaches to this problem [23], and [24] considers state representation, reward function, action definition and model specific distinctions (online-offline, policy-value, tabular-function approximation) in a survey across the field, but without performing comparisons. Previous work typically approximates in terms of road geometry, traffic demand, and operational constraints, creating models of intersections that preclude real-world applicability.

This points to a gap in the literature of directly comparing a broad range of reward functions, in a well-calibrated, geometrically-accurate simulation which accounts for real-world limitations (e.g. safety constraints): this is the topic of this paper.

III. Problem Background and Definition

A. Reinforcement Learning

An intelligent agent is an entity which acts towards a goal based on observations and a decision making process. RL is an area of Machine Learning which focuses on how agents can learn a policy π based on extended interaction with an environment. This approach treats the problem as a Markov Decision Process (MDP), defined in terms of the $< \mathrm{S}, \mathrm{A}, \mathrm{T}, \mathrm{R}, \gamma >$ tuple, defined in accordance to what is shown in Fig. 1:

- S: Set of possible observable states. At time t, the agent will be provided with an observation $O_t = \{s_t^1, ..., s_t^i, ..., s_t^n\}$ of the state variables $s_t^i \in \mathrm{S}$.
- A: Set of possible actions that the agent can execute in the environment. After being provided with $\mathrm{s_t}$, the agent chooses an action $a_t \in \mathrm{A}$.
- T: Probabilistic state transition function.
- R: Set of rewards that the agent will receive as a function of its performance. The reward $r_t(s, a)$ represents the reward at time t, obtained from performing action a while being in state s.
- $\gamma \in [0, 1]$: Discount factor for the reward of the next step, balancing the trade-off between future and immediate rewards.

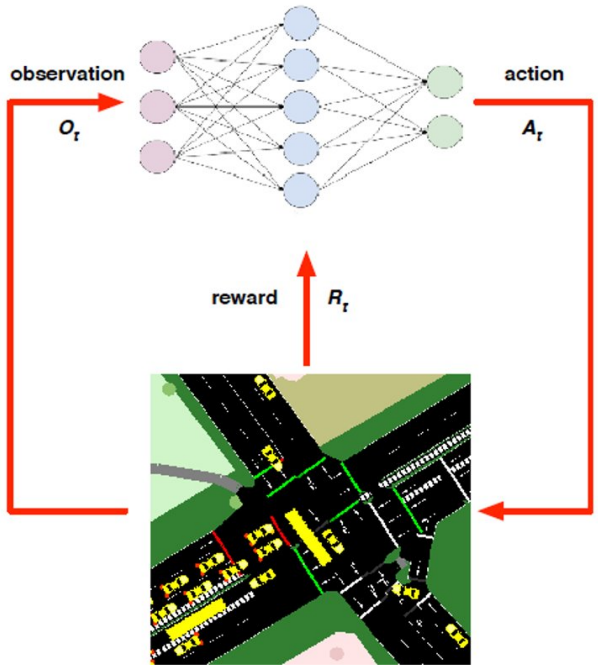

Figure 1. Schematic representation of information flows between Environment and Agent in a Reinforcement Learning framework.

The goal of the agent is to learn an optimal policy π^* that maximises the expected future reward. The discounted future reward at time t, R_t, is defined in equation (1).

$$R_t = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right] \tag{1}$$

B. Q-Learning and Deep Q-Learning

Q-Learning [25] defines the value of a state action pair as the Q-Value $Q(s, a)$, which represents the value of taking a certain action a while in state s, resulting in a transition to a new state s'. $Q(s, a)$ is approximated by successive Bellman updates:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha[r + \gamma \max_{a'} Q_t(s', a')] \tag{2}$$

If $Q(s, a)$ is known, eq. (2) can be solved to obtain π^* however its value is usually unknown, and so, estimating it is the task of Q-learning. Deep Q-Learning [26], thereby, uses a deep neural network as this function estimator.

