Assessment of Multi-Modal Reward Functions in Reinforcement Learning for Urban Traffic Control under Real-World limitations

Alvaro Cabrejas Egea
MathSys Centre for Doctoral Training,
University of Warwick & Vivacity Labs, London, UK
Email: a.cabrejas-egea@warwick.ac.uk

Colm Connaughton
Warwick Mathematics Institute
University of Warwick
Email: c.p.connaughton@warwick.ac.uk

***Abstract*— Traffic Signal Control is an important real-world challenge that has direct impact over the economic performance of urban areas. Recently, Reinforcement Learning is proving to be a successful tool that can effectively manage urban intersections with a fraction of the effort required to curate traditional Urban Traffic Controllers. However, literature dealing with the introduction and control of pedestrians in such intersections is scarce, and comparisons between the effect of the different approaches that can be taken in terms of architecture, state space or rewards are uncommon.**

This paper performs a robust comparison between different 31 different Reinforcement Learning reward functions for controlling intersections serving vehicles and pedestrians. We use a calibrated model in terms of demand, sensors, green times and other operational constrains of a real intersection in Greater Manchester, UK. The rewards can be broadly classified in 5 groups depending on the magnitudes used: queues, waiting time, delay, average speed and throughput in the junction.

The performance of different agents, in terms of waiting time, is compared across different demand levels ranging from normal operation to saturation of traditional adaptive controllers. We find that those rewards maximising the speed of the network obtain the lowest waiting time for vehicles and pedestrians simultaneously, closely followed by queue minimisation, showing better performance than other methods proposed in the literature.

I. Introduction

Effective traffic signal control is one of the key issues in Urban Traffic Control (UTC), effectively controlling how the available resources (green time) in our urban travel network are allocated. The efficiency associated with this allocation has an important impact in travel times, harmful emissions and economic activity.

First, fixed time controllers, and later adaptive systems have been used to further optimise the global flow through our cities. Recent improvements in CPU and especially GPU power are allowing for vision-based sensors to gather large amounts of real-time data that a few years ago seemed unattainable, such as individual vehicle position and speeds, all in a much cheaper way than with traditional actuated sensors. As a side effect of these developments, not only the area covered by them is ever increasing, but it is also becoming possible to direct some of these towards the pedestrians. This allows for the development of novel smart control approaches, that harness the power of real-time data to deliver cheap, responsive and flexible systems that can adapt to a variety of situations.

Reinforcement Learning (RL) approaches have been showing promising results in this field. However, most of the available works do not try to jointly optimise vehicular and pedestrian travel times, even when pedestrians are allowed and present in the great majority of urban intersections in any city.

This paper compares the performance of RL agents using 31 different reward functions split into 5 different classes by the magnitudes they use, when controlling a simulation of a real-world junction in Greater Manchester (UK) that has been calibrated using 3.5 months of data gathered from Vivacity Labs vision-based sensors.

The paper is structured as follows: Section II reviews previous literature in the field. Section III states the mathematical framework used and provides with some theoretical background. Section IV reviews the environment, the agents and their implementation. Section V introduces the reward functions tested in this paper and provides analytical expressions for them. Section VI contains details about the training and evaluation of the agents. Lastly, Section VII provides the experimental results and the discussion.

II. Related Work

RL for UTC has been previously explored and discussed in a variety of pieces of research, aiming to substitute existing adaptive control methods such as SCOOT [2], MOVA [1] and SCATS [3]. The field has evolved from early inquiries about its theoretical potential use [4] [5] [6] [7] [8], to more applied and realistic scenarios [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] that look towards real-world use and deployment. These use different methods, such as complex pixel-based inputs passed to a CNN or per-lane state signals using fully connected neural networks; although recent research suggests that more complex state representations only provide with marginal gains, if any [19]. Recently, extensive reviews are being carried out [20] [21] [22], indexing the different approaches taken and pushing the field towards maturity.

A common thread in most of the previous works is the need for approximations about the network being studied and the lack of pedestrian modelling and joint optimisation for vehicles and pedestrians' travel times.

As indicated in [21], pedestrian implementation has a high impact on learning performance, being often discarded as unimportant or left for future work save for three exceptions [23] [24] [25], one of which uses a genetic algorithm. In this paper we attempt to cover this gap in the literature, providing with a robust performance assessment of for RL agents serving both vehicles and pedestrians, using a variety of rewards, both novel and from the literature. These are used in a calibrated model of a real-world junction, using real geometry, calibrated demand, realistic sensory inputs and emulated traffic light controllers, making it immediately portable to the actual intersection.

