A Deep On-Policy Learning Agent for Traffic Signal Control of Multiple Intersections

Chia-Cheng Yen[1], Dipak Ghosal[1], Michael Zhang[2], and Chen-Nee Chuah[3]

***Abstract*— Reinforcement Learning (RL) is being rapidly adopted in many complex environments due to its ability to leverage neural networks to learn good strategies. In traffic signal control (TSC), existing work has focused on off-policy learning (Q-learning) with neural networks. There is limited study on on-policy learning (SARSA) with neural networks. In this work, we propose a deep dueling on-policy learning method (2DSARSA) for coordinated TSC for a network of intersections that maximizes the network throughput and minimizes the average end-to-end delay. To describe the states of the environment, we propose traffic flow maps (TFMs) that capture head-of-the-line (HOL) sojourn times for traffic lanes and HOL differences for adjacent intersections. We introduce a reward function defined by the power metric which is the ratio of the network throughput to the average end-to-end delay. The proposed reward function simultaneously maximizes the network throughput and minimizes the average end-to-end delay. We show that the proposed 2DSARSA architecture has a significantly better learning performance compared to other RL architectures including Deep Q-Network (DQN) and Deep SARSA (DSARSA).**

***Index Terms*— Deep reinforcement learning, Traffic signal control, Multi-intersection control, On-policy learning, Traffic flow maps, Power metric**

I. INTRODUCTION

Due to the mobility needs of the increasing population [1], new understanding of the health impacts of traffic related emissions [2], and the technological developments in connected and autonomous vehicles (CAVs), there is significant new research in traffic signal control (TSC). Besides improving safety, there are new opportunities and challenges to improve scheduling algorithms in TSC to increase the network throughput and lower average end-to-end delay. Towards this end, there is increasing trend in applying Reinforcement Learning (RL) [3] algorithms. A detailed survey on applying RL algorithms to TSC is summarized in [4]. As such, conventional RL algorithms have been shown to perform well for simple traffic environments with static traffic demands and regular traffic patterns [5]–[8]. However, for complex traffic environments including coordinated control of a network of intersections, the state space grows exponentially with the number of intersections. A large state space degrades the learning performance of a conventional RL agent as it may fail to explore all the state-action pairs within a reasonable time. This problem is referred to as the curse of dimensionality [9].

[1]Chia-Cheng Yen and Dipak Ghosal are with the Department of Computer Science, University of California, Davis, CA 95616, USA `ccyen@ucdavis.edu, dghosal@ucdavis.edu`

[2]Michael Zhang is with the Department of Civil and Environmental Engineering, University of California, Davis, CA 95616, USA `hmzhang@ucdavis.edu`

[3]Chen-Nee Chuah is with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA `chuah@ucdavis.edu`

The function approximation [10]–[12] using a neural network is a promising solution to the large state space problem. In this approach, Q-values are approximated by deep neural networks such as convolutional neural network (CNN). The methods usually require a significant number of training episodes to train a model because the action selection is learned by several layers of the neural network rather than updating values in a table. The deep neural network must be run through many episodes in order to fine-tune the weights through back propagation. Moreover, the convergence of the weights is not guaranteed if neural networks with non-linear activation functions are used.

In this work, we design a centralized TSC controller with a deep RL agent (DRL-agent) that is trained by a novel deep dueling on-policy learning method referred to as 2DSARSA. To the best of our knowledge, this is the first work applying 2DSARSA for a coordinated control of a network of intersections. While many studies have shown good prospects on DQN, our results show that DQNs do not converge quickly and do not yield optimal results. The proposed 2DSARSA performs better and converges faster than DQN.

The key contributions of this paper are as follows:

- We propose a deep dueling SARSA based RL-agent referred to as 2DSARSA, which combines the on-policy temporal-difference (TD) learning, deep learning, dueling architecture [13], and experience replay memory [14].
- We propose traffic flow maps (TFMs) as states to capture flow dynamics of a network of intersection. Using the CNN, the 2DSARSA agent is able to learn traffic features from the proposed TFMs.
- We introduce a reward function using the power metric, which maximizes the network throughput and minimizes the average end-to-end delay of a network. Our experiments show that this power metric enhance the learning performance of the proposed on-policy method.
- We compare our method with deep Q-Network (DQN) [14], double deep dueling Q-Network (3DQN) [15], and DSARSA [16] in terms of their learning performance.

