Traffic Light Management Systems Using Reinforcement Learning

1st Sultan Kübra Can
Department of Computer Engineering
Abdullah Gul University)
Kayseri, Turkey
sultankubra.kilic@agu.edu.tr

2nd Adam Thahir
Department of Computer Engineering
Abdullah Gul University
Kayseri, Turkey
adam.thahir@agu.edu.tr

3rd Mustafa Coşkun
Department of Computer Engineering
Abdullah Gul University
Kayseri, Turkey
mustafa.coskun@agu.edu.tr

4th V. Çağrı Güngör
Department of Computer Engineering
Abdullah Gul University
Kayseri, Turkey
cagri.gungor@agu.edu.tr

2022 Innovations in Intelligent Systems and Applications Conference (ASYU) | 978-1-6654-8894-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASYU56188.2022.9925333

***Abstract*—While reducing traffic congestion and decrease the number of traffic accidents in the intersections, most of the traffic light management approaches cannot adapt well to fast changing traffic dynamics and growing demands of the intersections with modern world developments. To overcome this problem, adaptive traffic controllers are developed, and detectors and sensors are added to systems to enable adoption and dynamism. Recently, reinforcement learning has shown its capability to learn the dynamics of complex environments, such as urban traffic. Although it was studied in single junction systems, one of the problems was the lack of consistency with how the real world system works. Most of the systems assume that the environment is fully observable or actions would be freely executed using simulators. This study aims to merge usefulness of reinforcement learning methods with real-world traffic constraints. Comparative performance evaluations show that the reinforcement learning algorithm (Advantage Actor-Critic (A2C)) converges well while staying stable under changing traffic dynamics.**

***Index Terms*—reinforcement learning, traffic light management**

I. Introduction

Traffic lights have been used in many cities since 19th century with gas-lit traffic lights in London, which was soon followed by a single electrical light in Ohio in 1914 and a network of traffic lights with a manual switch in Utah in 1917 [1]. The purpose of traffic lights is to control traffic to prevent traffic jams and congestion in the intersections and to decrease the number of accidents caused by uncontrolled intersections.

Traffic congestion and jams are practical problems in today's world causing delays, increased costs because of fuel and increased amount of harmful gas emission. As stated in 2021 Urban Mobility Report from Texas A&M, the amount of fuel consumption caused by traffic congestion is nearly 1.7 billion gallons with an annual delay of almost 4.3 billion hours which led to a cost of 100 billion dollars [2]. With the development of modern cities, the traffic lights are automated and optimized to increase the vehicle flow of the intersection using variety of methods as a solution.

This paper proposes a solution based on Reinforcement Learning for this problem. We have designed a traffic environment, trained three agents on selected algorithms, which are Advantage Actor-Critic (A2C), Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO), and evaluated each of the algorithms to show that Reinforcement Learning can optimize the traffic flow in the junctions. Comparative performance evaluations show that A2C algorithm converges well while staying stable under changing traffic dynamics.

The rest of this paper is organized as follows: Section II reviews a variety of methods in the field traffic controllers. Section III presents recent approaches based on reinforcement learning. Section IV presents the simulation environment and the design specifics of our proposed methodology. Section V presents experiment design, performance results and their analysis, while section VI concludes the paper discussing the contributions of our work.

II. Related Work

A. Offline Approaches

Offline or fixed-time traffic light control methods use predefined traffic light settings (the sequence of green, yellow and red phase durations) that are deduced and optimized based on offline historical traffic data analyzation. To maximize its success and optimize the predefined timings, time slots has to be identified. The actuated methods contain a set of defined rules where any of them could be triggered by violations or extreme situations to adapt the traffic environment more. However, even though these kind of methods work well in constant environments, they cannot adapt to the dynamism of the traffic environment and demands.