III. Problem Definition

A. Markov Decision Processes and Reinforcement Learning

The problem is framed as a Markov Decision Process (MDP), satisfying the Markov property: given a current state s_t, the next state s_{t+1} is independent of the succession of previous states $\{s_{t-1}, s_{t-2}, ..., s_0\}$. An MDP is defined by the 5-element tuple:

1) The set of possible states $\mathcal{S}, s_i \in \mathcal{S}$.
2) The set of possible actions $\mathcal{A}, a_i \in \mathcal{A}$.
3) The probabilistic transition function between states $\mathcal{T}$.
4) The discount factor $\gamma \in [0, 1]$
5) The scalar Reward Function $\mathcal{R}$.

The objective of an MDP optimisation is to find an optimal policy π^*, mapping states to actions, that maximises the sum of expected discounted reward.

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

In the case of RL for UTC, $\mathcal{T}$ is unknown, only allowing for model-free RL.

Model-Free RL is an sub-field of RL covering how independent agents can take sequential decisions in an unknown environment and learn from these in order to obtain π^*. There are two main approaches: Policy-Based RL, in which states are mapped to a distribution of potential actions, and Value-Based RL, which is used in this paper and estimates the value (expected return) of the different state-action pairs under a given policy π as defined in Eq. 2.

$$V^{\pi}(s) = \mathbb{E}[R_t|s, \pi] \tag{2}$$

B. Q Learning and Value-Based RL

Q-Learning [27] is an off-policy model-free value-based RL algorithm. For any finite MDP, it can find an optimal policy which maximises expected total discounted reward, starting from the current state [28]. Q-Learning aims to learn an optimal action-value function $Q^*(s, a)$, defined as the total return after being in state s, taking action a and then following policy π^*.

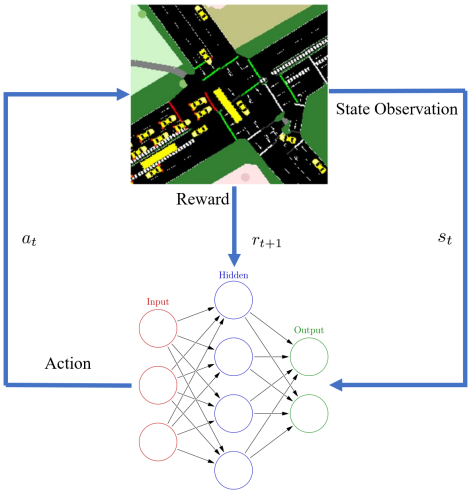

Fig. 1. Schematic representation of information transmission in Reinforcement Learning taking place between Agent and SUMO Environment

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\big[R_t|s = s_t, a = a_t, \pi^*\big] \tag{3}$$

Traditional table-based Q-Learning approximates $Q^*(s, a)$ recursively through successive Bellman updates,

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\big(y_t - Q(s_{t+1}, a)\big) \tag{4}$$

with y_t being the Temporal Difference(TD) target for the Q-function.

$$y_t = R_t + \gamma \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1}) \tag{5}$$

This table representation is not useful for high dimensionality cases, since the size of our table would increase exponentially, nor for continuous cases, since every distinct $s \in \mathcal{S}$ would require an entry.

C. Deep Q Network

One way of dealing with the issues of Q-Learning in high dimensional spaces is to use a Neural Network as function approximator, called Deep Q-Network (DQN) [29]. The Q-function approximation is denoted then in terms of the parameters θ of the DQN as $Q(s, a, \theta)$. One component of DQN that stabilises learning is the Target Network. DQN uses two neural networks: the main network with parameters θ, which approximates the Q-function, and the target network with parameters θ^- which provides the TD targets for the DQN updates and is updated every number of episodes by copying the weights $\theta^- \leftarrow \theta$. With $Q^{\pi}(s_{t+1}, a_{t+1}, \theta^-)$ representing the target network, it results in a TD target to approximate:

$$y_t = R_t + \gamma \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1}, \theta^-) \tag{6}$$

Fig. 2. Aerial view of Study Junction from Google Earth.

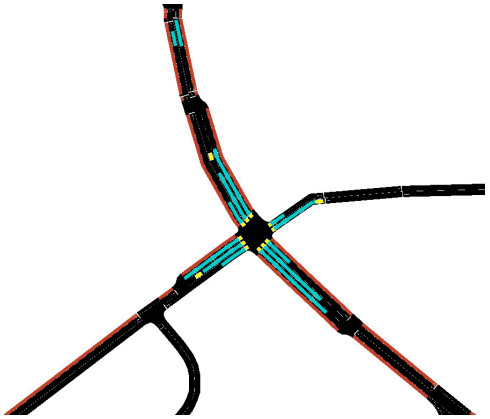

Fig. 3. Study Junction model in SUMO with a schematic representation of the areas covered by vision-based sensors.

IV. Methods

A. Reinforcement Learning Agent

The basic common agent used to obtain these results is a standard implementation of a DQN in PyTorch [31], optimising its weights via Stochastic Gradient Descent [32] using ADAM [33] as optimizer. The learning rate is $\alpha = 10^{-5}$ and the discount factor is $\gamma = 0.8$ for all simulations. The Neural Network in the agent uses 2 hidden, fully connected layers of sizes 500 and 1000 respectively, using ReLU as an activation function.

