Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 14936–14941

Control of traffic light timing using decentralized deep reinforcement learning [⋆]

Harshal Maske [*] Tianshu Chu [**] Uroš Kalabić [*]

[*] *Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA (e-mail: kalabic@merl.com)*
[**] *Uhana Inc., Palo Alto, CA 94306, USA (e-mail: tchu@uhana.io)*

Abstract: In this work, we introduce a scalable, decentralized deep reinforcement learning (DRL) scheme for controlling traffic signalization. The work builds on previous results using multi-agent DRL, with a new state representation and reward definitions. The state representation is a coarse image of traffic and the definitions of reward functions are tested based on the simulated Monaco SUMO Traffic (MoST) scenario. Based on extensive numerical experimentation, we have found the most appropriate choice of the reward function is related to minimizing the average amount of time vehicles spent in the network, but with various modifications that improve the learning process. The resulting algorithm performs better than the previous one on which it is based and markedly better than a non-learning based, greedy policy.

Keywords: Intelligent Transportation; traffic control systems; Automatic control, optimization, real-time operations in transportation; reinforcement learning control; integrated traffic management.

1. INTRODUCTION

The impact of vehicular traffic congestion is a societal and environmental concern throughout the world [Reed and Kidd, 2019] and it is recognized that alleviation of these concerns can be achieved by controlling traffic. The passive control of traffic is done using traffic signalization, i.e., using traffic signals, lights. Adaptive traffic signal control (ATSC) is the dynamic control of traffic signals with the aim of reducing congestion and improving traffic throughput. State-of-the-art ATSC systems applied to city traffic include the broadly deployed SCOOT and SCATS [Wei et al., 2020], which were developed long before the advent of current machine-learning-based techniques. It is therefore clear that such techniques could provide an improvement to their functions, at the very least modifying them if not replacing them outright. Application of learning-based control to traffic signalization has received attention in research for a few reasons, a main one being that, unlike traditional methods, learning does not necessarily rely on mathematical modelling, which is beneficial in application to complex systems lacking a physics-based model.

In this work we consider the application of multi-agent deep reinforcement learning (deep RL or DRL) to ATSC, the unique aspect of which is the state description, which uses an image of traffic or traffic flow. For representing the state, we use an image of traffic in the incoming lanes of a traffic signal; the image consists of a coarse grid, where each grid point represents the fraction of area in that grid element that is occupied by a vehicle. This image-like state representation is able to provide rich traffic information, and it can be obtained through multiple means, such as GPS data or directly from cameras. Each agent controls one traffic signal and is able to receive data from neighboring agents to compute its reward function. Another practical challenge of RL is to define an appropriate reward function in training to improve the robustness and stability of RL models. In this work, the reward function is designed to minimize delay, with refinements to avoid undesirable behaviors.

In this work, we consider four different reward function definitions, each one being a modification of the previous. The final reward function has a few modifications to the baseline reward function, which is a sum of all vehicles' delays. The first modification is that we take a sum of weighted exponentials of delays, to avoid the situation where a few drivers are greatly penalized for the benefit of the average. The weights are inversely proportional to the distance from the intersection and the second modification we make is to bound the weights from below so that the delay of distant vehicles is taken into account. The final modification we make is to bound the reward so that the controller avoids attempting to resolve hopeless, highly congested situations. Our design choices are supported in simulation of the Monaco SUMO traffic (MoST) simulation environment of Codecá and Härri [2018], the results of which we present in this paper.

To compare our work to the literature, we being by noting that DRL-based approaches to ATSC typically build on the work of Mnih et al. [2016], which uses DNNs for both policy and value approximation. One of the first works

[⋆] This work was supported by Mitsubishi Electric Research Laboratories. At the time of submission, H. Maske was an intern at Mitsubishi Electric Research Laboratories; he is now with Ford Motor Company (e-mail: hmaske1@ford.com).

