Reward Function Design in Multi-Agent Reinforcement Learning for Traffic Signal Control

Behrad Koohy[1], Sebastian Stein[1], Enrico Gerding[1] and Ghaithaa Manla[2]

[1]*University of Southampton, University Road, Highfield, Southampton, SO17 1BJ*

[2]*Yunex Traffic, Sopers Lane, Poole, Dorset, BH17 7ER*

Abstract

In recent years, there has been increased interest in Reinforcement Learning (RL) for Traffic Signal Control (TSC), with implementations of RL touted as a potential successor to the current commercial solutions in place. Commercial systems, such as Microprocessor Optimised Vehicle Actuation (MOVA) and Split, Cycle, and Offset Optimisation Technique (SCOOT), can adapt to the changing traffic state, but do not learn the specific traffic characteristics of an intersection, and leave much to be desired when performance is compared to the potential benefits of using RL for TSC. Furthermore, distributed RL can provide the unique benefits of scalability and decentralisation for road infrastructure. However, using RL for TSC introduces the problem of non-stationarity where the changing policies of RL agents, tasked with optimal control of traffic signals, directly impacts the observed state of the system and therefore the policies of other agents. This non-stationarity can be mitigated through careful consideration and selection of an appropriate reward function. However, existing literature does not consider the impact of the reward function on the performance of agents in a non-stationary environment such as TSC. In this paper, we select 12 reward functions from the literature, and empirically evaluate them compared to a baseline of a commercial solution in a multi-agent setting. Furthermore, we are particularly interested in the performance of agents when used in a real-world scenario, and so we use demand calibrated data from Ingolstadt, Germany to compare the average waiting time and trip duration of vehicles. We find that reward functions which often perform well in a single intersection setting may not outperform commercial solutions in a multi-agent setting due to their impact on the demand profile of other agents. Furthermore, the reward functions which include the waiting time of agents produce the most predictable demand profile, in turn leading to increased throughput than alternatively proposed solutions.

Keywords

Traffic Signal Control, Intelligent Traffic Management, Reinforcement Learning, Problem of Non-Stationarity, Multi-Agent Reinforcement Learning

1. Introduction

Reinforcement Learning (RL) for Traffic Signal Control (TSC) is an area which has been investigated in detail as a potential improvement on the current adaptive systems in use. Current commercially available systems do not use RL, and require manual setup of signal timings for each intersection, something which can be time-consuming to do and can have a negative impact on traffic flow if not configured correctly. In the UK, MOVA [1] (Microprocessor Optimised Vehicle Actuation) and SCOOT [2] (Split, Cycle, and Offset Optimisation Technique)

12th International Workshop on Agents in Traffic and Transportation (ATT 2022)

bk2g18@soton.ac.uk (B. Koohy); ss2@ecs.soton.ac.uk (S. Stein); eg@ecs.soton.ac.uk (E. Gerding); manla.ghaithaa@yunextraffic.com (G. Manla)

CEUR Workshop Proceedings (CEUR-WS.org)

are the most widely implemented commercial systems, with the latter being used mainly for regions of up to 30 traffic signal junctions. While adaptive (extending green signals when traffic demand is high in a given direction), these algorithms do not use RL to learn the specific characteristics of a traffic signal. The design of these algorithms was completed in the 1980s, and the iterative improvements made since then have not taken advantage of the vast amount of information available now from roadside sensors. In addition to this, modern approaches to the TSC problem can employ more advanced data sources such as traffic cameras, and information from connected and autonomous vehicles, allowing for a more accurate picture of the traffic flow through a road network. Furthermore, this allows for prioritisation of certain types of traffic, where appropriate, such as allowing heavy goods vehicles (HGVs) to pass through lights and avoid deceleration (followed by acceleration), or clearing the road network in a certain direction to allow for easier passage of emergency vehicles attending to an emergency.

RL based approaches for TSC, whilst not exposed to the decades of development which current approaches in use have had, have still been shown to outperform well-calibrated systems in simulations [3]. Current state-of-the-art approaches make use of some innovative methods such as junction pressure [4, 5], convolutional neural networks [6] and graph attention networks [7].