There is insufficient study on the comparison between deep on-policy (DSARSA) and deep off-policy (DQN) in the context of TSC. This paper is a first step towards addressing this gap for TSC researchers.

II. System Model and Problem Statement

We consider a 3 by 3 grid network of 9 intersections. At each intersection there are two traffic signal phases - one for North-South and the other for East-West. There is a single lane in each direction - one lane for North-to-South and one for South-to-North. For each intersection, the decision of which traffic signal phase to activate is given by the centralized controller based on the DRL-agent shown in Fig. 1.

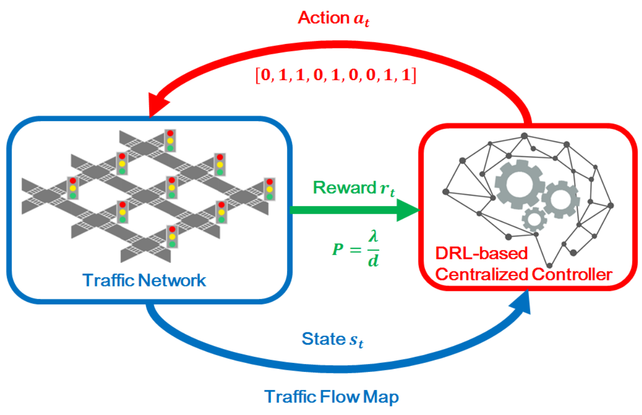


Fig. 1: A 3 by 3 grid traffic network consisting of 9 intersections. At each intersection there are only two possible movements - North-South and East-West. The DRL-agent in the centralized controller determines the selection of the phase for each intersection for every time slot.

The goal of DRL-agent is to schedule non-conflict traffic movements for the network so as to maximizes the network throughput and minimizes the average end-to-end delay. All the TSCs are synchronized and operate per time slot of duration t. At beginning of every time slot t, the DRL-agent receives the state of network denoted by s_t which is described by the Traffic Flow Map (TFM). Based on the state s_t, the DRL-agent takes an action a_t which is an assignment of traffic signal phases to each intersection. Since we consider only two movements at each each intersection, action a_t is a bit vector of length 9 as shown in Figure 1. For each interval, the DRL-agent receives a reward r_t which indicates how well the action a_t in state s_t impacts the the network throughput and/or average end-to-end delay. This is determined by the reward function that is discussed in Section II-C.

A. State Space

We propose to describe the state using a traffic flow map (TFM) which is an image encoding the Head-of-Line (HOL) sojourn times of each lane at each intersection and HOL sojourn times differences between adjacent intersections. Previous studies [10]–[12], [15], [17] have used queue length of vehicles at the intersection to describe the state. However, studies [18], [19], show that delay-based backpressure (BP) scheduling algorithm achieves better fairness than using queue lengths (queue-based).

Let $W_{i,k}(t)$ denote the HOL sojourn time of lane i at intersection k at time slot t in the traffic network. Employing the methods proposed in [20], we define two metrics for lane i at intersection k - the delay metric $\hat{W}_{i,k}$ and delay difference metric $\Delta\hat{W}_{i,k}$ as follows:

$$W_{i,\hat{k}}(t) = W_{i,k}(t) - W_{i,k-1}(t) \tag{1}$$

$$\Delta W_{i,\hat{k}}(t) = W_{i,\hat{k}}(t) - W_{i,\widehat{k+1}}(t) \tag{2}$$

where $k-1$ and $k+1$ denote the neighbouring upstream and downstream intersections, respectively. If an intersection has no neighbors (the edge intersections in our grid network), the HOL sojourn time values are set to 0. These delay metrics guarantee a linear relationship between queue lengths and delays for a traffic network with multiple intersections. Note that similar to the delay metrics one can define queue metrics based on the differences in the queue lengths between the adjacent intersections.