B. Adaptive Controllers

Adaptive controllers have been utilizing the use of sensors and data collection to adapt the changes in the traffic. An example to these methods is Sydney Coordinated Adaptive

Traffic System (SCATS) [3] where real time data is used to control the traffic lights. However, it is deemed to be costly and requires dramatic amount of modifications to existing road systems. Another notable early traffic system to be used in UK is Split Cycle and Offset Optimisation Technique (SCOOT) [4] where traffic lights are controlled only by small adjustments to avoid dramatic changes and focus more on long-term flow of the intersection using real time data by detectors. One of the more recent approaches is Intelligent Traffic Light Controlling algorithm (ITLC) [5]. It uses vehicular ad-hoc networks (VANETs) [6] as the backbone, which requires all the vehicles in network to be equipped with a form of GPS to identify its properties, and aims to decrease the vehicle wait times.

C. Reinforcement Learning Based Approaches

Machine learning, especially reinforcement learning, methods are used to make the agents or algorithm learn what the dynamics are and act based on those insights. As an example to reinforcement learning, Wiering has proposed different variations of reinforcement learning methods to be applied to various traffic environments and created a simulator called the Green Light District (GLD) with the purpose of demonstration [7] which was used in other works [8] for further development. Recently, reinforcement learning gained much attention as Deep Q-Networks (DQN) has shown robust performance on Atari games [9]. One of the similar approaches called Deep Deterministic Policy Gradient (DDPG) [10] has shown success on large continuous action and state spaces.

The methods to be applied at real world traffic environments have to have a robust performance under a wide variety of conditions and has to react fast when a dramatic change happens. As deep reinforcement learning has shown its success, in our approach, we are applying several on-policy reinforcement learning algorithms with real world constraints to achieve robust performance with an ability to adapt safely to real world intersections as the main contribution of this work.

III. Reinforcement Learning

Reinforcement learning is described as a subfield of machine learning that learns the solution by trial and error [11]. The agent, learner, is not been given what actions $(a \in A)$ to take under specific situations, but rather, it has to discover the best action to take by interacting with the given observable environment states $(s \in S)$ and apply the action to the environment. By defining a reward $(r \in R)$ function, the actions of the agent are evaluated that will make the agent to take better actions by learning the optimum policy $(\pi(a|s))$ to maximize its cumulative reward.

Each state that the agent observes is associated with a value $(V(s))$ calculated by a value function to predict the expected rewards under the taken policy, which is used to evaluate how good a given state is. Here, the main goal of the reinforcement learning is to learn policy and value function.

The interaction between the environment and the agent throughout simulation is a sequence of states (observations from the environment), actions that the agent takes and rewards from the agent in time $(t = 1, 2, \ldots, T)$. Throughout this process of learning, the agent uses the method of exploration and exploitation to accumulate the knowledge of environment hence to learn an optimal policy by taking the best actions until the process is terminated. This whole sequence is described by one episode and could be represented as below where state, action, and reward at time step t are S_t, A_t, and R_t respectively and S_T is the termination state:

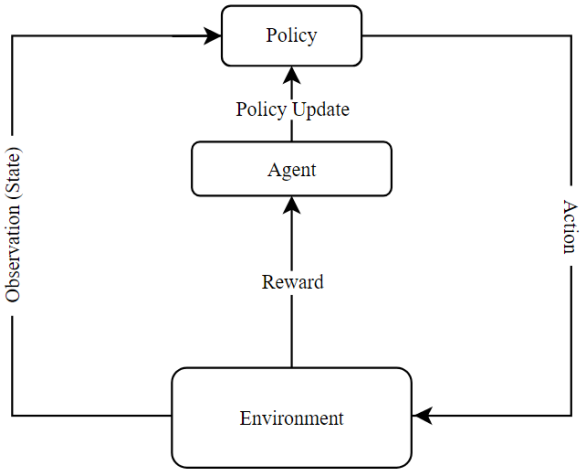

Fig. 1. Reinforcement Learning Diagram

$$S_1, A_1, R_1, S_2, A_2, R_2, \ldots, S_T \tag{1}$$

Policy gradient methods in reinforcement learning learn the policy using a parameterized function respect to θ, $\pi_\theta(a|s)$ without taking value function into consideration. The optimum value of θ could be found by gradient ascent to produce the highest expected reward by maximizing the score function given in (2) where $d^\pi(s)$ is state distribution, $Q^\theta\ (s, a)$ is state-action value and $\pi_\theta(a|s)Q^\pi(s, a)$ is action distribution.