Introducing independent RL agents at each intersection within a road network has a number of benefits. Firstly, it allows for easier scalability when compared to a centralised system as changes to the road network such as the addition of new roads or traffic signals can be tolerated by introducing new agents, rather than re-training or modifying a central system. Secondly, the state and action space of a centralised agent increases exponentially when more traffic signals are introduced, leading to the curse of dimensionality [8]. Independent RL agents deployed to each intersection suffer from neither of these problems, and each agent can learn the specific characteristics of the intersection under their control. However, a problem emerges when we consider the simultaneous learning process which is used to train the independent agents. As an agent updates their policy to be optimal from their observations, the optimal policy for the agents at connected intersections from this agent may change based on the impact to the demand profile of their intersection. We refer to this as the problem of non-stationarity.

In this paper, we evaluate reward functions from the literature and review them in the context of a real-world multi-agent scenario, using calibrated data from Ingolstadt, Germany, to test them, including an implementation of a commonly used commercial solution, MOVA, as the baseline. We highlight the impact of reward functions on the ability of the agent to learn, and how solutions to the problem of non-stationarity may not be feasible in the real world when used in the TSC context. To evaluate the performance of different reward functions, we compare the waiting time and trip duration of vehicles.

2. Background and Related Work

The non-stationary problem is one which has been observed in many multi-agent RL contexts [9, 10, 11]. We define the non-stationary problem as when independent agents are in an environment where they take actions to optimise their policy, aided by a reward function, but the actions of these agents impact the surrounding agents. The changing environment can be referred to as non-stationary. When thought of in the context of the TSC problem, we encounter a changing environment when agents change their policies to one which they believe is more optimal. This change can impact the demand characteristics which other agents see, and may result in their own policies no longer being optimal. Furthermore, in addition to the reason of changing road networks, as well as the curse of dimensionality, it is also not feasible to have a centralised system to learn and control traffic timings as computational complexity exponentially grows in the numbers of lanes and junctions [12].

A potential solution to this problem is to employ an actor-critic (AC) algorithm [13] for each agent, with a common critic. In the context of TSC, multi-agent AC and the derivatives have been implemented and tested, with Feudal AC [14] evaluated by Ault et al. [6], and their investigation found that they perform similarly to Deep Q-Learning algorithms but take significantly longer to converge on the solution. An alternative approach to the problem of non-stationarity is to introduce a form of communication between agents. Foerster et al. [15] introduce a Deep Distributed Recurrent Q-Network, where agents share hidden layers and are tasked with developing a communication protocol to expedite the solving of communication-based coordination tasks. Sukhbaatar et al. [16] introduced the architecture of CommNet, which incorporates a communication message, the average of the previous hidden layers from all other agents into the input of each layer of the agent. However, for both AC approaches and communication between agents, the issues around scalability remain, and may require the critic or communicative agent to be retrained when changes are made to the road network.

In work by Cabrejas-Egea et al. [17], an assessment of 15 common reward functions, aggregated into 5 groups (queue-length based rewards, waiting time based rewards, delay based rewards, average speed based rewards and throughput based rewards), is performed and it is found that average speed maximisation reduces the average vehicle waiting time. However, this was performed in a single agent scenario, with one junction. Whilst maximising speed may perform best in isolated junctions, it is unknown how nearby junctions will be affected. Wei et al. [18] provides more details on alternative approaches in RL for TSC, including the state and reward functions employed and the dataset used to verify results.

Moreover, it is suggested that there is a significant gap between the performance of agents in synthetic benchmarks and calibrated data from the real world. Ault et al. [6] compared implementations of MPLight [19], FMA2C [14] and DQN based approaches [20] (among others) and concluded that whilst synthetic benchmarks can prove challenging for RL agents, there is a difference in performance between them and calibrated data. There is a gap in the literature to explore whether this continues into reward functions as well. Specifically, we are interested in the performance of reward functions in realistic traffic scenarios.

3. Problem Formulation

The TSC problem can be formulated as a Partially Observable Markov Decision Process (POMDP) [21] $< S, A, \mathbb{P}, R, \Omega, O, \gamma >$, defined as S, the set of states, A, the set of possible actions, $\mathbb{P}(s_t, a, s_{t+1}) : S \times A \times S \to [0, 1]$, the state transition function, $R(s, a, s') : S \times A \to \mathbb{R}$ which describes the likelihood of transitioning from s to s' when action a is taken, Ω, the set of observations, O, the set of conditional observation probabilities $O : S \times A \times \Omega \to [0, 1]$, and γ, the discount factor. We define this problem as a POMDP rather than a standard Markov Decision

Process due to the limitations in sensor capability and knowledge of the global state. Therefore, S can be defined as the state of the system, contrasted to Ω, the observations of the system state from the sensors at an intersection.

