Reward Functions for Real-World Pedestrian and Vehicular Intersection Control through Reinforcement Learning

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*— Reinforcement Learning is proving a successful tool that can manage urban intersections with a fraction of the effort required to curate traditional traffic controllers. However, literature on the simultaneous introduction and control of pedestrians to such intersections is very scarce. Furthermore, it is unclear what traffic state variables should be used as reward to minimise waiting times. This paper robustly evaluates 30 different Reinforcement Learning reward functions for controlling a real-world intersection serving pedestrians and vehicles covering the main traffic state variables available via modern vision-based sensors. Some rewards proposed in previous literature solely for vehicular traffic are extended to pedestrians while new ones are proposed. We use a calibrated model in terms of demand, sensors, green times and other operational constraints of a real intersection in Greater Manchester, UK, to which the agent has been since deployed. The assessed rewards can be classified in 5 groups depending on the quantities 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, 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, demonstrating better performance than other previously proposed methods.**

I. Introduction

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

First, fixed time controllers, and later, adaptive systems have been used to further optimise the global traffic flow in 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, at a much lower marginal cost than would be feasible with traditional actuated sensors. As a side effect of these developments the area covered by sensors is ever increasing, also becoming possible to direct some of these towards pedestrians. This has allowed the development of novel smart control approaches, using real-time data to deliver cheap and responsive 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 existing works restrict themselves to vehicles only, not attempting to jointly optimise vehicular and pedestrian travel times, even though pedestrians are present in the great majority of real urban intersections.

This paper compares the performance of 30 different reward functions used by Deep Q-Network agents, split into 5 different classes based on the quantities 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 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 their analytical expressions. Section VI contains details about the training and evaluation of the agents. Lastly, Section VII provides the experimental results and discusses them.

II. Related Work

RL for UTC has been previously explored and discussed in a variety of research, aiming to eventually substitute existing adaptive control methods such as SCOOT [1], MOVA [2] and SCATS [3]. The field has evolved from early inquiries about its theoretical potential use [4] [5] [6] [7] [8], to progressively more applied and realistic scenarios that look towards real-world use and deployment. Recent works use different quantities (delay, queues, waiting time, throughput, ...) in the reward function of the agents, however, it is not clear what benefits are provided from choosing each of them. The different quantities used as reward are thoroughly indexed in [9] [10] [11], although no direct performance comparisons are made. Regarding the specific data used as inputs, different methods have been taken, such as pixel-based vectors passed to a CNN [12] [13] [14], per-lane state signals using fully connected neural networks [15] [16] [17], or hybrid approaches [18] [19]

[20]. Recent research suggests that more complex state representations, such as pixels, only provide marginal gains when compared with state vectors using per-lane measurements [21], so in this paper the per-lane encoding is taken. A common thread in most 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, which this paper addresses by using a non-simplified model in terms of topology and systems modelling, as well as by introducing the control of pedestrian crossings in conjunction with the traffic intersection. As indicated in [10], pedestrian implementation has a high impact on learning performance, being often discarded as unimportant or left for future work in the literature, save for two exceptions [22] [23], the first of which uses a genetic algorithm instead of RL, and the second explores a single reward function. In this paper we attempt to cover this gap in the literature, providing a robust performance assessment of RL agents simultaneously serving both vehicles and pedestrians, using a variety of rewards, both novel and from the literature, attempting to uncover what state variables should be used in the reward to obtain the best performance. These are applied to a RL agent in a calibrated model of a real-world junction, using real geometry, calibrated demand, realistic sensor inputs and emulated traffic light controllers, to which some of these agents have been deployed to control real traffic in it since these experiments took place. This paper delivers the future work deferred from [24] in terms of adding pedestrians, performing the required reward function re-engineering and shifting the focus towards multi-objective optimisation.

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

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

B. Q Learning and Value-Based RL

Q-Learning [25] 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 any state [26]. 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^*.

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

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 α the learning rate and y_t 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 dimensional 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 addressing the issues of Q-Learning in high dimensional spaces is to use neural networks as function approximators. This approach is called Deep Q-Network (DQN) [27]. The Q-function approximation is denoted then in terms of the parameters θ of the DQN as $Q(s, a, \theta)$. DQN stabilises the learning process by introducing a Target Network that works alongside the main network. The main network with parameters θ, approximates the Q-function, and the target network with parameters θ^- provides the TD targets for the DQN updates. The target network 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:

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

IV. Methods

A. Reinforcement Learning Agent

The agent used to obtain these results is a standard implementation of a DQN in PyTorch [28], optimising its weights via Stochastic Gradient Descent [29] using ADAM [30] 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.