B. Reinforcement Learning Environment

The environment is modelled in the microscopic traffic simulator SUMO [26], representing a real-world intersection in Greater Manchester, UK. The junction consists of four arms, with 6 incoming lanes (two each in North-South orientation, and one each in East-West orientation) and 4 pedestrian crossings. The real-world site also contains 4 Vivacity vision-based sensors, able to supply queue length, speed and flow data. The demand and turning rations at the junction have been calibrated using 3.5 months of journey time and flow data collected by these sensors. The environment includes an emulated traffic signal controller, responsible of changing between the different stages in the intersection and enforcing the operational limitations, focused on safety. This includes enforcing green times, intergreen times, as well as determining allowed stages.

A stage is defined as a group of non-conflicting green lights (phases) in a junction which move at the same time.

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, which are primarily safety-related. The site features 4 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.

C. State Representation

The agent receives an observation of the simulator state as input, the contents of which remains constant across all experiments here presented. Each observation is a combination of the state of the traffic controller (which stage is active) and data from the sensors. The data from the sensors is comprised of the occupancy in each lane area detector and a binary signal representing whether the pedestrian crossing button has been pushed or not. A series of these, covering the last 12 seconds at a resolution of 0.6 seconds (20 state-entries) are provided to the agent.

While there is a variety of state representations available in the literature, often being more information dense, many of their features are not obtainable in practice with available sensors. As stated in Section II, recent studies [19] show that more information-dense states provide marginal improvements, if any, being possible to control isolated intersections with simple inputs.

D. Actions of the Agent

The junction is configured to have 4 available stages. The Agent is able to choose Stage 2, Stage 3 or Stage 4, yielding an action space size of 3. 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 [30]. If the agent chooses the same stage, that stage is extended by 0.6s, otherwise the controler 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 of Stage 1 as an intermediate state and

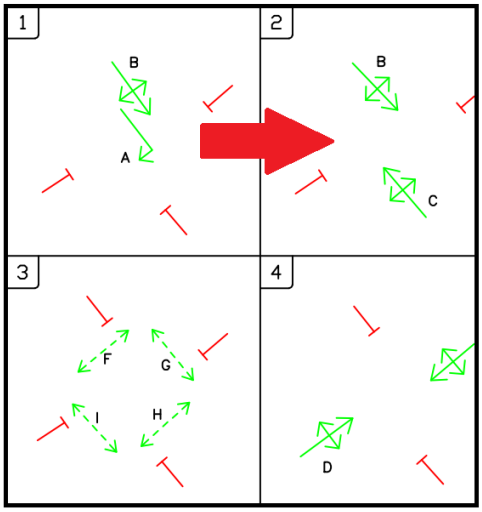

Fig. 4. Allowed stages and the phases that compose them. Stage 1 is an intermediate Stage, being necessary to go through it to reach Stage 2.

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.

E. Modal Prioritisation and Adjusting by Demand

The agent has to serve vehicles and pedestrians arriving at the intersection, seeking to jointly optimise the intersection for both modes of transport.

All the reward functions presented in this paper follow the same structure. The reward, as seen by the agent, will be a linear combination of an independently calculated reward for the vehicles and another for the agents, as it can be seen in Eq. 7.

$$R_t = \alpha * R_t^v + \beta * R_t^p; \qquad \alpha + \beta = 1 \tag{7}$$

In this way, α and β are the Modal Prioritisation coefficients for our rewards.

Of the rewards presented in the following section, those that were more sensitive towards the relative ratio of the demand between pedestrian and vehicles require of manual tuning of the modal prioritisation parameters. Even if this is undesirable from a modeller and operator point of view, since it partially counters the benefits that RL provides in terms of self-adjustment, they have been provided so potential users and researchers can evaluate the trade-offs between potential increased performance and increased configuration effort. The mentioned series will be identified by the weight applied to the pedestrians. As such, series identified as P80 and P95 represent those in which the weights were $\alpha = 0.2$, $\beta = 0.8$, and $\alpha = 0.05$, $\beta = 0.95$ respectively. Those series without an identifier did not require of modal prioritisation ($\alpha = \beta$).

Another addition that can be made to the rewards is to add a term scaling the difficulty with the demand level, implicitly accepting that higher demand typically worsens the performance of a network, independently of the actions of the controlling agent. These series are identified with the suffix AD (Adjusted by Demand).

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 M be the set of pedestrian occupancy sensors in the junction. Let V_t and P_t be respectively the set of vehicles on incoming lanes, and the set of pedestrians waiting to cross in the intersection at time t. Let s_v be the individual speeds of the vehicles, τ^v and τ^p the waiting times of vehicles and pedestrians respectively. Let ρ_v and ρ_p the vehicular and pedestrian flows across the junction 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. Lastly, let t_e^v and t_e^p be the entry times of vehicles and pedestrians to the area covered by sensors.