The traffic flow information of each intersection in terms of the delays and the delay metrics is stored in a 5×5 matrix as shown in Fig. 2. Note that for each movement there is one lane in each direction. Consequently, there are 4 HOL sojourn times. These are denoted by the orange circles in Fig. 2. The values of these circles will be the corresponding HOL sojourn time values. The blue circle in the center indicates the average HOL value of 4 HOL sojourn times. The red and brown circles represent delay metrics defined in Eq.1 and Eq.2 with corresponding neighbouring intersection. Note

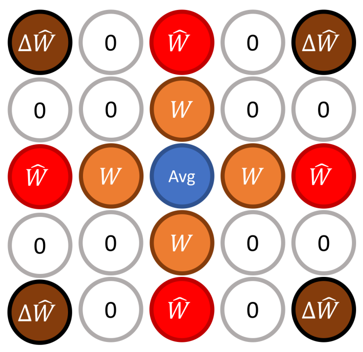

$\Delta\hat{W}$ 0 $\hat{W}$ 0 $\Delta\hat{W}$
0 0 W 0 0
$\hat{W}$ W Avg W $\hat{W}$
0 0 W 0 0
$\Delta\hat{W}$ 0 $\hat{W}$ 0 $\Delta\hat{W}$

Fig. 2: The traffic state information of one intersection encoding HOL sojourn times $W_{i,k}(t)$, delay metrics $W_{i,\hat{k}}(t)$ and delay difference metrics $\Delta W_{i,\hat{k}}(t)$.

value of each circle is a real number that can be mapped to a color scale in which high values are in red and low values are in yellow.

The TFM of the entire network is created by laying out the traffic state information of each intersection in a two dimensional grid while maintaining the adjacency of the intersections. The TFM for the 3×3 grid network (consisting of 9 intersections) is shown Fig. 3. The figure shows the TFM at time slot 455, 1155, and 1755, respectively. Note each TFM consists of 9 tiles each of which is a 5×5 matrix of data values shown in Fig. 2. These TFMs encode the delay information (the HOL sojourn times and the delay metrics) of the traffic flow in the network as it evolves over time. For the example shown in Fig. 3, the colours shows that traffic congestion has become worse over time.

B. Action Space

At each time slot t, each intersection can activate a traffic flow from either North-South or East-West directions. The action a_t is a bit vector, i.e., $a_t = \{0, 1\}^M$ where M is the number of intersections and $0, 1$ denotes which direction should be activated, 0 for East-West direction, 1 for the North-South direction.

C. Reward Function

The reward function should be carefully designed because temporal-difference (TD) methods update the Q-values for state-action pairs based on the estimate of reward. On-policy learning estimates rewards for state-action pairs by assuming that the same policy will be followed. In contrast, off-policy learning estimates rewards by selecting actions which maximize the expected cumulative reward. We found that on-policy learning is sensitive to reward functions. Compared to off-policy learning, on-policy learning which follows the initial policy is unable to explore other policies. Hence, the evaluation of the initial policy which continues to be followed is crucial to on-policy learning.

In this work we consider three different reward functions.

1) **Throughput based reward**: In this case high reward is given to state-action pairs that have high network throughput, denoted by λ.
2) **Average end-to-end delay based reward**: In this approach higher reward is assigned to state-action pairs that have lower average end-to-end delay denoted by d.
3) **Power based reward**: We define the power metric P as

$$P = \frac{\lambda}{d} \tag{3}$$

This reward function assigns higher reward to state-action pairs that have high P value. Note that maximizing P implies that the network throughput is maximized and the average end-to-end delay is minimized. The power metric has been used to design protocols in computer networks including [21]. Operating a packet switched network that maximizes power is referred to as Klienrock's optimal operating point as it achieves the maximum network throughput at the minimum end-to-end delay.

Both the network throughput and average end-to-end delay are calculated based on the vehicles which have exited the network during the current time slot. For larger networks it may be more appropriate to calculate the network throughput and the average end-to-end delay over a number of time slots to account for the nominal delay of vehicles going through the network which would result in a delay between the action and response (hence the reward) from the environment. In Section IV, we show that power based reward function achieves better performance and has better learning performance than using the only throughput or average end-to-end delay based reward.

III. DEEP DUELING SARSA (2DSARSA) FOR MULTIPLE INTERSECTIONS

The proposed 2DSARSA adopts SARSA (on-policy algorithm) [22], dueling architecture [13], deep neural networks (CNNs), and experience memory replay [14]. The advantages of the above techniques can be summarized as follows. First, our preliminary experiments reveal that SARSA has better performance than Q-learning [23]. Second, dueling architecture helps the selection of action by separating the estimations of state-value and action advantage. It benefits the 2DSARSA agent if the action space is large. Finally, the experience memory replay breaks the strong temporal correlations and speeds up the training process. In the following subsections we outline the key aspects of the implementation.