$$J(\theta) = \sum_{sS} d^\pi(s)V^\pi(s) = \sum_{sS} d^{\pi\theta}(s) \sum_{aA} \pi_\theta(a|s)Q^{\pi\theta}(s, a) \tag{2}$$

A. Advantage Actor-Critic (A2C)

There are two essential components in policy gradient based methods: policy model and the value function. In vanilla policy gradient methods, the value function is not optimized although it would be very useful as the value function could assist the policy update. However, Actor-Critic methods takes optimization of value function into consideration and consists of two models:

- Critic model updates and optimizes value function parameters w.
- Actor model is used to update the policy parameters θ, using critic model evaluations.

A2C [12] is a policy gradient algorithm designed to be used specifically on parallel training. Multiple critic models learn the value function by multiple actor models as they're being trained in parallel and both global θ and w are updated asynchronously using local gradients of the parameters.

B. Trust Region Policy Optimization (TRPO)

To increase the training stability, changing policy too much by parameter updates at one step should be avoided. TRPO [13] takes this idea into account by enforcing a constraint KL (Kullback–Leibler) divergence to limit policy updates at each iteration. It still aims to maximize objective function $J(\theta)$ subject to the given trust region constraint $\mathbb{E}$ while labeling behavior policy as $\pi_{\theta old}(a|s)$ (3). However, by applying $\mathbb{E}$ (3), it enforces the distance between new and old policies, which is calculated by KL divergence, to be within a parameter δ hence the improvement would be monotonic:

$$J(\theta) = \mathbb{E}_{s\sim p^{\pi_{\theta_{old}}}, a\sim\pi_{\theta_{old}}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\widehat{A}\theta_{old}(s|a)] \\ \mathbb{E}_{s\sim p^{\pi_{\theta_{old}}}}[D_{KL}(\pi_{\theta_{old}}(.|s) \parallel \pi_\theta(.|s))] \leq \delta \tag{3}$$

It should be noted that estimated advantage $\hat{A}$ is used in (3) instead of the base advantage function A as the base rewards are generally unknown.

C. Proximal Policy Optimization (PPO)

PPO adopts the same idea of TRPO, it implements similar constraint. PPO [14] simplifies this idea by using a clipped surrogate objective by forcing ratio between new and old policies denoted as $r(\theta)$, to stay within a relatively small interval at $[1-\epsilon, 1+\epsilon]$ given ϵ is a hyperparameter.

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{4}$$

Additionally, to encourage enough exploration, the objective function is enhanced with an error term applied on the value function $(c_1(V_\theta(s) - V_{target})^2)$ and an entropy term $(c_2 H(s, \pi_\theta(.)))$ so that objective function becomes:

$$J^{CLIP}(\theta) = \mathbb{E}[J^{CLIP}(\theta) - c_1(V_\theta(s) - V_{target})^2 + c_2 H(s, \pi_\theta(.))] \tag{5}$$

Where c_1 and c_2 are hyperparameter constraints and $H(s, \pi_\theta(.))$ is entropy function.

IV. Methodology

A. Traffic Simulation

Traffic simulations in microscale are designed and executed using a microscopic traffic simulation package, SUMO (Simulation of Urban MObility) [15], which allows each vehicle to be designed and tracked individually through the network. In our network, each road is represented as edges and the lanes in each road are displayed individually. Each phase is linked to lanes, and connections are drawn accordingly.

