Adaptive and Responsive Traffic Signal Control using Reinforcement Learning and Fog Computing

Chengyu Tang
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, Alabama USA
tangcy@auburn.edu

Sanjeev Baskiyar
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, Alabama USA
baskisa@auburn.edu

***Abstract*—As urban traffic becomes increasingly complex, conventional actuated traffic signal control methods are showing their limitations in mitigating congestion, and Deep Reinforcement Learning (DRL) is considered a promising solution. Existing studies mainly focus on using distributed agents, each intersection being controlled by a separate agent that can communicate with its neighbors to facilitate local coordination, which subsequently increases network complexity and usage. This article presents a DRL-based intelligent traffic signal control framework that leverages the fog computing paradigm. Traffic signals are divided into clusters, each controlled by a shared DRL agent in a fog node based on the collective information of the regional traffic situation. We have conducted simulation experiments in multiple scenarios, and the experimental results show that the proposed method yields lower average waiting times compared to existing methods.**

***Index Terms*—Traffic Signal Control, Reinforcement Learning, Fog Computing**

I. Introduction

Traffic congestion is becoming an increasingly severe problem, especially in urban areas, and it accounts for various impacts on human well-being: greenhouse gas emissions, energy waste, outdoor air pollution, and economic cost, to name a few [1], [2]. Hence, improving traffic efficiency by intelligently controlling traffic signals has received massive attention.

As traditional methods such as fixed time control (FTC) start to show their limitations in improving traffic efficiency, many studies have been conducted related to intelligent traffic signal control (ITSC) strategies. Deep reinforcement learning (DRL), among many others, has shown its efficacy in this domain because of its capability to search for optimal policy in dynamic environments. Most RL-based approaches utilize distributed patterns, where each intersection is controlled by a standalone DRL agent that makes signaling decisions based solely on local traffic data. Therefore, Multi-Agent Reinforcement Learning (MARL) methods have been proposed to facilitate coordination between intersections. Specifically, the agents take into account the traffic data from adjacent intersections, which allows signal control for larger-scale urban networks. However, these methods typically involve data exchange among all neighboring intersections, which consequently increases the network complexity and usage, thus resulting in additional investment in infrastructure and energy. Moreover, information from just neighboring intersections may not suffice, especially in urban areas, where intersections are densely distributed.

While cloud computing has great potential to tackle the cost issues related to computing complexity, it is also well-illustrated that network delay or response latency in cloud computing is inevitable, which conflicts with the strict response time requirement of ITSC. Fog computing, on the other hand, has been proposed to address the latency problem. Although several studies have presented various methods and evaluations based on separate contexts, Fog Computing-based ITSC in combination with DRL is a promising but under-explored direction in the existing literature. Therefore, we introduce an intelligent traffic signal control framework using deep reinforcement learning that can be applied to fog computing environments, which is optimized to lower the vehicles' waiting times. In this framework, the studied road networks, if necessary, were divided into sub-networks, each containing a cluster of multiple intersections collectively controlled by a shared fog node. Traffic data are collected and processed by edge nodes deployed at the intersections and then sent to the fog node, where they are aggregated. Based on the aggregated data, the fog node makes signaling decisions simultaneously for all traffic signals controlled by it, thus increasing regional traffic efficiency.

The main contributions of this study are:

- We presented a DRL-based cooperative intelligent traffic signal control framework designed for fog computing environments.
- We conducted simulation experiments on both synthetic and real-world scenarios, and the results indicate that our framework outperforms the baseline methods regarding the average waiting time and time loss.

The remainder of this paper is organized as follows. Section II briefly summarizes related studies with limitations, as well as a summary of the contribution of this study. Section III describes the proposed framework. Experiments and results are presented in Section IV. Finally, conclude this paper in Section V.

II. Background and Related Work

A. Adaptive Traffic Signal Control

A common improvement to FTC is to implement pre-determined offsets for adjacent traffic signals to create a "green wave" that prevents frequent stops, but it requires extensive fine-tuning and is typically effective only in one direction. With the advancement in sensing technology and computer vision, real-time traffic data collection has become feasible, enabling actuated traffic signal control. Varaiya [3] presented Max-Pressure control, a greedy method that selects the phase with the highest "pressure", which is defined as the difference in the total queue length between the approaching and the exiting lanes involved in the corresponding phase. However, it struggles to achieve satisfactory performance in modern traffic situations.