A. Queue Length based Rewards

1) Queue Length: Similar to [6], used in [17], the reward is the negative sum at time t of queues (q) over all (n) sensors.

$$R_t = -\alpha \sum_{n \in N} q_t^v - \beta \sum_{m \in M} q_t^p \tag{8}$$

2) Queue Squared: Seen in [12], this function squares the result of adding all queues.

$$R_t = -\alpha \Big(\sum_{n \in N} q_t^v \Big)^2 - \beta \Big(\sum_{m \in M} q_t^p \Big)^2 \tag{9}$$

3) Queues PLN: As Queue length, but dividing the sum by the phase length (Phase Length Normalisation), approximating the reward that the action generates by unit of time it is active.

$$R_t = -\frac{1}{t - t^p} \alpha \sum_{n \in N} q_t^v - \beta \sum_{m \in M} q_t^p \tag{10}$$

4) Delta Queue: The reward is the variation of the sum of queues between actions. Similar to Eqs. (14) and (17).

$$R_t = \alpha \Big(\sum_{n \in N} q_{t^p}^v - \sum_{n \in N} q_t^v \Big) + \beta \Big(\sum_{m \in M} q_{t^p}^p - \sum_{m \in M} q_t^p \Big) \tag{11}$$

5) Delta Queue PLN: As Delta Queue, but dividing the sum by the phase length (Phase Length Normalisation).

$$R_t = -\frac{1}{t - t^p} \Big(\alpha \big(\sum_{n \in N} q_{t^p}^v - \sum_{n \in N} q_t^v \big) - \big(\beta \sum_{m \in M} q_{t^p}^p - \sum_{m \in M} q_t^p \big) \Big) \tag{12}$$

B. Waiting Time based Rewards

These rewards use Modal Prioritisation weights.

1) Wait Time: The reward is the negative sum of time in queue accumulated since the last action.

$$R_t = -\Big(\sum_{v \in V_t} \tau_t^v + \sum_{p \in P_t} \tau_t^p \Big) \tag{13}$$

2) Delta Wait Time: Seen in [16], similar to Eq. 11. The reward is the variation in queueing time between actions.

$$R_t = \alpha \Big(\sum_{v \in V_t} \tau_{t_p}^v - \sum_{v \in V_t} \tau_t^v \Big) + \beta \Big(\sum_{p \in P_t} \tau_{t_p}^p - \sum_{p \in P_t} \tau_t^p \Big) \tag{14}$$

3) Waiting Time Adjusted by Demand: Negative sum of waiting time, adding a factor to scale it accordingly with an estimate of the demand ($\hat{d}$).

$$R_t = -\frac{1}{\hat{d}} \Big(\alpha \sum_{v \in V_t} \tau_t^v + \beta \sum_{p \in P_t} \tau_t^p \Big) \tag{15}$$

C. Delay based Rewards

These rewards use Modal Prioritisation weights.

1) Delay: Seen in [15]. Negative weighted sum of the delay by all entities. Delay is understood as deviation from the maximum allowed speed. For the pedestrians, the time in queue is used given that, from the point of view of the sensors, pedestrian presence is binary. Assuming a simulator time step of length δ:

$$R_t = -\Big(\alpha \sum_{v \in V_t} \sum_{t_e^p}^{t} \delta * \big(1 - \frac{s_v}{s_{max}}\big) + \beta \sum_{p \in P_t} \tau_t^p \Big) \tag{16}$$

2) Delta Delay: First seen in [7] and used in [20] [10] [13] [14] and [19]. Similar to Eq. (14). The reward is the variation between actions of the delay as calculated in Eq. (14).

$$R_t = \alpha \Big(\sum_{v \in V_t} \sum_{t^{pp}}^{t^p} ts * \big(1 - \frac{s_v}{s_{max}}\big) - \sum_{v \in V_t} \sum_{t^p}^{t} ts * \big(1 - \frac{s_v}{s_{max}}\big) \Big) + \beta \Big(\sum_{p \in P_t} \tau_{t^p}^p - \sum_{p \in P_t} \tau_t^p \Big) \tag{17}$$

3) Delay Adjusted by Demand: Same as in Eq. (18), introducing a scaling demand term.

$$R_t = -\frac{1}{\hat{d}} \Big(\alpha \sum_{v \in V_t} \sum_{t^p}^{t} ts * \big(1 - \frac{s_v}{s_{max}}\big) + \beta \sum_{p \in P_t} \tau_t^p \Big) \tag{18}$$

D. Average Speed based Rewards

1) Average Speed, Wait Time Variant: The vehicle reward is the average speed of vehicles in the area covered by sensors and normalised by the maximum speed. The pedestrian reward is the maximum between the sum of the waiting time of the pedestrian divided by a theoretical desirable maximum waiting time τ_{max} and 1. This produces two components of the reward $R_p, R_v \in [0, 1]$.