10.1016/j.ifacol.2020.12.1980

has been that of Li et al. [2016], which considered a simplified, isolated traffic intersection where the vehicles are not allowed to make turns and the reward is defined using the queue lengths in the incoming lanes. Other works include that of Chu et al. [2016] and Casas [2017], which verified the superior approximation capabilities of deep Q-learning and deep deterministic policy gradient in a simplified traffic environment, respectively. Furthermore, the work of van der Pol [2016] applied deep independent Q-learning, addressing issues of partial observability using transfer planning, but the states therein were infeasible and the simulated traffic environments were oversimplified. For reward definition, the authors used a weighted function of factors such as waiting time, delay, emergency stops and frequency of traffic light switches and experimentally determined that delay and waiting time should be highly weighted in the reward function. Genders and Razavi [2018] sought to answer key questions about data and sensor requirements with the goal of bringing reinforcement learning to real-world ATSC application in the near future. Their work considered three different definitions of state, varying resolution of information from low to high, where the highest resolution corresponded to the discretization of each incoming lane into cells of fixed length; the difference from our work is that these cells were binary-encoded, representing presence or absence of a vehicle. Traffic-grid state representations have also been considered elsewhere in the literature under alternative nomenclature. For example, Genders and Razavi [2018] term it discrete cell encoding, van der Pol [2016] as a position matrix, and Thorpe and Anderson [1996] as a fixed-distance representation.

In the literature cited above, it is not clear how to scale DRL algorithms to large-scale application to traffic; most studies conduct experiments in either isolated intersections or small traffic networks. Furthermore, there is no agreement in the literature on an appropriate definition of reward function. The scalability issue of applying DRL to ATSC has only been considered recently, in the work of Chu et al. [2019], which used a multi-agent and, importantly, decentralized DRL approach to optimize traffic delay in the same Monaco environment that we consider. However, that work utilized road sensor collected states, which provide limited information regarding the traffic state thereby increasing learning difficulty. In this paper, we explore a more generic state and reward representation while achieving similar or better performance.

The rest of the paper is structured as follows. Section 2 presents the DRL algorithm design. Section 3 presents numerical results in the MoST environment. Section 4 is the conclusion.

2. DRL DESIGN

The scheme used in this work is based on the work of Chu et al. [2019], which developed a multi-agent approach using the advantage-actor-critic (A2C) algorithm for deep reinforcement learning. In this multi-agent A2C (MA2C) approach, each RL agent controls a single traffic light and shares its policies and discounted local states to neighboring agents. In our work, we have modified the state input as an image-like representation of traffic in a neighborhood for which the agent is responsible. We refer to this method as regional-grid A2C (RGA2C).

In the following, we describe the representation of the state, description of actions, and definition of the reward function.

2.1 *State representation*

Every agent is responsible for a region of traffic and, for this reason, we define the state to consist of images of traffic. The images are represented by a coarse grid of pixels, with each pixel representing the percentage of the grid area occupied by a vehicle. This representation was introduced in our previous work [Maske et al., 2019], where we argued that states representing images are natural for use as states in a machine-learning framework. In this work, the width of the pixels was chosen to equal the width of the lane and their length was fixed at 2m.

Each agent observes the images of the incoming lanes to the traffic signal it controls as well as the images of incoming lanes of neighboring agents' signals. The images are represented by a stacked vector X_t, whose elements are pixel values. The state is then set to be,

$$s_t = \begin{bmatrix} X_{t-2} \\ X_{t-1} \\ X_t \\ f_t \end{bmatrix}, \tag{1}$$

which consists of the three most recently received images, combined with f_t, which is a "fingerprint" of the policy of the neighboring agents, consisting of parameters that identify neighboring agents' policies. Note that a fingerprint carries important information in decentralized RL that reduces the nonstationarity of system transition from the viewpoint of each local agent.