Algorithm 1: Schematic Learning Process

```
Create main network with random weights θ;
Create target network with random weights θ⁻;
Create replay memory M with capacity L;
Define frequency F for copying weights to target
  network;
for each episode do
    measure initial state s_0;
    while episode not done do
        select action a_t according to ε-greedy policy;
        implement a_t;
        advance simulation until next action is 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 minibatch 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
    Stochastic Gradient Descent on θ over all
      (x_i, y_i) ∈ b;
    if number of episode is multiple of F then
        θ⁻ ← θ
    end
end
```

B. Reinforcement Learning Environment

The environment is modelled in the microscopic traffic simulator SUMO [31], representing a real-world intersection in Greater Manchester, UK. The junction consists of four arms, with 6 incoming lanes (two each in the north-south orientation, and one each in the east-west orientation) and 4 pedestrian crossings. The real-world site also contains 4 Vivacity vision-based sensors, able to supply occupancy, queue length, waiting time, speed and flow data. The demand and turning ratios 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 for changing between the different stages in the intersection and enforcing the operational limitations, which are focused on safety. This includes enforcing green times, inter-green times, as well as determining allowed stages. A stage is defined as a group of non-conflicting green lights (phases) in a junction which change 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 data available to the agent is restricted to what can be obtained from the sensors.

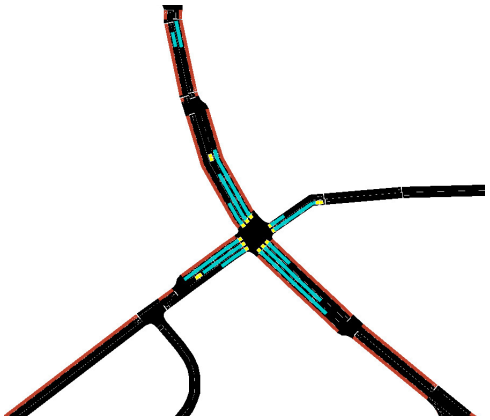

Fig. 1. Study Junction model in SUMO with a schematic representation of the areas covered by vision-based sensors.

C. State Representation

The agent receives an observation of the simulator state as input, using the same state information 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. The agent receives a concatenation of the last 20 measurements at a time, covering the previous 12 seconds at a resolution of 0.6 seconds, which is the sensor sampling frequency.

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 services a protected right turn coming from the north. It is used by the traffic light controller, as a transitional step for reaching Stage 2, as required by the transport authority. Stage 2 deals with the traffic in the north-south orientation. Stage 3 is the pedestrian stage, setting all pedestrian crossings to green, and all other phases to red. Stage 4 services the roads in the east-west orientation, which have considerable demand.

Once the controller has had a stage active for the minimum green time duration, the agent is requested to compute the value of all potential state-action pairs (i.e. the value of other stages given the current state) once per time-step. From these, the action with the highest expected value is selected following an ϵ-greedy policy [32]. Should the agent choose the same action, the current stage will be extended for a further time-step (0.6 seconds). There is no built-in limit to the maximum number of said extensions, leaving it for the agent to learn the optimal green time for any given situation. If a different stage is chosen, then the controller will proceed to the inter-green transition between them.

There are 2 situations that further add to the complexity of this control process:

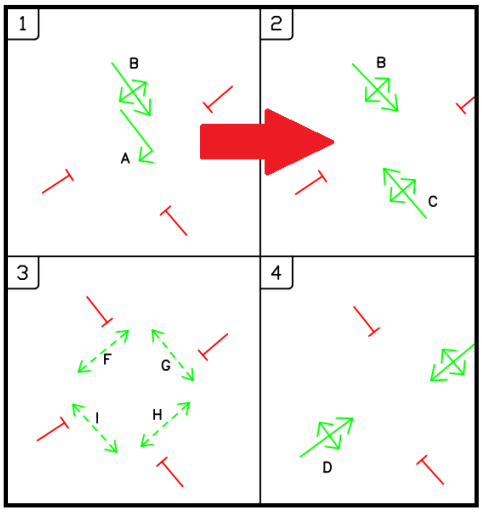

Fig. 2. Allowed stages and their phases. Stage 1 is an intermediate Stage, which is necessary to go through to reach Stage 2. Phase A lasts a minimum of 4 s, the rest last for at least 7s. New actions are requested after this time.

1) Variable number of extensions, and hence length of the stages, creates a distribution of values over the state-action pairs in most rewards, which the agent must approximate. The variance of this distribution will be higher than the variance that would be obtained using constant stage length.
2) The requirement that Stage 1 must be used as an intermediate step to reach Stage 2 implies less certainty in the control process than in other stages, since there is an unaccounted dilated temporal horizon between the state that triggered the action, and the effects of said action over the state variables.

E. Modal Prioritisation and Adjusting by Demand

The agent serves 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 and calculated for each action, 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.