The choice of reward function R is important to the performance of our agents. In the TSC problem, the high-level aim is to maximise throughput of vehicular traffic across all traffic signals. Part of increasing throughput is to reduce vehicle waiting time and increase average speed as these two factors directly contribute to how quickly vehicles reach their destination. However, for the same reasons that it is not feasible to use a centralised single agent to control all the intersections, it is not feasible to incorporate the total throughput of all agents as a reward function. Furthermore, the problem of non-stationarity is still prevalent as the reward that agents see will now be explicitly and directly impacted by the policies of other agents.

In the context of TSC, we define a phase φ as a group of non-conflicting green lights at a signalised intersection, and a signalised intersection as having a finite set of phases Φ such that $\varphi \in \Phi$. In each intersection, we construct the state space (S) as a combination of the current phase the intersection has selected and the observation of the current traffic state. In addition to this, we can define the action space (A) for an agent as Φ. If the selected phase is a change to the current phase, there must be a mandatory yellow phase interjected, and the selected phase must also be chosen for longer than the minimum limit [22]. Each intersection includes an emulated traffic signal controller, and if an agent selects a different phase or an action which does not fulfill the mandatory requirements, the traffic controller enforces the legal safety requirements. The reward function R differs between implementations, and how to choose this is the focus of our paper.

It should also be mentioned that by describing the TSC problem as an POMDP, we are assuming that the TSC problem fulfils the Markov property, that is, that the process of TSC is memory-less (the result of the next state only depends on the action taken from the current state). Formally, given a state history S_H:

$$S_H = S_t, S_{t+1}, ..., S_\infty \tag{1}$$

then, if following the Markov Property

$$\mathbb{P}(S_{t+1}|S_H) = \mathbb{P}(S_{t+1}|S_t, S_{t+1}, ..., S_\infty) = \mathbb{P}(S_{t+1}|S_t) \tag{2}$$

When applied to the context of TSC, it may seem like this assumption does not hold true, as traffic has known periodic cycles of greater and lesser demand. Son et al. [23] showed that fluctuations in traffic flow (and seasonality) can be modelled using a Fourier Transform, and used to make predictions about future traffic predictions. However, this is only possible when the states of traffic signals are viewed over a period of days to weeks, and in the TSC problem, this temporal horizon is very small (seconds to minutes) and in a resolution below the required amount to make assumptions regarding traffic seasonality. With the assumption that TSC does fulfil the Markov property, and taking into consideration the computational complexity of solving POMDP, we model the problem as a regular MDP.

When RL is applied to this MDP, the aim of the agents is to learn a policy π to maximise the future discounted reward defined by:

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \tag{3}$$

Where $\gamma \in [0, 1]$.

Q-learning, an off-policy model-free value-based RL algorithm is an effective and powerful tool in solving MDPs and has been shown to find an optimal policy (one which maximises expected total discounted reward) in any finite MDP [24]. This approach aims to learn an optimal action-value (Q) function $Q^*(s, a)$ given a state s and action a: when optimal policy π^* is followed.

$$Q^*(s, a) = \mathbb{E}[r|s, a] + \gamma \sum_{s'} \mathbb{P}(s'|s, a) \max_{a'} Q^*(s', a') \tag{4}$$

Q-learning, in this format, takes the form of a table-based algorithm which recursively approximates $Q^*(s, a)$ through iterative Bellman updates with a learning rate of α and temporal difference target of y_t for the Q-function:

$$\begin{aligned} Q^*(s_{t+1}, a_{t+1}) &\leftarrow Q^\pi(s_t, a_t) + \alpha(y_t - Q(s_t, a_t)) \\ y_t &= R_t + \gamma \max_{\alpha_{t+1}} Q^\pi(s_{t+1}, a_{t+1}) \end{aligned} \tag{5}$$

A major improvement to Q-learning performance came from using a convolutional neural network for the Q-value estimator combined with a novel experience replay mechanism and an iterative periodic update process which allowed the Deep Q-Network (DQN) agent to converge on an optimal policy when tested on the Atari 2600 dataset [25].