A. Learning based on Deep SARSA

Deep SARSA refers to the on-policy TD learning algorithm using a deep neural network as a function approximator. Similar to [14], a parameterized neural network replaces the conventional tabular approach for storing the Q-values for state-action pairs. The inputs to the Q-network are TFMs, the outputs are actions with the optimal Q-value over all state-action pairs, and θ is a set of network parameters. During the training stage, the loss function is defined by

$$L(\theta_t) = (y_t^{DSARSA} - Q^{\pi}(s_t, a_t; \theta_t))^2 \tag{4}$$

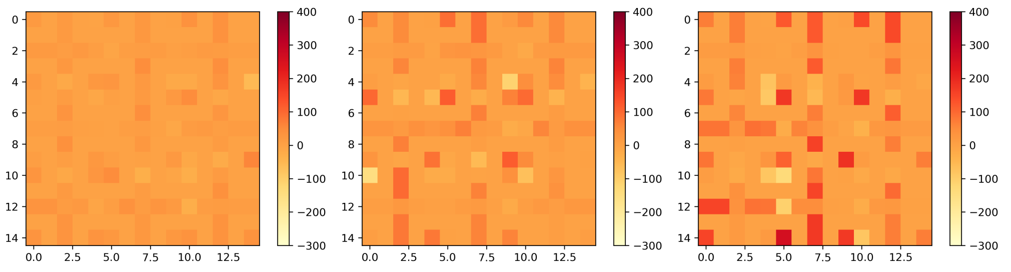
$$y_t^{DSARSA} = r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1}; \theta_t^{-}) \tag{5}$$

where s_t is the current state, a_t is the current action, s_{t+1} is the next state which is dependent on a_t, a_{t+1} is the next action selected by ϵ-greedy, and y_t^{DSARSA} is the TD target given by Eq. 5. The goal of the 2DSARSA agent is to update θ that optimizes the loss function $L(\theta_t)$. First, the partial derivative of $L(\theta_t)$ is calculated by differentiating Eq. 4. The gradient of the loss function can be obtained by

$$\bigtriangledown_{\theta_t} L(\theta_t) = (y_t^{DSARSA} - Q^{\pi}(s_t, a_t; \theta_t)) \bigtriangledown_{\theta_t} Q^{\pi}(s_t, a_t; \theta_t) \tag{6}$$

where $\bigtriangledown_{\theta_t} Q^{\pi}(s_t, a_t; \theta_t)$ denotes the gradient of the current state-action value. We optimize the loss function $L(\theta_t)$ by Eq. 6 and the network parameter set θ is updated by using stochastic gradient descent (SGD).

The action space considered in this work has a cardinality of 2^M which degrades the learning performance of the 2DSARSA agent. We apply dueling network architecture to solve the large action space issue because the agent, which is trained by dueling network architecture, performs well in a large action space. The overall architecture of the proposed 2DSARSA uses two separate neural networks [14] (Q-network and target network), dueling network architecture [13], and experience memory replay [14]. Given the on-policy TD learning algorithm, in each time slot, the current TFM as the current state is fed to the Q-network, previous experiences are fed to the target network, and at the same time the current experience $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ is pushed



(a) $t = 455$ (b) $t = 1155$ (c) $t = 1755$

Fig. 3: TFM for the 3×3 grid network of 9 intersections as it evolves over time. Note each TFM consists of 9 tiles each of which is a 5×5 matrix of data values shown in Fig. 2.

into the experience replay memory which stores the latest M experiences. Experience replay breaks the highly temporal correlation between consecutive samples to increase learning efficiency. Based on mini-batch size B, batches are randomly sampled from the replay memory. The 2DSARSA agent learns through mini-batches that increases the efficiency of the training as opposed to learning over full observations.

IV. Results and Discussions

For training, we generated 1-hour of traffic arrivals following the Poisson process. We trained the agents based on different DRL algorithms (DSARSA, 2DSARSA, DQN, and 3DQN summarized in Table II). The agents were trained using 1000 episodes; they were pre-trained with the first 100 episodes to accumulate replay data. Each agent took 24 to 27 hours to complete 1000 episodes. For the test stage, 10-hours traffic arrivals were used to verify the robustness of the agent. The reward, average end-to-end delay, and average queue length were used to evaluate the learning performance of the agents. The hyper-parameters are shown in Table I.

TABLE I: Summary of hyper-parameters and their corresponding values.