IV. Methods

A. Agent

The agent uses a Deep Q Network (DQN) implemented in PyTorch [27] with 2 fully connected hidden layers of 500 and 1000 neurons respectively, and an output layer of 2 neurons, one per allowed action. All layers use ReLU as an activation function. The network weights are optimised using Stochastic Gradient Descent [28], using ADAptive Moment Estimation (ADAM) [29] as the optimizer with a learning rate of $\alpha = 10^{-5}$. The discount factor was set to $\gamma = 0.8$ for all experiments.

Algorithm 1: Schematic Learning Process

```
Initialise agent network with random parameters θ;
Initialise memory M with capacity L;
for each episode do
    measure initial state s_0;
    while episode not done do
        choose action a_t according to ε-greedy policy;
        implement action a_t;
        advance simulator until next action needed;
        measure new state s_{t+1}, and calculate reward
          r_{t+1};
        store transition tuple (s_t, a_t, r_{t+1}, s_{t+1}) in M;
        s ← s_{t+1};
    end
    b ← sample batch of transitions tuples from M;
    for each transition x_i = (s_i, a_i, r_{i+1}, s_{i+1}) in b do
        y_i = r_{i+1} + γ max_a Q(s_{i+1}, a', θ')
    end
    perform a step of Stochastic Gradient Descent on θ
      over all (x_i, y_i) ∈ b;
end
```

B. Environment

Our environment is a real four-arm junction located in Greater Manchester, UK. This junction is modelled using the microscopic traffic simulator SUMO [30] and calibrated using 3.5 months of flow and journey time data collected by vision-based sensors. A stage is defined as a group of non-conflicting green lights (phases) in a junction. The Agent decides which stage to select next and requests this from an emulated traffic signal controller, which moves to that stage subject to its limitations. These limitations are primarily safety-related and examples include enforcing minimum green times, minimum intergreen times, and stage transitions that match reality. The site features four Vivacity vision-based sensors which can provide flow, queue length and speed data. The data available to the agent is restricted to what can be obtained from these sensors.

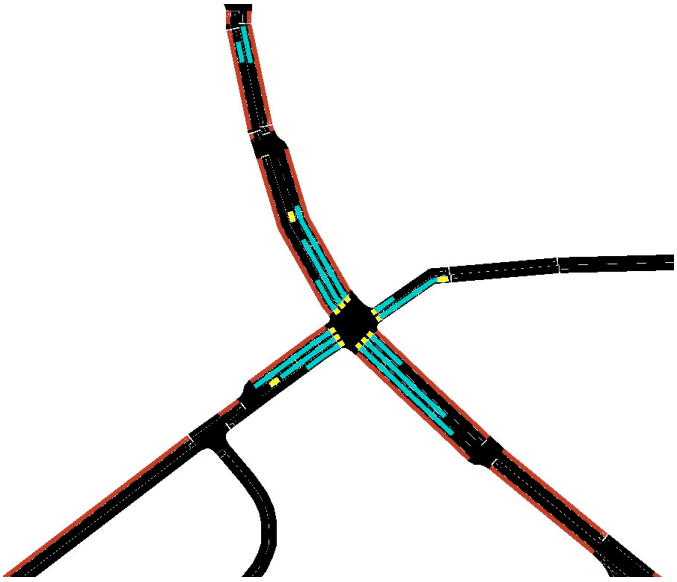

Figure 2. Intersection model in SUMO.

This paper does not consider pedestrians, thus promoting comparability with prior work; pedestrians will be considered for future work.