This state description is a significant deviation from the state descriptions found in the literature dealing with control of traffic, but similar to that used by RL algorithms designed to play video games such as, for example, by Vinyals et al. [2017]. The use of s_t as a state representation is a generalization of ordinarily-used state representations since they are typically defined using position and velocity, and position can be reconstructed from the current snapshot X_t and velocity can be reconstructed, albeit indirectly, from the approximation of a velocity snapshot $(X_t - X_{t-1})/2$. We assume the reconstruction would be performed through training of the deep neural network used in the DRL scheme. Note that the use of the the third-most recent snapshot gives us approximate information about the second derivative of the system.

2.2 *Action description*

The action of each agent corresponds to a light phase for each intersection. Since phases vary from intersection to intersection, the list of available choices of action are defined according to the phases available at the particular intersection controlled by the agent.

To enable more flexible and direct control by RL agents, we use the phase definition from Prashanth and Bhatnagar [2010], according to which each action is a possible phase given by red-green combinations of traffic lights at that intersection. Specifically, the agent chooses an action from a set of all feasible phases, where each action is implemented for a duration of $\Delta t = 5$s. In the case of a change between phases, the period Δt begins with a necessary transitional, yellow-light phase, which is a duration of 2s.

2.3 Reward function definition

A fair aim of traffic control is to minimize the cumulative time spent in traffic by participants in the system [Nilsson, 2019]. To do this, we consider rewarding reduced trip time for every vehicle in the traffic network. The definition we use is related to that which we considered previously, in which a speed below the free-flow speed was penalized [Maske et al., 2019]. This is done by defining a delay function,

$$d_i = \max\{0, 1 - v_i/v_{f,i}\}, \tag{2}$$

where v_i is the speed of a vehicle i and $v_{f,i}$ is the free-flow speed of the lane occupied by the same vehicle. The delay is a ratio varying between 0 and 1, with 0 corresponding to the situation where free-flow velocity has been reached and 1 corresponding to the situation where the vehicle is stopped.

Note that penalizing delay as opposed to queuing time introduces granularity in the definition and helps the RL algorithm to proactively avoid congestion as delay shows a slow-down in traffic sooner than the beginning of queuing at a traffic light. Furthermore, a delay of 1 is directly implied when a vehicle enters a queue and, therefore, delays can represent queuing; the reverse, however, is not the case.

For each incoming lane, we fix the free-flow speed to $v_{f,i} \equiv v_f = 10$m/s, which is a reasonable speed in a city's core.

For each agent j, we develop and investigate four different reward functions based on learning performance in a realistic traffic system. Formally,

$$r_j = \hat{r}_j^{(A)} + \sum_{k \in \mathcal{N}_j} \exp(-L_k) r_k, \tag{3}$$

where $A = L, E, W,$ or B. Similarly to Chu et al. [2019], we add to the reward an exponentially discounted sum of rewards of all neighboring agents, with $\mathcal{N}_j$ being the index of neighboring agents to agent j and L_k being the distance to the neighboring intersection $k \in \mathcal{N}_j$.

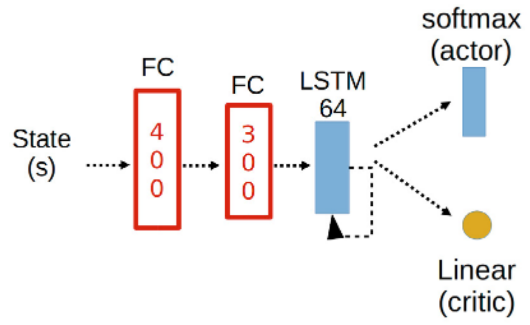

Fig. 1. DNN architecture

The reward $r_j^{(L)}$ is a sum of delays for all vehicles,

$$r_j^{(L)} = -\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} d_i, \tag{4a}$$

where $\mathcal{V}$ is the set of indexes of all vehicles in incoming lanes.