Hyperparameters	
Traffic demand	$\vec{D} = [0.2, 0.5, 0.2, 1] \times 0.125 \times 1.5$ v/s/l
Time slot	$t = 5$ sec
ϵ-greedy decay function	$\epsilon_f + (\epsilon_i - \epsilon_f) \times e^{-\frac{episode}{300}}$
Initializing ϵ	$\epsilon_i = 1.0$
Finalizing ϵ	$\epsilon_f = 0.01$
Learning ratio	$\alpha = 0.0001$
Discount ratio	$\gamma = 0.99$
Memory size	$M = 30000$
Mini-batch size	$B = 1024$
Episodes	$E = 1000$

A. Learning Performance of The Proposed 2DSARSA Agent using Different Reward Functions

Fig. 4 shows the reward and average end-to-end delay during the training stage for 2DSARSA with different reward functions. We first explain why there are spikes around the 100-th training episode. Note that we choose to pre-train with the first 100 episodes to accumulate tuples of the previous state, previous action, reward, current state, and current action (a.k.a experiences) for the replay memory. Initially, the number of experiences which can be sampled is insufficient for the agent to accurately compute the loss between the target Q-value and the current Q-value and update the neural network. For the first 100 episodes, the 2DSARSA agent randomly selects actions for exploration while accumulating replay data.

The results in Fig. 4a show that the reward quickly increased after the 100-th episode and continued to increase with the number of episodes. The average end-to-end delay decreased quickly after pre-training, and converged to a stable value around the 300-th episode. In contrast, DSARSA required 1000 episodes to converge to the same performance level as 2DSARSA (not shown here due to space limitations). Overall, the curves of the reward and average end-to-end delay converged eventually. The 2DSARSA agent with the on-policy learning algorithm is indeed capable of learning from high dimensional states and performing well in large action space. Furthermore, dueling architecture stabilizes and speeds up the training process.

Compared to Fig. 4a, Fig. 4b and 4c show the results of using throughput-based reward and average end-to-end delay based reward. Note that these reward functions have different scales. For these functions, the reward and average end-to-end delay were unstable and did not converge to the same performance level as the power based reward function. Unlike the power metric which maximizes the network throughput and minimizes the average end-to-end delay, they consider only one factor when updating Q-values, i.e., either the throughout or delay. In many Atari games which have comparatively simpler environments, agents are aware of how they perform by only one measurement, which is the *score*. However, such a one-factor based measurement to describe part of the agent's performance is not suitable for a complex environment such as coordinated control of multiple intersection. These results demonstrate that the robustness of the power based reward function.

B. Performance of the Proposed 2DSARSA Agent in 10-hour Traffic Arrival Data

Once the training was completed, the models of the neural networks were stored and tested for 10-hour traffic arrival data. The performance of BP scheduling schemes (QBPC and

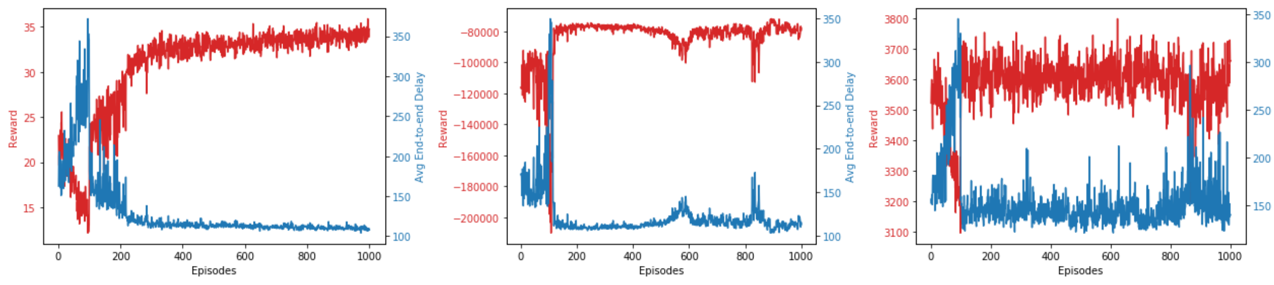

(a) Power based reward function (b) Average end-to-end delay based reward function (c) Throughput based reward function

Fig. 4: The reward and average end-to-end delay during the training stage by the proposed 2DSARSA with different reward functions.