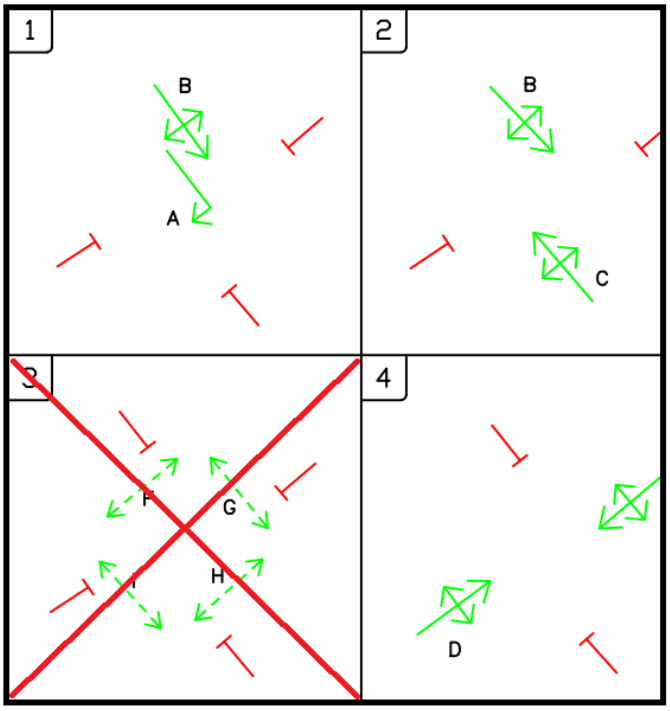

Figure 3. Allowed stages in the intersection (Stage 3 only serves pedestrians so isn't used). Stage 1 is an intermediate stage required to reach stage 2.

C. State Representation

The input to the agent is a combination of two parts: sensor data and traffic controller state. The sensor data is the occupancy of each lane area detector, while the controller state is a one-hot representation of which stage is active. A 12-second buffer at 0.6s resolution of both parts is provided to the agent.

While there are other state representations provided in the literature which are more information dense, many features of these cannot practically be obtained in the real world by the available sensors. Moreover, recent findings [19] indicate that the gain from more information-dense states is marginal, meaning an Agent can manage an isolated intersection with relatively simple state inputs. The state representation has been kept constant across the different experiments presented in this paper.

D. Action Set

The junction is configured to have 4 available stages. The Agent is able to choose Stage 2 or Stage 4, yielding an action space size of 2. Stage 1 serves a leading right turn phase from the main road, and was excluded by suggestion of the transport authority, since it is an intermediary stage that the controller will go through in order to reach Stage 2, which serves the main road. Stage 3 only serves pedestrians, which are not considered here, so was also excluded. Stage 4 serves the side roads, which do experience significant demand. In each timestep when a stage has been active longer than the minimum green time, the agent generates state-action values for each potential stage and the highest value is chosen according to an ϵ-greedy policy [31]. If the agent chooses the same stage, that stage is extended by 0.6s, otherwise the controller begins the transition to the other stage. The extension can be chosen indefinitely, as long as the agent identifies it as the best action.

The complexity in the decision-making stems from the combination of using Stage 1 as an intermediate state and the extensions to the stage duration. Traditional RL for UTC regards each Stage as an action for the agent to take, based on the instantaneous state of the system. However, in the case of the intermediate Stage 1, the agent has to choose when to start the transition without knowledge of the future state when Stage 2 begins. Regarding the extensions, given that their length is smaller than that of the initial phase, their impact on the state will be smaller, generating a distribution of reward and state-action value outcomes that the agent needs to approximate.

V. Reward Functions

In this section the individually tested reward functions are introduced.

Let N be the set of lane queue sensors present in the intersection. Let V_t be the set of vehicles on incoming lanes in the intersection at time t, let s be their individual speeds, τ their waiting times, and ρ the flow across the intersection over the length of the action. Let t^p be the time at which the previous action was taken and t^{pp} the time of the action before that.

A. Queue Length based Rewards

1) Queue Length: The reward will be the negative sum over all n sensors of the queues (q) at time step t. Similar to the reward introduced in [7] but without the need for thresholding the queue values, and used in [21].

$$r_t = -\sum_{n \in N} q_t \tag{3}$$

One of the first published in the field of Q-Learning, this reward function has low sensor requirements and is implementable using just induction loops.

2) Queue Squared: Introduced in [20], this function squares the result of adding all queues. This increasingly penalises actions that lead to longer queues.

$$r_t = -\left(\sum_{n \in N} q_t\right)^2 \tag{4}$$

3) Delta Queue: The reward will be the difference between the previous and current sum of queues, turning positive for those action that decrease the queue size and negative when it increases. Similar approach to the rewards shown in Eqs. (7) and (10).

$$r_t = \sum_{n \in N} q_{t^p} - \sum_{n \in N} q_t \tag{5}$$

B. Waiting Time based Rewards

1) Wait Time: The reward will be the negative aggregated time in queue (τ) that the vehicles at the intersection have accumulated since the last action.

$$r_t = -\sum_{v \in V_t} \tau_t \tag{6}$$

This function is more information-dense than queues, scaling with individual waiting times, but requires more advanced hardware for individual vehicle recognition.