At intersections, phases are switched on and off, where switching on means turning all the lights included in phase to green and switching off means turning all the lights to first yellow and then red. Each phase has a green time assigned to it and after green times are executed for all the phases, which is called a cycle [16]. Intersection and phase representation is shown in Fig. 2 and total cycle time is calculated as below where tp is the phase duration and tt is transition time:

$$t_{cycle} = tp_i + tt_i + tp_{i+1} + tt_{i+1} + ... \tag{6}$$

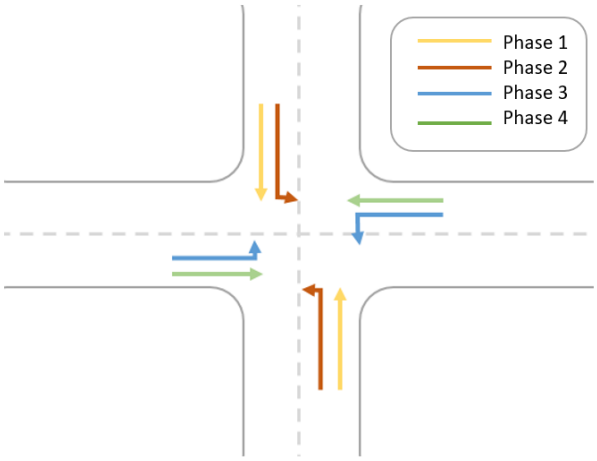

Fig. 2. Intersection and phase representation

OpenAI Gym [17] is a toolkit to enable development of reinforcement learning algorithms. The traffic environment is designed and implemented using Gym toolkit, which has defined properties and functions to modify according to each specifications. The step function is called each step of the simulation, and is used to return four values:

- Observation is the representation of the environment state.
- Reward is the amount of reward given by the previous action of the agent.
- Done is a true/false value indicating whether or not the episode ended. If the episode ended or quitted, the environment is reset by reset function.
- Info is a dictionary to be used for debugging.

The agent then chooses an action based on the observation and reward returned by step function as defined in section III.

B. State Representation

As we are using a traffic simulator for evaluation purposes, the traffic states at each step in the environments are fully observable to us. However, to be able to adapt our system to the real world dynamics safely, how we obtain the state must be available in a typical real world traffic setup. We have seen that, the most noticeable datasets are obtained by traffic detectors [18] and one of the most common detectors to obtain real time information of the traffic are induction loops that are placed under the pavement [19]. Therefore, we assume that there are loop sensors at the beginning and the end of each

incoming lane, and constrain the state space with the vehicle count and queue length inside the lane.

The current state of the simulation environment is represented as a nxm matrix where n is the edge count of the specific intersection and m is the maximum lane count, similar to concept used in [20]. While counting the vehicles, the whole capacity of lanes are taken into account and the vehicles waiting inside the intersections are ignored.

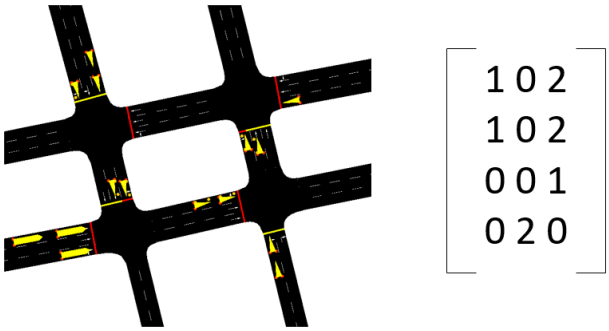

Fig. 3. Example Vehicle Positions and Corresponding Occupancy Matrix on Intersection B

o

C. Action Space Definition

Some approaches use single real value and use that value to set the duration of the next phase [21] and some of the other approaches use action to decide which phase to make a transition into [22]. However, these kind of approaches could lead to chaotic situations as the agent doesn't fully utilize the intersection dynamics as each phase should be set depending on all the phases in the intersection and doesn't take cycle demands into account. In order to avoid this, we set the phase durations off all phases at each cycle. This way, if a network has k phases, the action vector would have k components where each value is a real number for each phase duration.