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

In this way, a and b are the Modal Prioritisation coefficients for our rewards, with R^v, R^p being respectively the vehicular and pedestrian 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 manual tuning of the modal prioritisation parameters. While undesirable from a modeller and operator point of view since it partially counters the benefits that RL provides in terms of self-adjustment, they are 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 $a = 0.2$, $b = 0.8$, and $a = 0.05$, $b = 0.95$ respectively. Those series without an identifier did not require modal prioritisation ($a = b$).

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, independent of the actions of the controlling agent. These series are identified with the suffix AD (Adjusted by Demand).

V. Reward Functions

All reward functions tested are presented in this section with their analytical expressions.

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 in 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 be 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 [16], the reward is the negative sum at t of queues (q) over all (n, m) sensors.

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

2) Queue Squared: As seen in [19], this function squares the result of adding all queues.

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

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.

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

4) Delta Queue: The reward is the variation of the sum of queues between actions.

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

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

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

B. Waiting Time based Rewards

These rewards require Modal Prioritisation weights.

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

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

2) Delta Wait Time: As seen in [12], the reward is the variation in queueing time between actions.

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

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

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

C. Delay based Rewards

These rewards require Modal Prioritisation weights.

1) Delay: As seen in [20]. Negative weighted sum of the delay by all entities. Delay is understood as deviation from the maximum allowed speed. For the pedestrians, the sum of the individual times in queue τ^p is used given that, from the point of view of the sensors, pedestrian presence is binary. Assuming a simulator time step of length δ:

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

2) Delta Delay: First seen in [7] and used in [18] [13] [14] and [21]. The reward is the variation between actions of the delay as calculated in Eq. (14).

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

3) Delay Adjusted by Demand: Same as in Eq. (16), introducing a scaling demand term.

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

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 minimum between the sum of the waiting time of the pedestrian divided by a heuristically set maximum desirable waiting time τ_{max} and 1. This produces two components of the reward $R_p, R_v \in [0, 1]$.

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

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

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

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

E. Throughput based Rewards

These rewards require Modal Prioritisation weights.

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

$$r_t = a \sum_{t_p}^{t} \rho_v + b \sum_{t_p}^{t} \rho_p \tag{21}$$

VI. Experiments

A. DQN Agents Training

The training process covers 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 are trained, and their performance was compared against two reference systems. These are Maximum Occupancy (MO) (longest queue first) and Vehicle Actuated System D (VA) [33] (vehicle-triggered green time extensions), which is commonly used in the UK. The agent performing best against the reference systems in each class is selected for detailed scoring.

B. Evaluation and Scoring

Each selected agent is tested and its performance scored over 100 copies of 3 different scenarios with different demand levels. Each evaluation is the same length as the training episodes, with the demand 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 Scenarios. 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)