4. Reward Functions

The following functions are experimentally reviewed. We review reward functions from the literature (1, 3, 4, 6, 8, 11) and propose some functions here (2, 3, 5, 7, 9, 12), inspired by the previously proposed algorithms. We define V_t as the set of vehicles in incoming lanes and m_v as as the speed of vehicle v. Furthermore, we define τ^v as the waiting time of vehicles. Similar to the definition of upstream traffic V_t, we define pressure as P_x where $x \in \{up, down\}$, and $\{up, down\}$ representing the upstream and downstream traffic flows respectively.

1. **Average Speed:** Used in [26], we aim for the agent to maximise the flow of vehicles by reducing the amount of time stopped or at low speeds. This is the optimal solution proposed by [17].

$$r_t = \frac{1}{|V_t|} \sum_{v \in V_t} m_v \tag{6}$$

2. **Average Speed Normalised:** By normalising the average speed with the maximum observed speed in a lane m_{max} (defined as $\max_{V_t}(m_v)$), we aim to reduce any problems caused by different speed limits in the approaches to the junction.

$$r_t = \frac{1}{|V_t|} \sum_{v \in V_t} \frac{m_v}{m_{max}} \tag{7}$$

3. **Maximum Wait Time:** This approach prioritises the vehicles which have been waiting the longest.

$$r_t = -\max_{\{v \in V_t\}} \tau_t \tag{8}$$

4. **Aggregate Wait Time:** As suggested by [27] ,the reward is the negative sum of the wait time of all the queuing cars.
$$r_t = -\sum_{v \in V_t} \tau_t \tag{9}$$

5. **Aggregate Wait Time Normalised:** Similar to Aggregate Wait Time, but we use the maximum waiting time to normalise the value. This is so the agent is not forced into acting in a first in, first out manner which may happen with just using Aggregate Wait Time.
$$r_t = -\sum_{v \in V_t} \frac{\tau_t}{\tau_{max}} \tag{10}$$

6. **Pressure:** Used in [4, 5, 28, 29, 19], pressure is a very common reward function, and is defined as the difference of vehicle density in the upstream lanes P_{up} and downstream lanes P_{down}. This approach has been promising in simulations which use synthetic or grid based city layouts, and has been shown to synchronise the green phases of the main roads [30].
$$r_t = -P_i = -(P_{up}) - (P_{down}) \quad \text{Where} \tag{11}$$

7. **Pressure Squared:** Following on from pressure, we implemented pressure squared to test if penalising actions which lead to increased pressure is an effective approach to the reward function.
$$r_t = -(P_i)^2 \tag{12}$$

8. **Queue:** This reward function is trivial to calculate and implement in the real world, and is used in some VA implementations. In addition, it is one of the most common reward functions used in implementations, as seen in [18].
$$r_t = -|V_t| \tag{13}$$

9. **Queue Squared:** This reward function further penalises the actions which lead to larger queue. This was included due to the multi-agent scenario, as reducing the amount of queuing cars could increase the predictability of the traffic flow outbound from an intersection.
$$r_t = -(|V_t|)^2 \tag{14}$$

10. **Maximum Wait Aggregated Queue (MWAQ):** In this reward function, we use the value for the maximum waiting time multiplied with the length of the queue to approximate the worst case aggregate time waited for all the cars. This approach is a modification of the approach used by Ma et al. [14].
$$r_t = -(\max_{\{v \in V_t\}} \tau_t * \sum_{n \in N} q_t) \tag{15}$$

11. **Neighbourhood Adjusted Maximum Wait (NAMW):** In this approach, we include basic information (number of vehicles) from a neighbouring intersection, as demonstrated in [31]. This may pose some implementational problems in the real world use due to the changing nature of traffic networks. However, this information is collectable via the most common type of sensor used in UK roads, induction loop sensors, which are low-cost and

effective. In addition, it is possible to retrofit these sensors into existing infrastructure [32].

$$r_t = -(\max_{\{v \in V_t\}} \tau_v + \gamma \max_{\{v \in V_{tn}\}} \tau_v) \quad \text{Where } V_{tn} \text{ is the vehicles at neighbour intersections} \tag{16}$$

In the definition of NAMW, we include an additional discount factor γ. This value is applied to the information from the neighbouring intersections, to ensure that the component of this function which has the greatest impact on the overall value is the component from the agent in question.