$$R_t = \alpha \frac{\sum_{v \in V_t} \frac{s}{s_{max}}}{\sum_{V_t} v} + \beta \min \big(\sum_{p \in P_t} \frac{\tau_t^p}{\tau_{max}}, 1 \big) \tag{19}$$

2) Average Speed, Occupancy Variant: Vehicle reward as in the previous entry. Pedestrian reward is the maximum between the sum of pedestrians waiting divided by a theoretical maximum desirable capacity p_{max} and 1.

$$R_t = \alpha \frac{\sum_{v \in V_t} \frac{s}{s_{max}}}{\sum_{V_t} v} + \beta \min \big(\sum_{p \in P_t} \frac{p}{p_{max}}, 1 \big) \tag{20}$$

3) Average Speed Adjusted by Demand, Demand and Occupancy Variants: As in the previous two entries, adding a multiplication by an estimation of the demand $\hat{d}$, scaling the reward with the difficulty of the task.

E. Throughput based Rewards

These rewards use Modal Prioritisation weights.

1) Throughput: The reward is the sum of the pedestrians and vehicles that cleared the intersection since the last action.

$$r_t = \alpha \sum_{t_p}^{t} \rho_v + \beta \sum_{t_p}^{t} \rho_p \tag{21}$$

VI. Experiments

A. DQN Agents Training

The training process covered 1500 episodes running for 3000 steps of length $\delta = 0.6$ seconds for a simulated time of 30 minutes (1800 seconds). The traffic demand is increased as the training advances, with the agent progressively facing sub-saturated, near-saturated and over-saturated scenarios, with a minimum of 1 vehicle/3 seconds (1200 vehicles/h) and a maximum of 1 vehicle/1.4 seconds (2571 vehicles/h).

For each reward function, 10 copies of the agent were trained, and their performance was compared against two reference systems. These were Maximum Occupancy (longest queue first) and Vehicle Actuated System D [34] (vehicle-triggered green time extensions), commonly used in the UK. The agent performing best in each class was selected for scoring.

B. Evaluation and Scoring

Each agent is tested and its performance scored over 100 copies of 3 different scenarios with different demand levels. Each evaluation has the same length as the training episodes, and the demand is kept constant during each run. These three scenarios are aimed to test the agents during normal operation, peak times and over-saturated conditions, and will be henceforth referred to as Normal, Peak and Over-saturated. Peak Scenario uses the level of demand observed in the junction that results in saturated traffic conditions under traditional controllers.

The Normal Scenario uses an arrival rate of 1 vehicle / 2.1 seconds (1714 vehicles/h). Peak Scenario uses an arrival rate of 1 vehicle / 1.7 seconds (2117 vehicles/h). Over-saturated Scenario uses an arrival rate of 1 vehicle / 1.4 seconds (2400 vehicles/h)