The agents in reinforcement learning typically use either a discrete (or multi discrete) or continuous (or Box) action space [11]. Using continuous values, actions are expressed as real-values in the given range but discrete action space allows the agent to take a distinct action to execute from a defined finite action set [23]. We have defined the action space as given:

$$A_d = a_1, a_2, \ldots, a_k \tag{7}$$

where a is the phase duration for each phase in the range of $[0, a_{max} - a_{min}]$, a_{min} is the minimum phase time and a_{max} is the maximum phase time.

This definition allows us to utilize real world problems better as in the real world we have green time limits that each phase can execute.

D. Reward Function Design

To define the solution in the best way, the traffic has to be analyzed in a good way. To achieve this, two measurements are analyzed in this approach:

- Queue detection: Queue is defined as a line of vehicles waiting [24], therefore queuing means that the vehicles on the front of the queue is not able to flow efficiently inside an intersection. To reduce and eliminate queuing, no lane should exceed a maximum number of vehicles q_{max_i} on it, which is calculated based on the following formula:

$$q_{max_i} = \begin{cases} lane\_capacity_i * 0.6 & lane\_length_i > 100 \\ lane\_capacity_i * 0.8 & otherwise \end{cases} \tag{8}$$

  If q_i exceeds q_{max_i}, then the episode is ended with a punishment by a value R_q where q_i is the current queue length at lane i.
- Congestion measurement: Traffic congestion is defined as a condition that occurs when the use of the intersection increases and causes slower speeds for vehicles and longer trip times which leads to queues eventually [25]. Congestion is detected by first checking if there is enough queue in each lane which is denoted as lb_i and whether or not the vehicles are passing through the intersection easily by checking if there are less vehicles passed through the intersection in the cycle than the average count of vehicles passed at previous k cycles with the following formula where pv denotes passed vehicle count:

$$lb_i = lane\_capacity_i * 0.3 \tag{9}$$

$$pv\_current = total\_current - total\_previous \tag{10}$$

$$congestion = \begin{cases} true & q_i > lb_i \;\&\; pv_{current} < pv_{avg} \\ false & otherwise \end{cases} \tag{11}$$

  If a congestion is detected, then the episode is terminated with a punishment by a constant negative reward R_c.

E. Vehicle Generation Method

Interaction with the SuMO simulation is enabled by a tool called TraCI (Traffic Control Interface) [26]. TraCI provides access to the simulation and allows program to retrieve and make use of properties of the network during training. The vehicle generation process takes place at the beginning of the simulation. After each episode is ended, the present values are checked to determine whether or not a level, a vehicle trip setting generated, is solved. After a level is solved, new vehicles are generated, meaning a new level is created. Otherwise, the episode starts with the current level again.

V. Results

A. Experiment Design

In order to evaluate how well the algorithms work, we have used two different intersections with similar features of road and phase number selected in the city of Kayseri. The network is drawn using NetEdit [28], which is a tool provided

by SUMO package that is used to create and modify traffic networks. All the phases in the intersections are obtained by monitoring the intersections and the connections are drawn to indicate which paths a vehicle can use, and are shown as blue and yellow lines in Figs 4, 5.

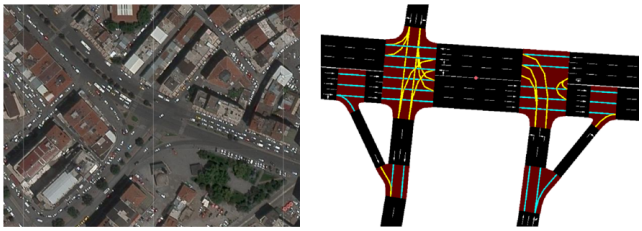

Fig. 4. Obtained image and drawing of Intersection A

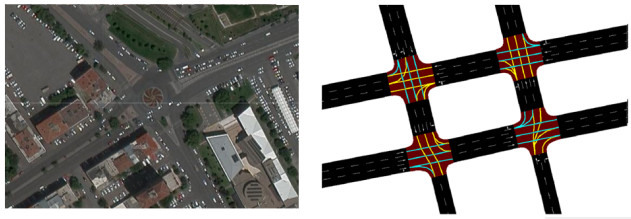

Fig. 5. Obtained image and drawing of Intersection B

The algorithms explained in section III are applied using Stable Baselines [29] from OpenAI to both junctions and adaptive parameter noise is added for further exploration. As we are using parameter noise, we used MLP (a multilayer perceptron) for both the policy and the value function. To enable bounding the total reward, a discount factor ($\gamma = 0.95$) is used where G_t is the discounted reward and R_t is the reward at time step t:

$$G_t = R_0 + \gamma R_1 + \gamma^2 R_2 + ... + \gamma^t R_t \tag{12}$$

The SuMO simulation is stochastic by its nature as a microscopic traffic simulator. Therefore, the results obtained at each level depend on the randomly generated seed set. In order to avoid algorithm overfitting to a single level, we randomize the seed and change the network at each level as number of roads, maximum number of lanes and phases are same for both networks.

B. Experiment Results

To be able to evaluate the performances of the algorithms, we compare the total reward gained and discounted rewards at one episode although our main reference measure will be the total reward. Each simulation is run for 200000 time steps due to computation costs.

In Fig. 6, we can find the performance comparison of the algorithms by comparing the total reward gained by episode. Although TRPO and PPO reaches a similar level, A2C outperforms them in a notably short period. A2C was able to eliminate queuing and congestion successfully. Additionally,

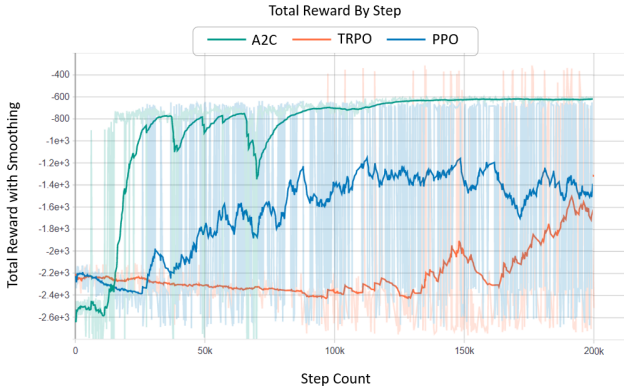

Fig. 6. Total reward with smoothing gained by step for each algorithm

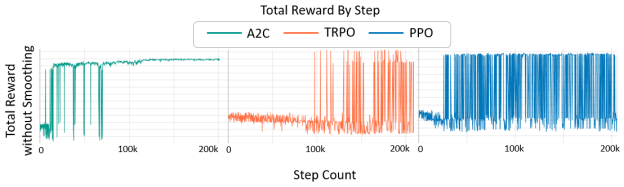

Fig. 7. Total reward without smoothing

it can be said that A2C is much stable by Fig. 6 where reward plots are shown without smoothing. We can see that there are steep decreases in the plots given in Fig. 7, which means the algorithm is punished for either queuing or creating a traffic congestion. And even though TRPO couldn't converge within the specified time range, it shows a promising trend in reward accumulation.

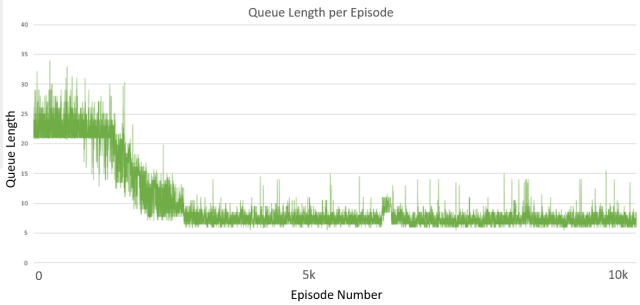

Fig. 8. Queue Length per Episode

As the objective of this paper is to decrease the queue length inside intersections and decrease the waiting times, we observed the queue lengths during training of the agents. We can see how queue length decreases over time before it stabilizes using A2C algorithm in Fig. 8. By Fig. 8, we deduce that the optimized queue length is highly dependent on the q_{max_i} value mentioned in section IV-D. Therefore, we could say that the q_{max_i} value itself needs to be optimized and R_c value should depend on the q_i instead of being a default, fixed value.