The reward $r_j^{(E)}$ is an exponentially weighted sum of delays for all vehicles,

$$r_j^{(E)} = -\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} C^{w_i d_i} - 1, \tag{4b}$$

where w_i is the weight given to the delay of vehicle $i \in \mathcal{V}$, and $C > 0$ is a weighting constant. This reward function corresponds to the sum of all delays in the region with the exponentiation of C being introduced to penalize higher delays more than low delays, i.e., avoid situations where delays of a few are increased greatly for the small benefit of the majority. The weights w_i are determined according to,

$$w_i = 1 - D_i/L_i,$$

where D_i is the distance of the vehicle away from the intersection and L_i is equal to the length of the incoming road on which the vehicle is traveling.

The reward $r_j^{(W)}$ is equal to $r_j^{(E)}$ with weights w_i lower-bounded by 0.5, i.e.,

$$r_j^{(W)} = r_j^{(E)}, \tag{4c}$$

where,

$$w_i = \max\{1 - D_i/L_i, 0.5\}.$$

In this way, we penalize vehicles more than half the road length away equally and closer vehicles, i.e., vehicles more likely to be affected by changes in traffic signal phase, proportionally to their proximity to the intersection. This choice has been informed by experimentation showing that far-off vehicles do not change the reward significantly without lower-bounding the weighting of their delay, as these vehicles often travel at free-flow speed.

Finally, the reward $r_j^{(B)}$ is equal to an upper-bounded $r_j^{(W)}$, i.e.,

$$r_j^{(B)} = \max\left\{r_j^{(W)}, -c_{\max}\right\}. \tag{4d}$$

This is done to avoid suboptimal minima during optimization; we lower-bound r_j by $-c_{\max} = -20$. In this way, we avoid attempting to resolve highly-congested situations where more than approximately $c_{\max}$ vehicles are queued near a light.

2.4 Deep neural network architecture

Although it is possible to model traffic using various techniques [Garavello et al., 2016], traffic flow is a complex spatio-temporal process, implying that using only the currently available data in RL training results in an non-stationary Markov decision process. An alternative is to use all historical states as an input to the RL algorithm, but this increases the state-space dimension to the point of prohibiting practical application. To avoid the curse of dimensionality, we follow Chu et al. [2019] to implement a long-short term memory (LSTM) layer as the last layer in a deep neural network (DNN) architecture. As compared to that work, we use a deeper network to better parametrize the larger amount of data contained in the image. The DNN architecture is presented in Fig. 1, with two fully-connected layers of 400 and 300 neurons each, followed by an LSTM layer. The output layer is a softmax layer for the actor and a linear layer for the critic. In training, we use the state-of-the-art orthogonal initializer of Saxe et al. [2014] and RMSprop as the gradient optimizer.

3. SUMO SIMULATION OF MONACO TRAFFIC

In this section, we present the simulation environment in which we evaluate the RGA2C scheme and results from our simulations.

To achieve as realistic a simulation that is available to us, we have implemented our scheme in the Monaco SUMO Traffic (MoST) scenario of Codecá and Härri [2018] subject to time-varying traffic flows. In the simulation, there are 30 signalized intersections in total: 11 are two-phase, 4 are three-phase, 10 are four-phase, 1 is five-phase, and the remaining 4 are six-phase. A map of the simulated environment is shown in Fig. 2, with the above 30 intersections identified. To test the robustness and optimality of algorithms, stochastic, and time-variant traffic flows that mimic actual peak-hour flows in the city of Monaco were recreated in simulation by randomizing the start positions of the vehicles, with quantity determined according to Codecá and Härri [2018].

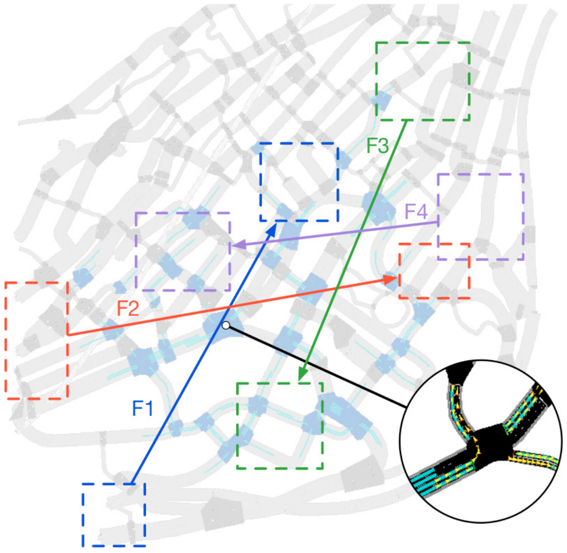

Fig. 2. Map of the simulated environment with the controlled intersections marked in blue [Codecá and Härri, 2018]