12. **MOVA (referred to as VA in our results):** As a benchmark, we used an implementation of one of the most commonly found TSC algorithms in the UK, MOVA [1].

5. Experimental Setup

We used the RESCO benchmarking environment as introduced by Ault et al. [6], based on the Simulation of Urban MObility (SUMO) simulator. Included in RESCO is the Ingolstadt environment [33], a demand-calibrated scenario for SUMO. The traffic network and traffic demand were set up as described in the Ingolstadt scenario [33].

We chose to use Deep Q-Learning for all of our agents as Deep Q-Learning is commonly used within the literature [18, 34]. Furthermore, it was found by Genders et al. that the agent is not sensitive to the state representation [35], and so in our experiments we chose to use the state representation provided by [20]. This state definition at an intersection includes number of vehicles in each incoming lane, the speed of the incoming vehicles, the queue length and the total waiting time of the vehicles at that intersection. The DQN used was implemented in PyTorch, and included a convolutional layer, followed by two fully connected layers of 32 neurons. The parameters for the DQN were set as in [20].

Each reward function was repeated with $n = 20$, with the total waiting time calculated for each run, and the average of this cumulative waiting time was used to evaluate the functions. Moreover, in our initial experiments, we found that the traffic scenario did not include enough vehicles to saturate the road network, and definitively test the reward functions. In order to resolve this, we chose to modify the traffic scale option within SUMO. This option, which is set to 1 by default, proportionally increases the traffic by that percentage. We set it to 1.5, meaning that each car in the network had a 50% chance of being duplicated. We chose this instead of generating random data as it would still maintain the flow of traffic which is seen in the Ingolstadt dataset. We chose a scale of 1.5 as it was a compromise, due to a quirk in how SUMO processed uncompleted journeys at the end of the simulation. If cars do not arrive at their destination by the simulation end time, they are not included in the output data, leading to misleading information as worse-performing agents outperform those which can (despite long delays) allow a greater throughput of vehicles.

6. Results

Figures 1 and 2 contain box plots of our results for traffic scale factors of 1 and 1.5, respectively. Tables 1 and 2 contains the tabular waiting time results for the traffic scale factor of 1 and 1.5