As TRPO and PPO algorithms were not able to converge and stabilize, the results are not presented here.

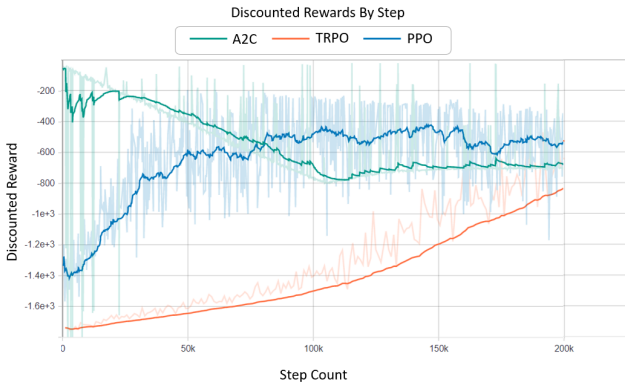

Fig. 9. Calculated discounted reward by step for each algorithm

In Fig. 9, discounted reward plot can be found. We can see that while discounted reward for TRPO gradually increases, PPO shows an increase until halfway and then converges with no further increase. Even though A2C performed best in terms of total reward, we see a decrease in discounted reward.

VI. Conclusions

The main objectives of traffic light management systems are to control vehicle traffic to prevent traffic jams and congestion in the intersections and to decrease the number of accidents caused by uncontrolled intersections. In this paper, we have studied the effectiveness of reinforcement learning to real-world vehicle traffic environments with constraints and have successfully applied three of the on-policy reinforcement learning algorithms. Comparative performance evaluations show that A2C algorithm converges well while staying stable under changing traffic dynamics and is able to decrease queue length inside intersections. Future work includes investigation of hyper parameter optimization with longer training periods with generalization.

References

[1] R. Ross, "Invention of the Traffic Light," Live Science, 2016. https://www.livescience.com/57231-who-invented-the-traffic-light.html.

[2] T. Lomax, D. Schrank and B. Eisele, "2021 Urban Mobility Report," 2021. https://mobility.tamu.edu.

[3] New South Wales Government, "SCATS: Sydney Coordinated Adaptive Traffic System," 2011. https://www.qtcts.com.au/media/512152-RTA532_SCATS_A4_Product_Brochure_07.pdf.

[4] R. D. Bretherton, "Scoot Urban Traffic Control System—Philosophy and Evaluation," IFAC Proceedings Volumes, vol. 2, no. 23, pp. 237-239, 1990.

[5] M. B. Younes and A. Boukerche, "An Intelligent Traffic Light scheduling algorithm," in 39th Annual IEEE Conference on Local Computer Networks Workshops, 2014.

[6] V. Hamakumar ve H. Nazini, "Optimized traffic signal control system at traffic intersections using VANET," Chennai Fourth International Conference on Sustainable Energy and Intelligent, 2013.

[7] M. Wiering, J. Vreeken, J. v. Veenen and A. Koopman, "Simulation and Optimization of Traffic in a City," in Intelligent Vehicles Symposium, 2004 IEEE, 2014.

[8] L. Prashanth and S. Bhatnagar, "Reinforcement Learning With Function Approximation for Traffic Signal Control," in IEEE Transactions on Intelligent Transportation Systems, 2011.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra ve M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," CoRR, 2013.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," in ICLR , 2016.

[11] R. S. Sutton and A. G. Barto, "1.1 Reinforcement Learning," in Reinforcement Learning: An Introduction, The MIT Press, 2017.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," Proceedings of Machine Learning Research, no. 48, pp. 1928-1937, 2016.

[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan and P. Moritz , "Trust Region Policy Optimization," Proceedings of Machine Learning Research, no. 37, pp. 1889-1897, 2015.