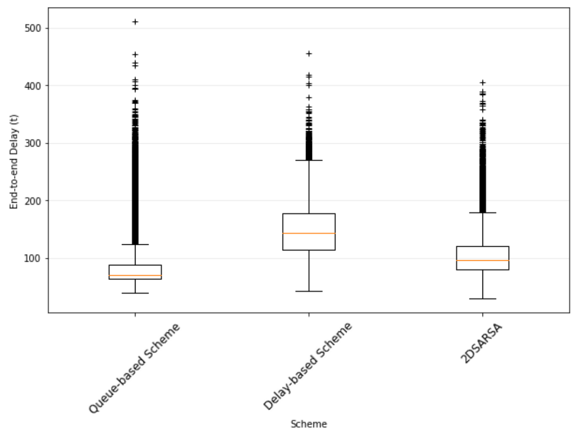


Fig. 5: Performance comparison on the delay with baseline algorithms: queue-based BP control (QBPC) and delay-based BP control (DBPC). 10-hour traffic arrival data is used as the test data. 2DSARSA was trained by using the power based reward function.

DBPC) as the baseline is compared to our proposed method in Fig. 5. BP-based schemes determine a traffic signal phase based on the highest traffic pressure (maximum queue length) in TSC problems [18], [19], [24]. In each intersection, a BP-based controller independently determined the phase without considering any additional information from its neighbors.

Fig. 5 shows that the outliers (vehicles) which could encounter with a longer sojourn time. In QBPC and DBPC, the sojourn time which outliers encounter are close to 500 sec. With heterogeneous arrivals, DBPC has a shorter sojourn time which vehicles could encounter, and it achieves a better fairness than QBPC [18], [20]. On the contrary, the performance of the 2DSARSA agent achieves a even shorter sojourn time for outliers and a better fairness for all vehicles than DBPC. It is quite obvious that the 2DSARSA outperforms the conventional BP-based algorithm proposed in [18], [20] for multiple intersections after 1000 training episodes. Regarding the overall fairness, less outliers (vehicles) would experience a end-to-end delay of more than 400 sec compared to QBPC, DBPC, and other DRL-agents (not shown here due to space limitations).

C. Learning Performance of Different DRL-Agents

TABLE II: Properties of different DRL-agents compared in this study.

Methods	TD Learning	Neural Network	Architecture
DQN [14]	Q-learning	CNN	Single
3DQN [15]	Q-learning	CNN	Dueling
DSARSA [16]	SARSA	CNN	Single
2DSARSA	SARSA	CNN	Dueling

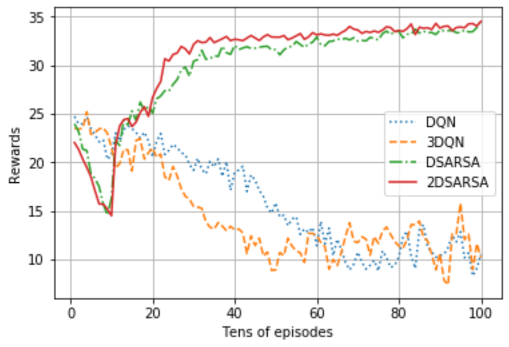


Fig. 6: Learning performance of DQN [14], 3DQN [15], DSARSA [16], and 2DSARSA. The power based reward function was applied to all agents.

We implemented different DRL-based methods shown in Table II for comparison. The learning performance of DQN and 3DQN show that Q-learning with deep learning and advanced architectures is not appropriate for the traffic network environment. The fundamental difference between on-policy and off-policy methods is that off-policy learns by searching the best action from several policies. DQN and 3DQN utilize an off-policy learning method, which switches among multiple policies to maximize the expected accumulated reward. In many Atari games that apply Q-learning, this approach easily eliminates a poor initial policy and jumps to a promising one, which returns a better accumulated reward. However, as the state and action space grow exponentially for complex environments, this approach has limitations. Fig. 6 shows learning performance of different agents in terms of the reward (based on the power metric). DQN and 3DQN agents were not able to learn effectively from the environment and had convergence issues. The reward, average end-to-end delay and average queue length had high fluctuations towards the end of the training (not shown due to space limitations).