2) Delta Wait Time: Similar to Eq. 5, used in [16]. The reward will be the difference between total intersection τ between the current time and the previous action.

$$r_t = \sum_{v \in V_t} \tau_{t_p} - \sum_{v \in V_t} \tau_t \tag{7}$$

3) Waiting Time Adjusted by Demand: The reward will be the negative aggregated waiting time as above, but in this case it is divided by an estimate of the current demand ($\hat{d}$) or arrival rate, implicitly accepting that given a wait time as a result of an action, the penalty should scale with the difficulty of the task.

$$r_t = -\frac{1}{\hat{d}} \sum_{v \in V_t} \tau_t \tag{8}$$

C. Time Lost based Rewards

1) Time Lost: Used in [15], the reward will be the negative aggregated delay accumulated by all vehicles upstream from the intersection, understanding the delay as deviations from the vehicle's maximum allowed speed (s_{max}). Assuming a simulator time step of length ts:

$$r_t = -\sum_{v \in V_t} \sum_{t^p}^{t} ts * \big(1 - \frac{s_v}{s_{max}}\big) \tag{9}$$

This reward provides a more accurate representation of the total delay caused, since it also accounts for all deceleration happening around the intersection.

2) Delta Time Lost: Introduced in [6] and used in [11] [13] [14] [18] and [19]. Similar to Eq. (7). The reward will be the change of global delay in the vehicles around the intersection since the last action was taken.

$$r_t = \sum_{v \in V_t} \sum_{t^{pp}}^{t^p} ts * \big(1 - \frac{s_v}{s_{max}}\big) - \sum_{v \in V_t} \sum_{t^p}^{t} ts * \big(1 - \frac{s_v}{s_{max}}\big) \tag{10}$$

This reward function provides both punishment and reward centered around zero.

3) Delay Adjusted by Demand: The reward will be the same as in the point above, but divided by an estimate of the demand level.

$$r_t = -\frac{1}{\hat{d}} \sum_{v \in V_t} \sum_{t^p}^{t} ts * \big(1 - \frac{s_v}{s_{max}}\big) \tag{11}$$

D. Average Speed based Rewards

1) Average Speed: This reward seeks to maximise the average joint speed of all vehicles around an area of influence around the intersection.

$$r_t = \frac{1}{\sum_{V_t} v} \sum_{v \in V_t} s \tag{12}$$

2) Average Speed Scaled by Demand: The reward will be, as in the previous section, multiplied by an estimation of the demand. This function scales the reward with the difficulty of the task.

$$r_t = \frac{\hat{d}}{\sum_{V_t} v} \sum_{v \in V_t} s \tag{13}$$

E. Throughput based Rewards

1) Throughput: The reward will be the total number of vehicles that cleared the intersection between the last time that an action was taken and now.

$$r_t = \sum_{t_p}^{t} \rho \tag{14}$$

2) Other reward functions: There are several other reward functions which were not considered. For example minimising the frequency of signal change [12] [33], and accident avoidance [12]: both of these concerns are already addressed by traffic signal controllers. Also, pressure has been used as a reward function in the control of a large network [23] [24] [32] [33] [3] [34] and achieved good results; however, this requires data from upstream and downstream of the target junction, so is beyond the current scope.

VI. Experimental Setup

A. Training Process

In each training run, the agent is subject to a training curriculum including a variety of scenarios, including sub-saturated, near-saturated, and over-saturated situations. The agent is first shown sub-saturated episodes, before the difficulty level is increased. Each episode runs for 3000 steps of length 0.6 seconds, for a total simulated time of 30 minutes (1800 seconds).

Ten training runs were conducted for each of the reward functions described in the following section, to capture variance in training run outcome.