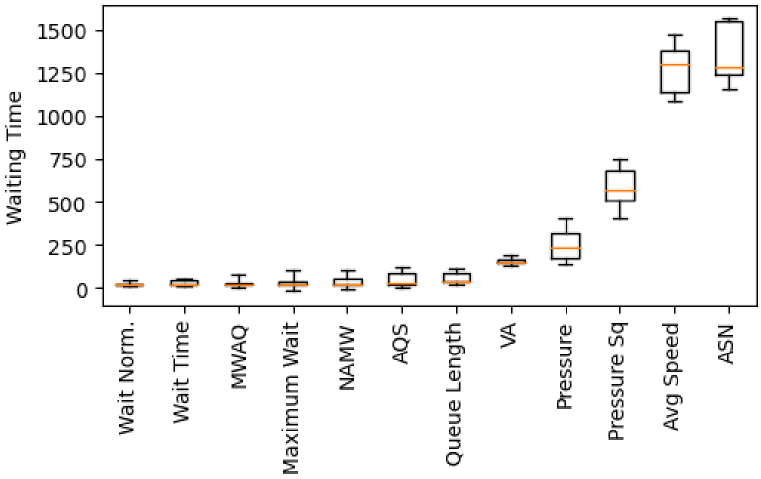

Figure 1: Average time spent waiting for each vehicle with traffic scale of 1

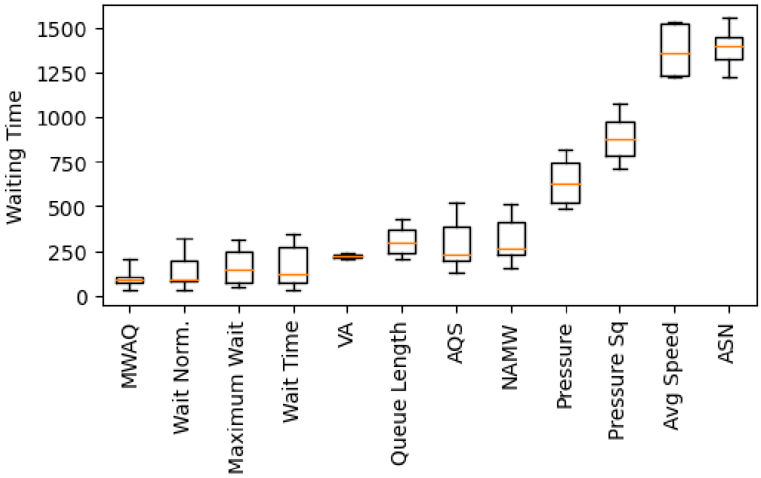

Figure 2: Average time spent waiting for each vehicle with traffic scale of 1.5

respectively.

Our initial run with the default traffic flow found that pressure (pressure and pressure squared) and average speed (average speed and average speed normalised) based methods were the only methods to not outperform the MOVA/VA benchmark when the traffic scale factor was set to 1 and the traffic did not saturate (or near-saturate) the network. Whilst no conclusions can be made between the RL algorithms in this scenario, 7 of the reward functions used outperformed the benchmark, showing that there is significant potential in the use of RL for TSC. Furthermore, we note that the average speed and ASN reward functions performed significantly worse than in [17] when used in a multi-agent scenario. We speculate that this is caused by the problem of non-stationarity as agents could struggle to differentiate what is causing their low reward results when a signal upstream is essentially controlling the flow of traffic into that junction. Furthermore, junctions which see few vehicles passing through are likely to be impacted more by the changing policy of upstream intersections, a factor which could penalise the average speed functions moreso.

In addition, the pressure based reward functions did not outperform the benchmark either. We hypothesise that this is in part caused by the structure of the road layout, and the type of road layout which was used to develop these algorithms. Whilst these algorithms may perform well in arterial road layouts [4] and grid based layouts, they may struggle when faced with other road networks. It's important to note that this kind of road layout is rare in Europe, which is where the data originates from.

Once the traffic scale was increased to 1.5, a greater divergence is seen between the functions used. We see the trend of average speed and pressure being outperformed by the baseline. It is also important to note the reduced variance in the VA baseline. In real world deployments, this may be important as it increases the predictability of the algorithm.

Whilst the MWAQ was only slightly better than the other algorithms at the higher traffic scale, the reduced variance in the runs mean that in almost every run, it outperformed the other algorithms. We believe that this improvement is due to the fact that when only queue metrics are used, there is chance that the priority will almost certainly be given to the direction which has the greatest flow of traffic, leaving some cars to wait for a significant amount of time as they are travelling perpendicular to the flow of traffic. When the maximum wait time is included, the agent is less likely to prioritise the flow of traffic as often.

	Waiting Time					***Trip Duration***				
Function	**Mean**	**Median**	q_1	q_3	**Standard Error**	**Mean**	**Median**	q_1	q_3	**Standard Error**
Wait Norm.	28.791	21.491	19.819	23.655	4.543	207.819	199.417	197.287	201.661	5.33
Wait Time	33.643	20.249	19.342	42.941	5.567	212.394	197.618	196.397	222.908	6.24
MWAQ	40.887	20.754	19.745	30.783	9.351	220.773	198.447	197.03	207.584	10.789
Maximum Wait	45.391	20.506	19.417	40.467	14.139	225.214	198.029	196.553	217.523	15.334
NAMW	50.480	23.823	20.980	56.555	12.167	235.558	203.446	199.835	248.108	14.34
AQS	63.187	26.455	22.122	91.080	14.334	243.592	202.958	198.775	280.628	15.04
Queue Length	66.559	42.146	35.005	91.689	11.423	248.604	221.337	213.839	273.386	12.899
VA	163.024	153.495	146.243	168.025	6.134	373.013	365.53	355.626	381.188	6.36
Pressure	274.838	239.822	176.647	322.013	30.024	467.834	430.757	368.948	516.469	30.188
Pressure Sq	582.607	570.367	507.248	680.664	39.542	771.338	769.859	681.624	874.386	38.275
Avg Speed	1282.135	1301.641	1138.004	1382.999	43.859	1435.277	1452.359	1289.616	1522.031	40.924
ASN	1359.657	1281.806	1244.946	1548.654	47.114	1507.227	1429.812	1383.023	1684.1	44.252

Table 1

Results from Ingolstadt dataset with a traffic scale factor of 1. All measurements are in seconds, and sorted by the mean.