[14] J. Schulman, F. Wolski, P. Dhariwal and A. Radford, "Proximal Policy Optimization Algorithms," OpenAI Researches, 2017.

[15] Eclipse Foundation, "Simulation of Urban MObility," Eclipse Foundation. https://www.eclipse.org/sumo/.

[16] F. H. A. (FHWA), "Traffic Signal Design," in Signal Timing Manual - Second Edition, Federal Highway Administration (FHWA), 2008.

[17] OpenAI, "Gym," OpenAI. https://gym.openai.com/.

[18] Papers With Code, "PeMSD7 - Traffic Dataset," Papers With Code, 2012. https://paperswithcode.com/dataset/pemsd7.

[19] A. Spears, "Sensors At Traffic Lights," Eltec Corp., 2019. https://elteccorp.com/news/other/are-there-sensors-at-traffic-lights/.

[20] J. A. Calvo and I. Dusparic, "Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control," in AICS 2018, 2018.

[21] H. Joo and Y. Lim, "Traffic Signal Time Optimization Based on Deep Q-Network," Applied Sciences, 2021.

[22] M. Coşkun, A. Baggag and S. Chawla, "Deep Reinforcement Learning for Traffic Light Optimization," in IEEE International Conference on Data Mining Workshops (ICDMW), 2018.

[23] W. Masson, P. Ranchod and G. Konidaris, "Reinforcement Learning with Parameterized Actions," in AAAI Conference on Artificial Intelligence, 2016.

[24] Federal Highway Administration (FHWA), "Definition, Interpretation, and Calculation of Traffic Analysis Tools Measures of Effectiveness," in Traffic Analysis Tools, U.S. Department of Transportation, 2021.

[25] Ministry of Transport, New Zealand, "The Congestion Question, Could road pricing improve Auckland's traffic?," Auckland Council, 2019.

[26] German Aerospace Center (DLR), "TraCI," German Aerospace Center (DLR). https://sumo.dlr.de/docs/TraCI.html.

[27] Yandex, "Yandex Maps," Yandex. https://yandex.com.tr/harita.

[28] German Aerospace Center, "NetEdit," German Aerospace Center. https://sumo.dlr.de/docs/Netedit/index.html.

[29] OpenAI, "Stable Baselines," OpenAI, 2018. https://stable-baselines.readthedocs.io/en/master/.

[30] D. Kahneman, A. B. Krueger ve D. A. Schkade, A Survey Method for Characterizing Daily Life Experience: The Day Reconstruction Method, Science, cilt 306, no. 5702, pp. 1776-1780, 2004.

[31] N. Lelyveld and S. Grad, "Traffic still tops crime, economy as top L.A. concern, poll finds," Los Angeles Times, 2015. https://www.latimes.com/local/lanow/la-me-ln-traffic-still-tops-crime-economy-as-top-l-a-concern-poll-finds-20151007-story.html.

[32] J. J. Kim, S. Smorodinsky, M. Lipsett, B. C. Singer, A. T. Hodgson ve B. Ostro, Traffic-related Air Pollution near Busy Roads The East Bay Children's Respiratory Health Study, American Journal of Respiratory and Critical Care Medicine, cilt 5, p. 170, 2004.

[33] Turkish Statistical Institute, TUIK, "Road Motor Vehicles, March 2022," Turkish Statistical Institute, TUIK, March 2022. https://data.tuik.gov.tr/Bulten/Index?p=Road-Motor-Vehicles-March-2022-45706.

[34] L. Weng, "Curriculum for Reinforcement Learning," 2020. https://lilianweng.github.io/posts/2020-01-29-curriculum-rl/.

[35] OpenAI, "Quantifying Generalization in Reinforcement Learning," OpenAI, 2018. https://openai.com/blog/quantifying-generalization-in-reinforcement-learning/.

[36] N. Casas, "Deep Deterministic Policy Gradient for Urban Traffic Light Control," 2017.

[37] L. Weng, "Policy Gradient Algorithms" 2018. https://lilianweng.github.io/posts/2018-04-08-policy-gradient/.