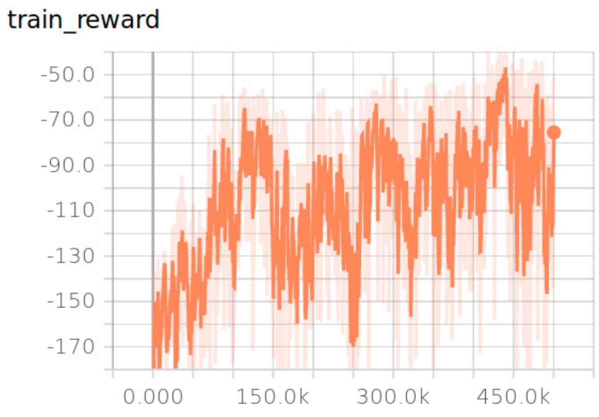

Fig. 3. Learning curve (reward vs. time-steps) corresponding to $r_j = r_j^{(L)}$

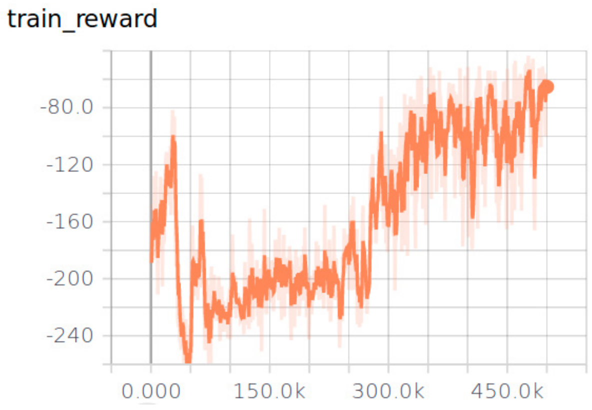

Fig. 4. Learning curve (reward vs. time-steps) corresponding to $r_j = r_j^{(E)}$

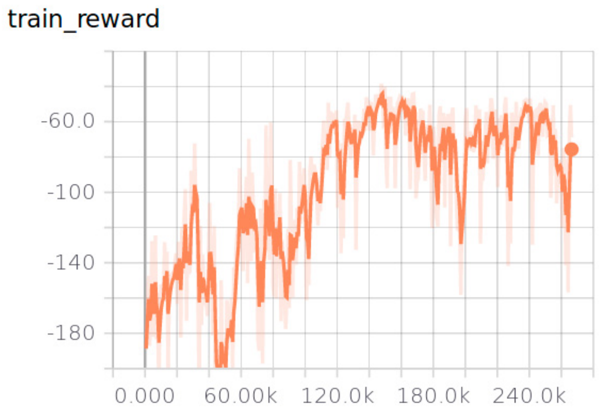

Fig. 5. Learning curve (reward vs. time-steps) corresponding to $r_j = r_j^{(W)}$

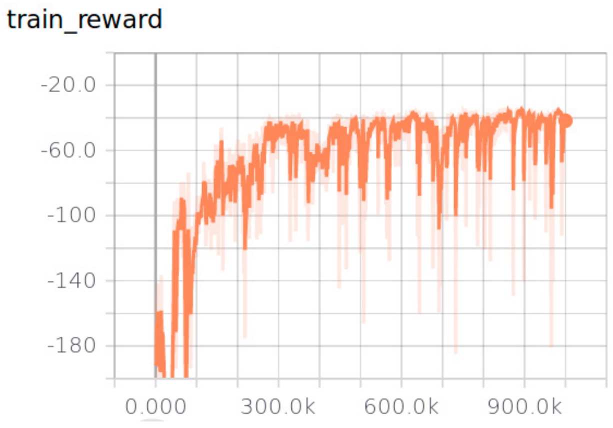

Fig. 6. Learning curve (reward vs. time-steps) corresponding to $r_j = r_j^{(B)}$