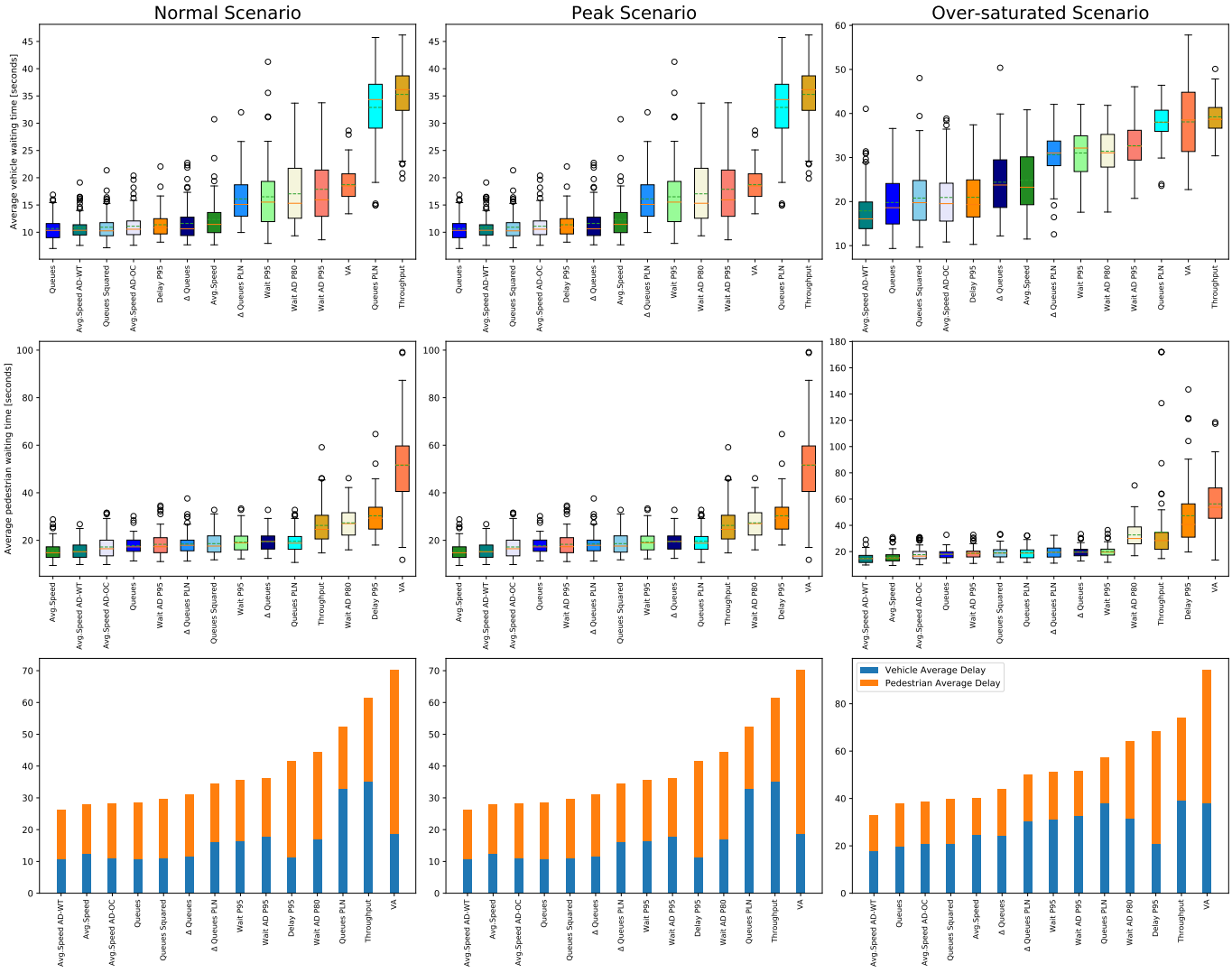

Fig. 5. Average waiting time for the best performing agents across all three demand scenarios.

VII. Results and Discussion

The results obtained from the simulations of the different reward functions for DQN agents are summarised in Fig. 5, including the performance of the 14 rewards found to have lower waiting times and seeming most desirable in practice, and detailed for all 31 rewards in Table I. In Fig. 5, the distribution of pedestrian and vehicle waiting times, and the combination of mean performances for both modes of transportation across 100 repetitions of each demand level are presented. Table I shows the mean waiting time for each distribution and their standard deviation, also calculated across all three demand levels.

The paper presents further evidence that RL agents can reach better performance than reference adaptive methods, being more evident when pedestrians are added. In the case of MO, the bad performance can be framed within the need of having more pedestrians queued than vehicles in any sensor in order to start the pedestrian stage. VA suffers due to its predisposition towards extending green times by 1.5s in the presence of any vehicle, making it more difficult to reach a state in which the pedestrian stage can be started. Both of these characteristics make the vanilla reference methods less suited for intersections including pedestrians than the RL methods presented, especially in situations with high demand.

On a global level, methods based on maximisation of the average network speed show the lowest global waiting times for pedestrians and vehicles combined across all demand levels while also obtaining some of the lowest spreads. They are closely followed by Queue minimisation, which nevertheless obtains the lowest average waiting times for vehicles in the Normal and Peak Scenarios, but falls behind in Over-saturated conditions and when dealing with pedestrians. Queue Squared minimisation has a comparable yet slightly lower performance, followed by Delta Queues and Delta Queues Phase Length Normalisation. This last reward has shown to obtain better performance with higher demand, which is consistent with less