	Waiting Time					***Trip Duration***				
Function	**Mean**	**Median**	q_1	q_3	**Standard Error**	**Mean**	**Median**	q_1	q_3	**Standard Error**
MWAQ	117.634	90.464	72.807	107.991	22.19	322.803	297.896	282.083	310.232	21.442
Wait Norm.	174.404	90.856	79.898	200.412	36.587	377.971	298.703	283.512	411.616	36.413
Maximum Wait	182.354	148.895	75.359	249.336	33.884	386.453	356.629	283.238	463.342	32.84
Wait Time	185.514	118.674	74.394	270.065	36.146	390.144	321.725	279.735	486.621	34.771
VA	219.628	217.749	213.689	227.69	3.683	466.817	465.58	457.547	477.21	5.264
Queue Length	320.364	294.727	236.463	373.412	25.802	522.955	502.564	446.207	568.677	23.689
AQS	328.802	228.553	196.05	390.11	45.808	525.009	422.639	400.687	593.93	42.878
NAMW	337.1	261.962	225.949	408.744	42.093	557.431	480.261	451.02	629.928	39.652
Pressure	652.319	630.52	521.041	748.133	38.259	836.341	819.337	706.234	922.415	34.719
Pressure Sq	891.48	875.225	784.9	978.235	41.964	1064.347	1045.587	968.079	1127.347	38.403
Avg Speed	1380.317	1360.389	1232.442	1528.785	38.149	1530.289	1524.673	1389.29	1669.204	35.42
ASN	1392.806	1400.591	1329.833	1450.066	43.31	1538.26	1541.388	1486.369	1587.584	39.787

Table 2

Results from Ingolstadt dataset with a traffic scale factor of 1.5. All measurements are in seconds, and sorted by the mean.

7. Conclusion

In this paper, we discuss the non-stationary problem and how it may impact the use of RL for the TSC problem. We evaluate 11 different reward functions, including some of the more commonly used examples, and compare them to a benchmark of a real world function through simulations on a calibrated dataset from Ingolstadt. We believe that one potential avenue of further work is to conduct these experiments on a larger scenario, which would allow for further validation of the optimal reward function for use in the TSC problem. There may also be benefits to an ensemble approach to the TSC, where multiple agents with different reward functions are used to come to a conclusion on the optimal decision.

Moreover, an approach which could be explored is to employ pretraining on new agents, training each agent on an individual intersection with the same number of lanes as the one they will control before being implemented in the network with other agents. Whilst this may increase the time required to train an agent, it may allow all agents to converge on a solution sooner.

Additionally, there could be a focus on the environmental impacts of using one reward function over another. For example, HGVs emit significantly more emissions when they accelerate compared to private vehicles. Therefore, an algorithm which does not differentiate between these types of vehicles will not prioritise this (or the impact on traffic once the HGV slows down, and the corresponding environmental impact of this), and therefore cause more damage to the environment.

Acknowledgements

Behrad Koohy is supported by an ICASE studentship funded by the Engineering and Physical Sciences Research Council (EPSRC) and Yunex Traffic. Enrico Gerding and Sebastian Stein are funded by the EPSRC AutoTrust platform grant (EP/R029563/1). Sebastian Stein is also supported by an EPSRC Turing AI Acceleration Fellowship on Citizen-Centric AI Systems (EP/V022067/1).

References

[1] R. A. Vincent, J. R. Peirce, MOVA: Traffic responsive, self-optimising signal control for isolated intersections, TRRL Research Report (1988).

[2] P. B. Hunt, D. I. Robertson, R. D. Bretherton, M. Royle, The scoot on-line traffic signal optimisation technique, Traffic engineering and control 23 (1982).

[3] A. Cabrejas-Egea, R. Zhang, N. Walton, Reinforcement learning for traffic signal control: comparison with commercial systems, Transportation research procedia 58 (2021) 638–645.