B. Evaluation and Scoring

After each training run is complete, the agent's performance is evaluated and compared to that of reference agents. Two reference agents have been implemented to give context to the RL performance: Maximum Occupancy (longest queue first) and a Vehicle Actuated Controller. The Vehicle Actuated algorithm was an implementation of System D [35], a common algorithm in the UK. For each training run, the RL and reference agents are each tested on 300 scenarios: this is 100 scenarios at each of 3 demand levels. In each repetition the average waiting (stopped) time for all vehicles was computed.

The first scenario involves a demand of 1714 vehicles/hour (1 vehicle/2.1 seconds), and will be referred as Low Demand scenario. The second scenario uses a demand of 2117 vehicles/hour (1 vehicle/1.7 seconds), and will be referred as Medium Demand scenario, although this level of demand is around the observed peak and the saturation level for the junction. The third scenario test a demand level slightly above what the junction can currently serve. It uses a demand of 2400 vehicles/hour (1 vehicle/1.5 seconds) and will be referred as High Demand scenario.

For each reward function, the best result of the 10 training runs was selected, based on the above evaluation method.

VII. Results

In this section, the distribution of average waiting times per vehicle across the different scenarios is presented for each reward function.

Figures 4 - 6 show for each agent, the distribution of average waiting times per vehicle across 100 repetitions of each scenario as described in the section above. The throughput reward function is not shown to improve graph readability, as

TABLE I

AVERAGE WAITING TIME IN SECONDS ACROSS DEMAND SCENARIOS

Reward Function	Low	Medium	High
Queues	6.59±0.46	8.97±1.27	16.21±5.07
Queues Squared	6.35±0.53	8.56±1.26	16.22±5.08
Delta Queues	6.47±0.41	8.96±1.59	18.87±5.07
Delta Queues PLN	7.13±0.42	10.22±2.02	22.55±5.18
Stopped Time	6.64±0.52	11.88±3.39	26.27±4.61
Stopped Time DN	6.70±0.46	9.60±1.66	17.68±4.95
Delta Stopped Time	7.15±0.81	16.59±4.95	29.21±3.67
Time Lost	6.79±0.42	9.23±1.15	15.84±4.36
Time Lost DN	6.59±0.46	8.97±1.27	16.21±5.07
Delta Time Lost	8.27±1.48	13.48±4.04	22.54±5.54
Average Speed	6.24±0.39	8.61±1.07	14.95±3.40
Average Speed DN	**5.96±0.40**	**8.08±1.07**	**13.68±2.98**
Throughput	28.02±9.36	51.16±7.23	55.72±7.02
Vehicle Actuated	8.70±0.62	14.76±1.69	27.9±6.05
Maximum Occupancy	6.32±0.51	9.51±1.87	21.33±5.77

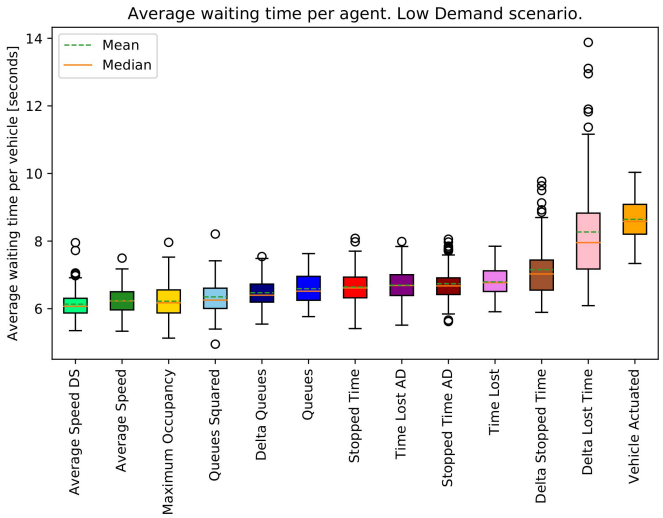

Figure 4. Distribution and medians of average waiting time in seconds across agents in Low Demand scenario. Sub-saturation demand of 1714 vehicles/hour (1 vehicle/2.1 seconds).