On the other hand, DSARSA and 2DSARSA were capable of learning effectively from the environment as shown in Fig. 6. Without greedily searching the best policy, DSARSA learns by following the initial policy and selecting the next action based on this policy. These algorithms performed significantly well during the training stage and both of their training processes converged at the end. On-policy learning with deep learning shows a certain level of stability while learning from a complicated environment. Because on-policy learns by following the same policy while training, the reward converges even if the policy chosen by SARSA is initially not good enough. The dueling technique, which separates the state-value and action advantage, can effectively filter out useless actions and force the entire training to converge eventually. 2DSARSA using dueling architecture significantly reduced oscillation, and converged faster during the training stage.

In summary, 2DSARSA combining dueling architecture and DSARSA, outperforms other methods and agents. Based on our experiments, purely applying BP-based algorithms to a traffic network does not achieve global optimality. Further optimization could still be achieved. Learning by following the same policy provides a good entry point to explore better performance when doing scheduling in a traffic network.

V. CONCLUSION

Previous studies mainly focused on applying off-policy methods with deep learning to the traffic control problem. Off-policy methods generally work well in a simple environment such as an isolated intersection. However, they do not work well in complicated environments like a grid network. The proposed 2DSARSA is a deep dueling on-policy method using TFMs as inputs and applying power metric as the reward function. It performs well in a more complex environment than a single intersection. Our results have shown a promising prospect of using 2DSARSA for multi-intersection traffic light control. A 2DSARSA based centralized controller learning by TFMs and the power based reward is able to achieve better fairness than BP-based algorithms and other DRL-based agents.

For our future work, we will consider larger real traffic networks with more traffic phases at each intersection. As the power metric only considers performance (network throughput and average end-to-end delay) we will study how fairness can be incorporated into the reward function. For larger networks we will investigate a multi-level control which is more suitable for the estimation and action selection than the dueling architecture.

REFERENCES

[1] "UN estimates: 68% of the world population projected to live in urban areas by 2050," https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html, 16 May 2018.

[2] "Fast facts: U.s. transportation sector greenhouse gas emissions 1990-2017," https://www.epa.gov/greenvehicles/fast-facts-transportation-greenhouse-gas-emissions.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction, 2nd edition*. MIT press, Oct. 2018.

[4] A. Haydari and Y. Yilmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," May 2020, arXiv:2005.00935.

[5] T. L. Thorpe and C. W. Anderson, "Traffic light control using sarsa with three state representations," no. Technical Report, 1996.

[6] M. Abdoos, N. Mozayani, and A. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in *Proc. IEEE 14th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 5-7 Oct. 2011, pp. 1580–1585.

[7] S. Araghi, A. Khosravi, M. Johnstone, and D. Creighton, "Q-learning method for controlling traffic signal phase time in a single intersection," in *Proc. IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 6-9 Oct. 2013, pp. 1261–1265.

[8] J. Jin and X. Ma, "Adaptive group-based signal control by reinforcement learning," *Transportaion Research Procedia*, vol. 10, pp. 207—-216, 2015.

[9] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, Sept. 2013.

[10] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," Nov. 2016, arXiv:1611.01142.

[11] E. van der Pol and F. A. Oliehoek, "Coordinated Deep Reinforcement Learners for Traffic Light Control," in *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, 2016.

[12] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," May 2017, arXiv:1705.02755.

[13] Z. Wang, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot, and N. D. Freitas, "Dueling network architectures for deep reinforcement learning," Apr. 2016, arXiv:1511.06581.

[14] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[15] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, Feb. 2019.

[16] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on sarsa," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 6-9 Dec. 2016.

[17] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[18] J. Wu, D. Ghosal, M. Zhang, and C.-N. Chuah, "Delay-based traffic signal control for throughput optimality and fairness at an isolated intersection," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 2, pp. 896–909, Feb. 2018.

[19] C.-C. Yen, D. Ghosal, M. Zhang, C.-N. Chuah, and H. Chen, "Falsified data attack on backpressure-based traffic signal control algorithms," in *IEEE Vehicular Networking Conference (VNC)*. IEEE, 5-7 Dec. 2018, pp. 588–595.

[20] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1539–1552, Oct. 2013.

[21] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad Hoc Netw.*, vol. 80, pp. 142–157, 2018.

[22] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," vol. 37, 1994.

[23] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[24] T. Wongpiromsarn, T. Uthaicharoenpong, Y. Wang, E. Frazzoli, and D. Wang, "Distributed traffic signal control for maximum network throughput," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.* IEEE, 16-19 Sept. 2012, pp. 588–595.