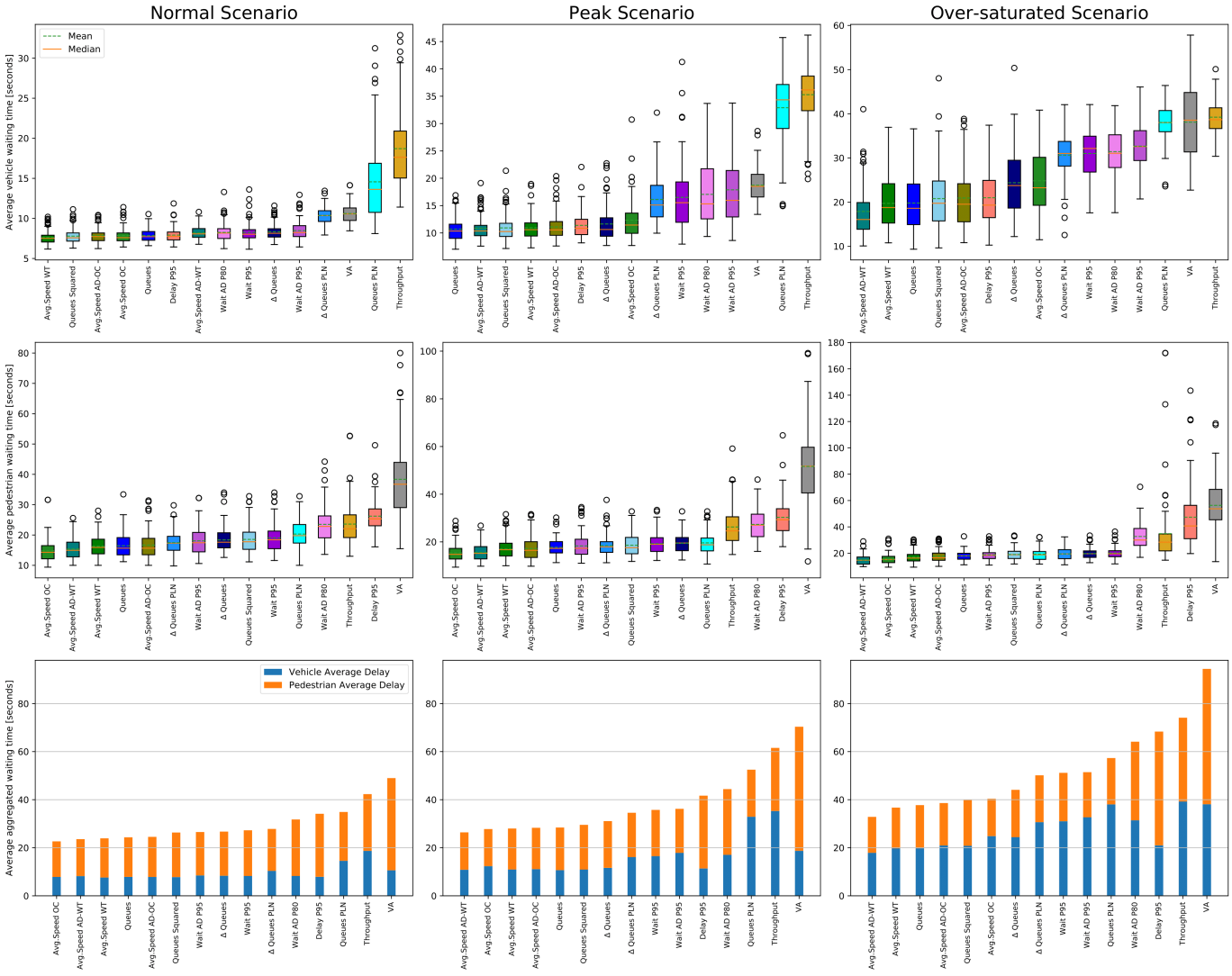

Fig. 3. Top row: Average vehicular waiting time distribution for the fifteen best performing agents across demand levels. Middle row: Average pedestrian waiting time distribution across demand levels. Bottom row: Aggregated vehicular and pedestrian mean performance across demand levels.

VII. Results and Discussion

The results from the simulations of the different reward functions are summarised in Fig. 3, including the performance of the 15 rewards found to have lower waiting times and seeming most desirable in practice. They are detailed for all 30 rewards in Table I. In Fig. 3, 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 results display further evidence that RL agents can reach better performance than reference adaptive methods, more evidently so 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 reference methods less suited for intersections including pedestrians than the RL methods presented in Fig. 3, especially in situations of high demand.

At 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, as shown in the case with no pedestrians [24]. Their performance is closely followed by Queue minimisation, which 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 worse performance, followed by Delta Queues and Delta Queues PLN. This last reward has shown

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 - Wait	7.61 ± 0.84	16.31 ± 3.82	10.94 ± 2.19	17.05 ± 4.67	10.62 ± 1.17	38.36 ± 12.66
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

to obtain better performance with higher demand, which is consistent with it generating less variance in the state, since it is modelling for arrival rates given an action, and makes it an option that could be further explored for permanently congested intersections. Prioritised rewards based on Waiting Time show acceptable performance, but also a high sensitivity to the changes in the modal prioritisation weights. This is similar to the behaviour shown by the Delay-based rewards, which overall perform worse, potentially due to the need to use Wait Time for pedestrians, mixing the state variables, although this does not seem to be an issue for average speed based rewards. 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 $P95$ obtains better results (e.g. Wait Time and Delay), for certain functional choices the prioritisation $P80$ is the one producing the 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 wait time 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.

Overall, dominance shown by speed maximisation methods could be attributed to several factors. Average Speed based functions, as Queue based functions, obtain an instantaneous snapshot of a magnitude that does not intrinsically grow over time, as opposed to Delay, Wait and Throughput, so it exclusively encodes information about the moment the action is requested. It can also be argued that speed maximisation

rewards are not affected by 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, making them theoretically less suitable for the underlying MDP than speed maximisation. 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 junction is not only untranslatable to any other intersection, but may also not result in reduced planning and execution 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 this, make them in our view more applicable in a wider and faster manner than any of the other reward functions here presented.

One 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, and not 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, cyclists, 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 or weighting vehicles according to the expected number of passengers).

ACKNOWLEDGMENT

This work was funded by EPSRC Grant EP/L015374 and by InnovateUK grant 104219. Vivacity Labs thanks Transport for Greater Manchester, Immense, and InnovateUK.

REFERENCES

[1] 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).

[2] 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.

[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] 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.

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

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

[12] 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.

[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] 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.

[16] 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.

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

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] Cabrejas Egea, A., Howell, S., Knutins, M., & Connaughton C. (2020, October). Assessment of Reward Functions for Reinforcement Learning Traffic Signal Control under Real-World Limitations. IEEE Systems, Man, and Cybernetics (October 2020) (Accepted).

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

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

[27] 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.

[28] 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).

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

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

[31] 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.

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

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