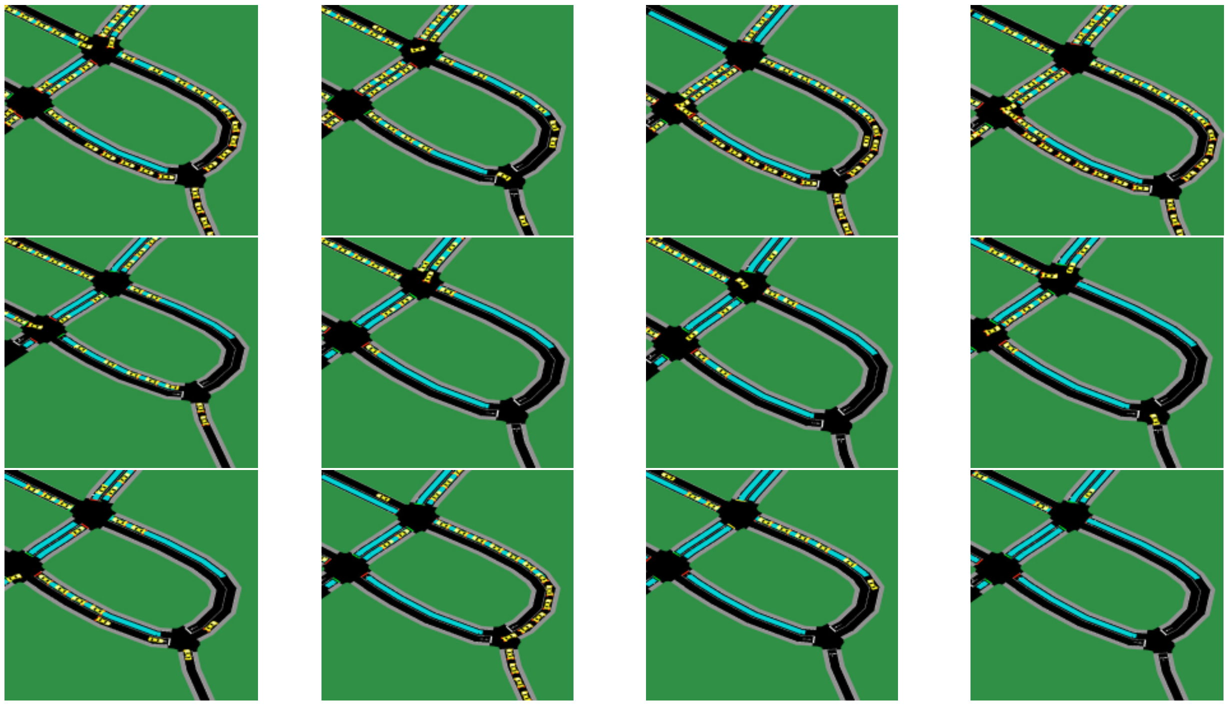

Fig. 7. Traffic distribution at different time-intervals : 1800s, 1950s, 2100s, 2400s, from left to right. Traffic light governed by: Top row: greedy policy, Mid row: MA2C policy, Bottom row: RGA2C policy.

For comparison to the available literature of Chu et al. [2019], the baseline algorithm was set to be a decentralized greedy policy that selects the phase which has the maximum number of vehicles approaching in the first 50m away from the intersection.

3.1 Training

Training was performed over 1 million time-steps, equivalent to about 1388 episodes, as each episode lasts one hour, or 720 time-steps. At the end of each episode, the simulation was reset to initial conditions and the process continued.