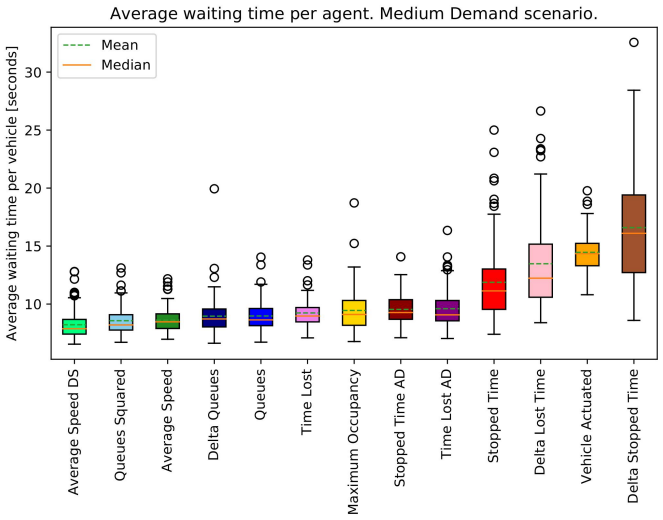

Figure 5. Distribution and medians of average waiting time in seconds across agents in Medium Demand scenario. Near-saturation demand of 2117 vehicles/hour (1 vehicle/1.7 seconds).

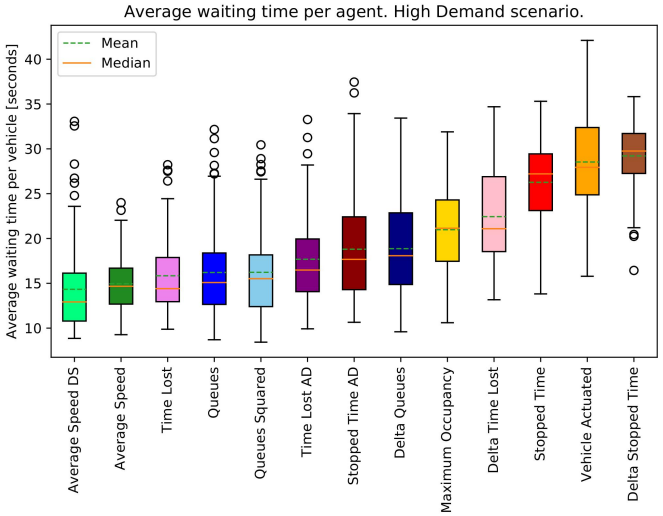

Figure 6. Distribution and medians of average waiting time in seconds across agents in High Demand scenario. Over-saturation demand of 2400 vehicles/hour (1 vehicle/1.5 seconds).

it performed poorly by all metrics. Complete corresponding results are reported in Table I.

Reward functions optimising the Average Speed of around the intersection were found to be best performing across all three scenarios and consistently outperformed the reference agents. Adjusting the average speed by an estimate of the demand level shows to improve the performance of this reward function in all cases.

From those rewards using queues, Queues Squared performed best, except for the High Demand scenario in which the sum of Queues obtains marginally lower waiting times.

The results from agents using reward functions based on either Wait Time or Time Lost are inconsistent across different demand levels. While sum of Time Lost is the third best-performing agent at High Demand, it is sixth in Medium Demand and Tenth in Low Demand. Sum of Wait Time follows an inverse, yet less extreme trajectory as the demand is lowered, from worse to better performance.

Throughput was found to be consistently the worst-performing of the reward functions tested.

The rewards based on difference in delay (Delta Wait Time and Delta Time Lost), which were used in seven previous items of research as a most desirable reward function, were amongst the worst performers.