TABLE I

AVERAGE WAITING TIME IN SECONDS FOR ALL AGENTS ACROSS DEMAND LEVELS

Scenario	Normal Scenario		Peak Scenario		Oversaturated Scenario	
	Vehicles	Pedestrians	Vehicles	Pedestrians	Vehicles	Pedestrians
Queues	7.87 ± 0.83	16.47 ± 3.95	10.68 ± 2.06	17.73 ± 3.64	19.80 ± 6.01	17.94 ± 3.48
Queues Sq.	7.79 ± 0.93	18.55 ± 4.47	10.92 ± 2.41	18.60 ± 4.81	20.80 ± 6.88	19.02 ± 4.38
Queues PLN	14.57 ± 4.91	20.31 ± 4.94	32.90 ± 6.36	19.59 ± 4.78	38.04 ± 3.93	19.28 ± 4.87
Δ Queues	8.34 ± 1.04	18.37 ± 3.94	11.63 ± 3.09	19.45 ± 3.75	24.40 ± 7.20	19.70 ± 3.80
Δ Queues PLN	10.37 ± 1.10	17.45 ± 3.59	16.11 ± 4.38	18.44 ± 4.45	30.64 ± 5.00	19.49 ± 4.32
Average Speed - Occ	7.86 ± 0.94	14.79 ± 3.97	12.34 ± 3.44	15.43 ± 3.84	24.84 ± 7.31	15.56 ± 4.49
Average Speed AD - Wait	8.20 ± 0.80	15.37 ± 3.48	10.85 ± 2.11	15.55 ± 3.84	17.89 ± 5.68	14.95 ± 3.80
Average Speed AD - Occ	7.85 ± 0.88	16.68 ± 4.83	11.10 ± 2.44	17.20 ± 4.93	20.93 ± 6.83	17.66 ± 5.01
Wait Time	7.80 ± 0.90	41.05 ± 19.40	14.65 ± 4.73	110.34 ± 59.56	28.82 ± 4.83	228.46 ± 159.81
Wait Time P80	8.20 ± 1.26	28.80 ± 9.05	14.94 ± 4.81	54.29 ± 35.00	30.01 ± 4.84	113.68 ± 52.00
Wait Time P95	8.26 ± 1.15	19.00 ± 4.67	16.51 ± 5.94	19.24 ± 4.44	31.02 ± 5.26	20.14 ± 4.16
Wait Time AD	7.83 ± 0.99	56.00 ± 30.22	14.84 ± 4.84	169.11 ± 92.44	27.52 ± 5.01	324.12 ± 212.37
Wait Time AD P80	8.25 ± 1.13	23.52 ± 5.73	17.05 ± 5.91	27.35 ± 6.29	31.43 ± 4.95	32.69 ± 9.67
Wait Time AD P95	8.48 ± 1.19	18.07 ± 4.75	17.88 ± 6.25	18.30 ± 5.02	32.67 ± 5.28	18.78 ± 4.37
Δ Wait Time	9.12 ± 1.23	82.57 ± 36.55	15.28 ± 5.09	326.07 ± 175.84	24.16 ± 6.77	594.03 ± 273.64
Δ Wait Time P80	8.94 ± 1.38	33.35 ± 17.34	16.68 ± 4.65	81.64 ± 49.48	30.38 ± 4.50	149.79 ± 105.07
Δ Wait Time P95	10.02 ± 1.66	42.36 ± 16.59	16.27 ± 5.33	72.27 ± 44.89	26.88 ± 6.22	174.85 ± 109.01
Delay	6.39 ± 0.40	849.52 ± 318.33	8.59 ± 0.89	849.52 ± 318.33	14.43 ± 3.16	849.52 ± 318.33
Delay P80	8.39 ± 1.10	46.78 ± 16.52	11.52 ± 2.36	78.91 ± 35.73	20.93 ± 7.02	143.27 ± 72.39
Delay P95	7.92 ± 0.89	26.21 ± 4.86	11.38 ± 2.42	30.30 ± 7.64	20.99 ± 6.29	47.34 ± 23.27
Delay AD	6.71 ± 0.43	811.38 ± 352.38	8.79 ± 0.96	811.38 ± 352.38	14.08 ± 3.31	811.38 ± 352.38
Delay AD P80	7.74 ± 0.81	44.55 ± 17.51	10.68 ± 1.95	122.05 ± 112.18	18.92 ± 6.46	404.54 ± 252.47
Delay AD P95	7.83 ± 0.84	48.76 ± 24.28	11.62 ± 3.02	180.59 ± 123.18	21.77 ± 6.82	425.35 ± 234.33
Δ Delay	11.18 ± 2.93	211.41 ± 116.86	26.98 ± 6.75	546.51 ± 263.71	34.97 ± 3.45	393.81 ± 267.95
Δ Delay P80	10.62 ± 2.34	66.46 ± 30.32	20.51 ± 6.04	180.64 ± 107.80	29.70 ± 4.97	307.76 ± 218.11
Δ Delay P95	8.23 ± 1.29	99.40 ± 59.76	15.22 ± 4.97	221.92 ± 133.24	25.35 ± 6.43	398.13 ± 240.03
Throughput	18.71 ± 4.79	23.60 ± 6.88	35.28 ± 5.60	26.26 ± 8.14	39.24 ± 3.72	34.86 ± 28.54
Throughput P80	35.53 ± 10.87	51.96 ± 31.20	47.60 ± 5.99	65.91 ± 37.86	47.85 ± 5.15	84.93 ± 49.08
Throughput P95	26.28 ± 8.81	101.07 ± 65.21	56.39 ± 10.72	130.98 ± 84.11	74.10 ± 13.94	74.46 ± 57.96
Vehicle Actuated System D	10.62 ± 1.17	38.36 ± 12.66	18.73 ± 2.92	51.62 ± 16.50	38.10 ± 8.26	56.32 ± 19.58
Maximum Occupancy	6.92 ± 0.54	196.09 ± 130.04	10.02 ± 1.75	397.20 ± 213.06	21.57 ± 5.10	596.32 ± 253.80

variance in the arrival times and makes it an option that could be further explored for permanently congested intersections.