Figs. 3-6 show the results of training corresponding to the use of different reward functions (4). In these results, we show a plot of the learning curve, i.e., a plot of average episode reward against the duration of learning in time-steps. In Fig. 3, we show the training progress with the purely linear reward $r_j^{(L)}$ (4a). Since the results do not exhibit convergence, we modified our choice of reward to the exponentially weighted sum $r_j^{(E)}$ (4b), with results shown in Fig. 4. These results provide better convergence but it is slow and remains unsteady. In our data, we noticed that vehicles further from intersections had little impact as their corresponding weighting was low. This is not consistent with the fact that vehicles at neighboring intersections have a relatively high positive weighting. For this reason, we modified the reward function to $r_j^{(W)}$ (4c), with results shown in Fig. 5. The results are an improvement but again do not show steady convergence. At this point, we noticed that the agents would try to improve hopelessly difficult, congested situations. To remedy this, we capped the reward at each intersection at a maximum sum of delay at about 10 vehicles, i.e., we set $c_{\max} = 10C = 20$ in $r_j^{(L)}$ (4d). The final results are provided in Fig. 6. Here, the curve steadily increases as learning progresses, converging to the optimal value after about 500 episodes. This implies that, without warm starting, the process needs to be trained on about 500 hours of traffic congestion before discovering the optimum.

3.2 Results

We tested the performance of our scheme using five scenarios different from those used in training. In Fig. 7, we show a visual comparison of the greedy, MA2C, and RGA2C policies for one particular run of the algorithm.

The performance is evaluated by various means. In Fig. 8, we plot the average length of each incoming lane queue. Here the results show that the RGA2C scheme shows a

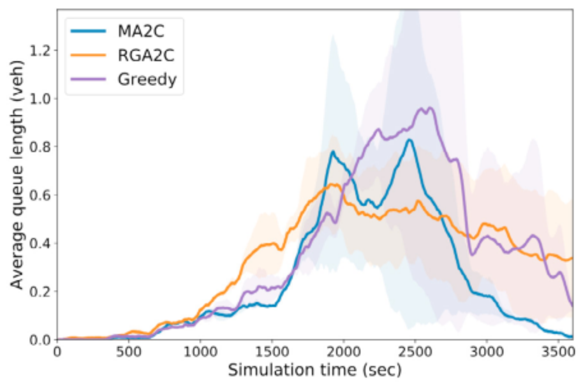

Fig. 8. Average queue length of MA2C (blue), RGA2C (orange), and greedy (purple) algorithms plotted with error bounds (shaded)