VIII. DISCUSSION AND CONCLUSION

This paper has explored the performance of several reward functions for deep Q-learning agents in the context of Urban Traffic Control. The paper reaffirms earlier findings that RL outperforms deterministic baseline algorithms. Overall, the

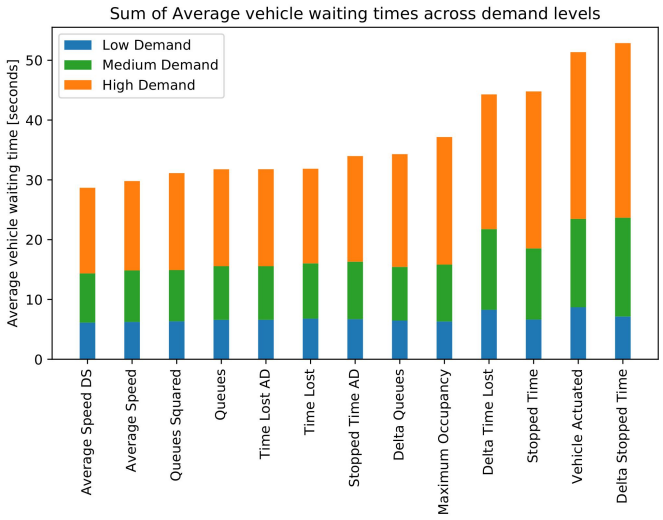

Figure 7. Stacked bar chart of Average Waiting Time across scenarios.

proposed reward functions based on maximisation of the average speed of the vehicles in the network resulted in the lowest waiting times across demand levels, including both baseline algorithms. The performance of queue length and stopped time rewards was comparable, but when reliability is taken into account, average speed is clearly preferred since it produces a smaller variation in waiting times.

In sub-saturated conditions, a simple approach of always serving the largest queue probably suffices, although rewarding based on queue length had similar performance, and rewarding for average speed yielded slight improvements. In busier conditions, which are more important to transport authorities, RL provides significant improvements in waiting time, with speed-based rewards again performing best, followed by queue-based rewards in near-saturation conditions, and followed by penalising for lost time in saturated conditions.

Rewarding based on delay minimisation or stopped time may perform worse than speed-based rewards as they encode information about previous timesteps, whereas queue lengths and vehicle speeds are snapshots at a single timestep, which better suits the underpinning Markov Decision Process framing of the problem. Speed maximisation may have yielded better results than queue minimisation under high demand as it penalises the agent for vehicles which are moving slowly forward in congested conditions, which would not have been penalised by the queue length reward. This would also not have been captured by the stopped time reward, which may explain its worse performance overall. One key finding is that whilst rewards based on difference in delay were found to perform well in earlier works, this was found to perform poorly in the present work. The Maximum Occupancy reference agent performs well at low demand, but mediocre at medium demand and poorly at high demand. It does not extend stage durations in busy traffic when the density of vehicles using the active stage is slightly lower than that of stationary queuing traffic. This results in shorter than optimal stage lengths and so too much time is wasted in interstage transitions. However, it did consistently perform better than the Vehicle Actuated reference agent. The Vehicle Actuated algorithm extends stages until the related loops have been unoccupied for 1.5s, hence extending stages in congested traffic. However, its performance suggests that this standard extension length, coupled with the maximum stage durations found empirically to perform relatively well, is overly eager to extend stages. From a visual inspection via the simulator of the best RL agent's behaviour, it appears to learn an 'adaptive stage extension' behaviour: learning when to prioritise a large queue waiting to be served and when to prioritise avoiding the cost of transitioning stages. A key benefit to the DQN approach is that it can learn how eagerly it should extend stages under varying conditions at a specific site, rather than requiring manual calibration. This automated calibration can be repeated regularly so that the site continues to perform well long-term without manual intervention.

One limitation of the study is that the results presented only relate to the training of a deep Q-learning agent as specified in Section IV, and are not necessarily representative of the performance in other type of architectures such as those representing the state as an image and processing it with Convolutional Neural Networks, or Policy Gradient methods. Another limitation is that most intersections found in an urban setting do not exclusively serve vehicles. As stated in Section IV, a similar exploration including pedestrians is left as future work for modal prioritisation based rewards.

ACKNOWLEDGMENT

A.C.E thanks N. Walton and W. Chernicoff for the insights and discussions that led to this paper, and the Toyota Mobility Foundation for previous support that made this work possible. Vivacity Labs thanks Transport for Greater Manchester for helping take this work from simulation to the real world, Immense for their work calibrating the simulation model, and InnovateUK for funding that made the work possible.