[4] H. Wei, C. Chen, G. Zheng, K. Wu, V. Gayah, K. Xu, Z. Li, 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, 2019, pp. 1290–1298.

[5] P. Varaiya, Max pressure control of a network of signalized intersections, Transportation Research Part C: Emerging Technologies 36 (2013) 177–195.

[6] J. Ault, G. Sharon, Reinforcement learning benchmarks for traffic signal control, in: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2021.

[7] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, Z. Li, Colight: Learning network-level cooperation for traffic signal control, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1913–1922.

[8] R. Bellman, Dynamic programming, Science 153 (1966) 34–37.

[9] W. C. Cheung, D. Simchi-Levi, R. Zhu, Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism, in: International Conference on Machine Learning, PMLR, 2020, pp. 1843–1854.

[10] V. Lomonaco, K. Desai, E. Culurciello, D. Maltoni, Continual reinforcement learning in 3d non-stationary environments, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 248–249.

[11] A. Nareyek, Choosing search heuristics by non-stationary reinforcement learning, in: Metaheuristics: Computer decision-making, Springer, 2003, pp. 523–544.

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

[13] H. R. Berenji, D. Vengerov, A convergent actor-critic-based frl algorithm with application to power management of wireless transmitters, IEEE Transactions on Fuzzy Systems 11 (2003) 478–485.

[14] J. Ma, F. Wu, Feudal multi-agent deep reinforcement learning for traffic signal control, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '20, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2020, p. 816–824.

[15] J. N. Foerster, Y. M. Assael, N. de Freitas, S. Whiteson, Learning to communicate to solve riddles with deep distributed recurrent q-networks, arXiv preprint arXiv:1602.02672 (2016).

[16] S. Sukhbaatar, R. Fergus, et al., Learning multiagent communication with backpropagation, Advances in neural information processing systems 29 (2016).

[17] A. C. Egea, S. Howell, M. Knutins, C. Connaughton, Assessment of reward functions

for reinforcement learning traffic signal control under real-world limitations, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2020, pp. 965–972.

[18] H. Wei, G. Zheng, V. Gayah, Z. Li, A survey on traffic signal control methods, AAMAS 2019 (2019).

[19] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, Zhenhui, Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control, in: AAAI, 2020.

[20] J. Ault, J. P. Hanna, G. Sharon, Learning an interpretable traffic signal control policy, arXiv preprint arXiv:1912.11023 (2019).

[21] R. Bellman, A markovian decision process, Journal of mathematics and mechanics (1957) 679–684.

[22] Department for Transport, Traffic signs manual, 2020. URL: https://www.gov.uk/government/publications/traffic-signs-manual.

[23] P. Sun, N. AlJeri, A. Boukerche, A fast vehicular traffic flow prediction scheme based on fourier and wavelet analysis, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.

[24] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.

[25] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, Journal of Artificial Intelligence Research 47 (2013) 253–279.

[26] E. van der Pol, F. A. Oliehoek, Coordinated deep reinforcement learners for traffic light control, in: Coordinated Deep Reinforcement Learners for Traffic Light Control, 2016.

[27] T. Chu, J. Wang, L. Codecà, Z. Li, Multi-agent deep reinforcement learning for large-scale traffic signal control, IEEE Transactions on Intelligent Transportation Systems 21 (2019) 1086–1095.

[28] Q. Wu, L. Zhang, J. Shen, L. Lü, B. Du, J. Wu, Efficient pressure: Improving efficiency for signalized intersections, arXiv preprint arXiv:2112.02336 (2021).

[29] N. Rouphail, A. Tarko, J. Li, Traffic flow at signalized intersections (1992).

[30] R. Roess, E. Prassas, W. McShane, Traffic engineering, 4th ed., Prentice Hall, 2011. Includes bibliographical references and index.

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

[32] G. Leduc, et al., Road traffic data: Collection methods and applications, Working Papers on Energy, Transport and Climate Change 1 (2008) 1–55.

[33] S. C. Lobo, S. Neumeier, E. M. Fernandez, C. Facchi, InTAS–the ingolstadt traffic scenario for SUMO, arXiv preprint arXiv:2011.11995 (2020).

[34] D. Zhao, Y. Dai, Z. Zhang, Computational intelligence in urban traffic signal control: A survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (2011) 485–494.

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