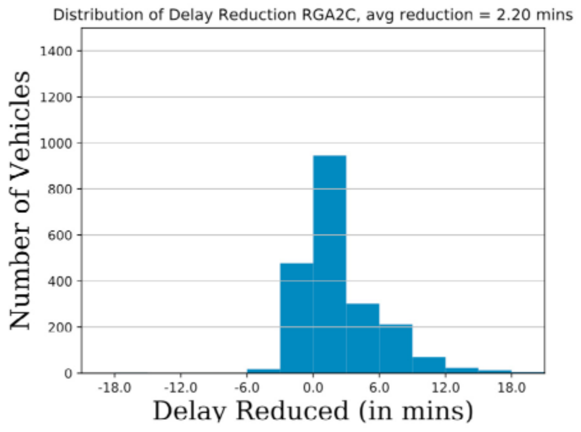

Fig. 9. Comparison of RGA2C and greedy algorithms

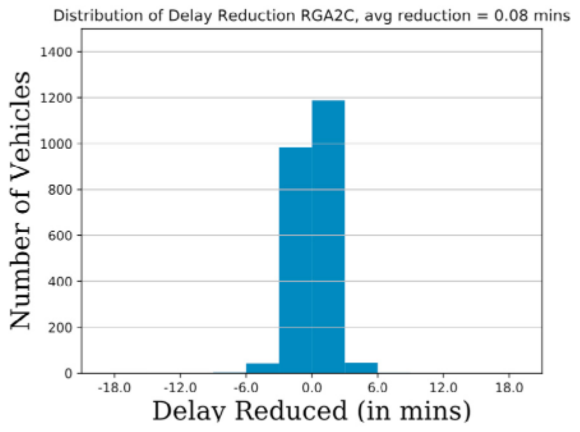

Fig. 10. Comparison of RGA2C and MA2C algorithms

marked improvement over both the baseline MA2C scheme and the greedy control policy. Specifically, the performance is better during peak congestion between 1700s and 3000s and is a little worse before and after the peak. The metric more relevant to our aim is average trip length for each vehicle. In the table below, we show the average trip length, i.e., the amount of time a vehicle spends in the network, and the average wait time of vehicles, i.e., the amount of time a vehicle is fully stopped during its trip.

Algorithm	Trip duration (s)	Wait time (s)
MA2C	161.6	84.6
Greedy	196.0	111.8
RGA2C	152.3	78.7

The RGA2C scheme is superior in both metrics. It achieves about 30% improvement in the wait time when compared to the greedy controller and performs 7% better than the MA2C approach.

In Figs. 9-10, we show the average distributions of delay reduction when using our scheme. In Fig. 9, we compare the RGA2C scheme to the greedy policy. The results show that a small minority are delayed whereas most have their trip times greatly improved. In Fig. 10, we compare the RGA2C and MA2C schemes, showing a small improvement when using the new scheme, but not exhibiting any remarkable deterioration in trip time for vehicles upon implementation of the newer scheme.

4. CONCLUSION

In this work, we tested our multi-agent reinforcement learning scheme for controlling traffic signalization and presented a study of our design of the reward function. The main novelty of our approach is the state representation, which represents an image of traffic, as well as the design of an appropriate reward function in training. The results show improvement over previous work on which this work is based, and we attribute this improvement to the choice of state representation and the design of the reward function.

REFERENCES

Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. `arXiv:1703.09035`.

Chu, T. et al. (2016). Large-scale multi-agent reinforcement learning using image-based state representation. In *Proc. Conf. Decision and Control*, 7592–7597. Las Vegas.

Chu, T. et al. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 21(3), 1086–1095.

Codecá, L. and Härri, J. (2018). Monaco SUMO Traffic (MoST) Scenario: A 3D mobility scenario for cooperative ITS. In *Proc. SUMO Conf.*, 43–55.

Garavello, M. et al. (2016). *Models for Vehicular Traffic on Networks.* AMS, Springfield, MO.

Genders, W. and Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia Comput. Sci.*, 130, 26–33.

Li, L. et al. (2016). Traffic signal timing via deep reinforcement learning. *J. Autom. Sinica*, 3(3), 247–254.

Maske, H. et al. (2019). Large-scale traffic control using autonomous vehicles and decentralized deep reinforcement learning. In *Proc. Intell. Transp. Syst. Conf.*, 3816–3821. Auckland, New Zealand.

Mnih, V. et al. (2016). Asynchronous methods for deep reinforcement learning. In *Proc. Int. Conf. Mach. Learn.*, 1928–1937. New York.

Nilsson, G. (2019). *On robust distributed control of transportation networks.* Ph.D. thesis, Lund Univ.

Prashanth, L.A. and Bhatnagar, S. (2010). Reinforcement learning with function approximation for traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 12(2), 412–421.

Reed, T. and Kidd, J. (2019). INRIX Global Traffic Scorecard. Report, INRIX.

Saxe, A.M. et al. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Int. Conf. Learn. Rep.*, `arXiv:1312.6120`. Banff, AL.

Thorpe, T.L. and Anderson, C.W. (1996). Traffic light control using SARSA with three state representations. Technical report, IBM Corporation.

van der Pol, E. (2016). *Deep reinforcement learning for coordination in traffic light control.* Master's thesis, Univ. Amsterdam.

Vinyals, O. et al. (2017). Starcraft II: A new challenge for reinforcement learning. `arXiv:1708.04782`.

Wei, H. et al. (2020). A survey on traffic signal control methods. `arXiv:1904.08117`.