REFERENCES

[1] Vincent, R. A., & Peirce, J. R. (1988). 'MOVA': Traffic Responsive, Self-optimising Signal Control for Isolated Intersections. Traffic Management Division, Traffic Group, Transport and Road Research Laboratory.

[2] Hunt, P. B., Robertson, D. I., Bretherton, R. D., & Royle, M. C. (1982). The SCOOT on-line traffic signal optimisation technique. Traffic Engineering & Control, 23(4).

[3] Wei, H., Chen, C., Zheng, G., Wu, K., Gayah, V., Xu, K., & Li, Z. (2019, July). PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1290-1298).

[4] Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In Machine Learning: Proceedings of the 17th International Conference (ICML'2000) (pp. 1151-1158).

[5] Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering, 129(3), 278-285.

[6] El-Tantawy, S., & Abdulhai, B. (2010, September). An agent-based learning towards decentralized and coordinated traffic signal control. In 13th International IEEE Conference on Intelligent Transportation Systems (pp. 665-670). IEEE.

[7] Prashanth, L. A., & Bhatnagar, S. (2010). Reinforcement learning with function approximation for traffic signal control. IEEE Transactions on Intelligent Transportation Systems, 12(2), 412-421.

[8] Lowrie, P. R. (1990). Scats, Sydney Co-ordinated Adaptive Traffic System: A traffic responsive method of controlling urban traffic.

[9] Richter, S., Aberdeen, D., & Yu, J. (2007). Natural actor-critic for road traffic optimisation. In Advances in neural information processing systems (pp. 1169-1176).

[10] Araghi, S., Khosravi, A., Johnstone, M., & Creighton, D. (2013, October). Intelligent traffic light control of isolated intersections using machine learning methods. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (pp. 3621-3626). IEEE.

[11] Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In Autonomic road transport support systems (pp. 47-66). Birkhäuser, Cham.

[12] Van der Pol, E., /& Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016).

[13] Genders, W., & Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.

[14] 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.

[15] 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.

[16] Liang, X., Du, X., Wang, G., & Han, Z. (2018). Deep reinforcement learning for traffic light control in vehicular networks. arXiv preprint arXiv:1803.11115.

[17] Zhou, P., Braud, T., Alhilal, A., Hui, P., & Kangasharju, J. (2019, March). ERL: Edge based Reinforcement Learning for optimized urban Traffic light control. In 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 849-854). IEEE.

[18] 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.

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

[20] Genders, W. (2018). Deep reinforcement learning adaptive traffic signal control (Doctoral dissertation).

[21] Aslani, M., Mesgari, M. S., Seipel, S., & Wiering, M. (2019, October). Developing adaptive traffic signal control by actor–critic and direct exploration methods. In Proceedings of the Institution of Civil Engineers-Transport (Vol. 172, No. 5, pp. 289-298). Thomas Telford Ltd.

[22] El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. Journal of Intelligent Transportation Systems, 18(3), 227-245.

[23] Yau, K. L. A., Qadir, J., Khoo, H. L., Ling, M. H., & Komisarczuk, P. (2017). A survey on reinforcement learning models and algorithms for traffic signal control. ACM Computing Surveys (CSUR), 50(3), 1-38.

[24] Wei, H., Zheng, G., Gayah, V., & Li, Z. (2019). A Survey on Traffic Signal Control Methods. arXiv:1904.08117.

[25] Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[26] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[27] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Desmaison, A. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (pp. 8024-8035).

[28] Kiefer, J., & Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics, 23(3), 462-466.

[29] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[30] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y. P., Hilbrich, R., ... & WieBner, E. (2018, November). Microscopic traffic simulation using sumo. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2575-2582). IEEE.

[31] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[32] Varaiya, P. (2013). The max-pressure controller for arbitrary networks of signalized intersections. In Advances in Dynamic Network Modeling in Complex Transportation Systems (pp. 27-66). Springer, New York, NY.

[33] Wei, H., Zheng, G., Yao, H., ... (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control. Proceedings of 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2496-2505).

[34] Chacha Chen, H. W., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., ... Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control.

[35] Highways Agency (2002). Siting Of Inductive Loops For Vehicle Detecting Equipments At Permanent Road Traffic Signal Installations. MCE 0108 Issue C.