Prioritised rewards based on Waiting Time had acceptable performance, although they are was very susceptible to the changes in the modal prioritisation weights. This is similar to the behaviour shown by the Delay-based rewards, which overall perform worse. Without a weight configuration heavily favouring the pedestrians, these reward functions were found to converge for vehicles only, obtaining the lowest vehicle waiting times overall in the case of the Delay functions, at the expense of rarely, if ever, serving pedestrians. The suitability of a given choice of modal prioritisation weights is further affected by the functional form of the reward. In the results, it can be observed that while in general the choice $(\alpha = 5, \beta = 95)$ obtains better results (e.g. Wait Time and Delay), for certain functional choices the prioritisation $(\alpha = 20, \beta = 80)$ produces best results, which would not be the case if the suitability of the weights was only affected by the relative demand ratios between vehicles and pedestrians. This is the case with Throughput based functions, which unlike the Wait and Delay functions, obtained lower waiting times with equal modal weights, and a general increase as the weights become more skewed towards the pedestrians.

Rewards using Differences in Delay or Wait Time, having good performance in the literature, were found either not to converge for pedestrians or to produce mediocre results.

The addition of a demand scaling term generates, in general, a slight improvement in waiting times across the rewards using Wait Time and Delay, particularly at higher demand levels. Average Speed rewards do not seem to benefit from this term, with all variants scoring similarly.

Overall, the dominance shown by speed maximisation methods could be attributed to several factors. On one side, Average Speed based functions It can also be argued that due speed maximisation rewards are better suited for the underlying MDP than others, due to their independence of the correspondence between agent actions and time-steps in the environment, . In the specific case of RL for UTC, the values of the reward received by the agent using a reward based on Queues, Delay, Wait or Throughput are a function of the length of the phase that generated them. Lastly, speed maximisation and queue minimisation have an extra benefit that makes them into

serious candidates for expansive real-world use: the lack of need for modal prioritisation tuning. One of the main selling points of ML and RL methods stems from their ability to perform equal or better than traditional systems at a lower cost. However, a lengthy manual tuning process in order to find the exact weights for a given situation is not only untranslatable to any other location but also may not result in reduced planning times compared with traditional control. The lack of need for manual tuning, especially in the case of Average Speed functions, which are specifically crafted to avoid it, make them more applicable in a wider and faster manner than any of the other reward functions here presented.

A limitation of this paper, is that the results are only relevant in the case of value-based DQN agents as introduced in Section III and Section IV, not being relevant for CNN or Policy Gradient architectures.

This work could be extended to account for other modes of transportation, performing a similar optimisation based on different vehicle classes (buses, cyclist, personal vehicles, trucks, etc). The optimisation could seek to prioritise them based on different criteria (e.g. priority to cyclists and public transport during rush hours).

ACKNOWLEDGMENT

This work was part funded by EPSRC Grant EP/L015374 and part funded by InnovateUK grant 104219. We are also grateful to W. Chernikoff, Toyota Mobility Foundation and The Alan Turing Institute for support in the initial stages.

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] Lowrie, P. R. (1990). Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic.

[4] Wiering, M. A. (2000). Multi-agent reinforcement learning for traffic light control. In Machine Learning: Proceedings of the Seventeenth 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] 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.

[7] 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.

[8] Abdoos, M., Mozayani, N., & Bazzan, A. L. (2011, October). Traffic light control in non-stationary environments based on multi agent Q-learning. In 2011 14th International IEEE conference on intelligent transportation systems (ITSC) (pp. 1580-1585). IEEE.

[9] 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.

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

[11] Liang, X., Du, X., Wang, G., & Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. IEEE Transactions on Vehicular Technology, 68(2), 1243-1253.

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

[13] 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.

[14] 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.

[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] 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.

[18] Genders, W., & Razavi, S. (2019). Asynchronous n-step Q-learning adaptive traffic signal control. Journal of Intelligent Transportation Systems, 23(4), 319-331.

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

[20] 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.

[21] Haydari, A., & Yilmaz, Y. (2020). Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey. arXiv preprint arXiv:2005.00935.

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

[23] Turky, A. M., Ahmad, M. S., Yusoff, M. Z. M., & Hammad, B. T. (2009, July). Using genetic algorithm for traffic light control system with a pedestrian crossing. In International Conference on Rough Sets and Knowledge Technology (pp. 512-519). Springer, Berlin, Heidelberg.

[24] Liu, Y., Liu, L., & Chen, W. P. (2017, October). Intelligent traffic light control using distributed multi-agent Q learning. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-8). IEEE.

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

[26] 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.

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

[28] Melo, F. S. (2001). Convergence of Q-learning: A simple proof. Institute Of Systems and Robotics, Tech. Rep, 1-4.

[29] 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.

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

[31] 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).

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

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

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