B. DRL-Based TSC

A reinforcement learning problem can be described as a Markov Decision Process (MDP) composed of five components $< S, A, p, R, \gamma >$, denoting the state space, action space, state transition probability, reward function, and the reward discount factor, respectively. Here we summarize some existing studies regarding the five components of RL.

- **State**. In the majority of literature, traffic statistics of all lanes are stacked/concatenated into a matrix or vector. The data normally include queue lengths, waiting time, and speeds, etc. The pressure of movements is used in [4], [5], which yields applausible performance. Because of the inherent temporal dependency of traffic, frame-stacking and/or recurrent mechanisms are also employed to capture traffic dynamics [6]. On the other hand, van der Pol et al. use a bird's view of the intersection transformed into a 2D matrix, representing vehicle positions [7], which, however, results in a much more sparse state representation.
- **Action**. The action space is typically defined in two ways: the duration and the next phase. The former usually implies that the order of the phases is fixed, and the agent determines the duration of the subsequent phase when the maximum duration of the current phase has been reached. The latter action design provides more flexibility by selecting one phase from a list of available phases for the next Δt time period [8].
- **Reward**. The reward is supposed to be related to the goal we are optimizing for. A widely used reward is to use the negative of the number of queueing vehicles as a penalty. In [4], [5], the negative of the total pressure of an intersection is used as the reward signal.
- **State Transition**. The state transition depends on the driving patterns of the vehicles, which are typically not directly accessible to the agent.
- **Discount factor**. The discount factor implies how much the agent cares about the future reward. Among the related studies, its value is typically within the $[0.9, 0.99]$ interval.

Simply utilizing distributed independent individual agents yields superior traffic efficiency over traditional TSC strategies. However, the information that a single intersection can rely on is limited. Some studies have leveraged information exchange to enhance the coordination among traffic signals [9], [10]. On the flip side, communications among intersections result in a more complex communication network and higher network usage. In addition to perimeter information, researchers came up with various strategies to capture the spatiotemporal characteristics of the traffic network, such as graph convolution neural networks [11] and graph attention networks [6].

C. Fog Computing

The term "Fog Computing" was introduced by Cisco in 2012. By deploying fog layer(s) between the cloud data center and the edge devices, it addresses several QoS requirements that traditional computing struggles to satisfy, such as latency, bandwidth, mobility, geo-awareness, heterogeneity, etc. [12].

Only a few studies have been conducted to improve traffic efficiency through ITSC strategies using Fog Computing and DRL. A fog computing-based traffic signal control strategy was proposed in [13]. They offer a single intersection control architecture in which the local fog node handles phase timing, and the centralized cloud handles regional optimization. Nevertheless, the multi-intersection ITSC strategy using Fog Computing and Deep Reinforcement Learning needs to be further investigated.

III. Proposed Framework

A. Fog Computing

For a large region with tens to hundreds of signalized intersections, simply stacking the observations of multiple intersections is infeasible due to the curse of dimensionality. Instead, the region of interest can be divided into several sub-regions, each containing a certain reasonable number of traffic signals (the clustering principle is not in the scope of this study). The traffic signals of one sub-region are coordinated by a local parent node, i.e., the fog node. The fog node receives the data collected by the edge devices deployed near each traffic signal and makes the signaling decision accordingly for all the traffic signals it controls. The edge devices then send control signals to the corresponding traffic lights.

In case of road network changes, such as the addition or removal of roads or traffic signals, the fog node agent will need to re-train its model if it has sufficient resources. Otherwise, it will submit a training request along with the necessary data to higher layers that have more resources. When the training is complete, it will be redeployed to the original fog node. The data needed for the re-training include the new network and recent traffic flow data or summarized representative traffic characteristics. The overall architecture is depicted in the left part of Fig. 1.

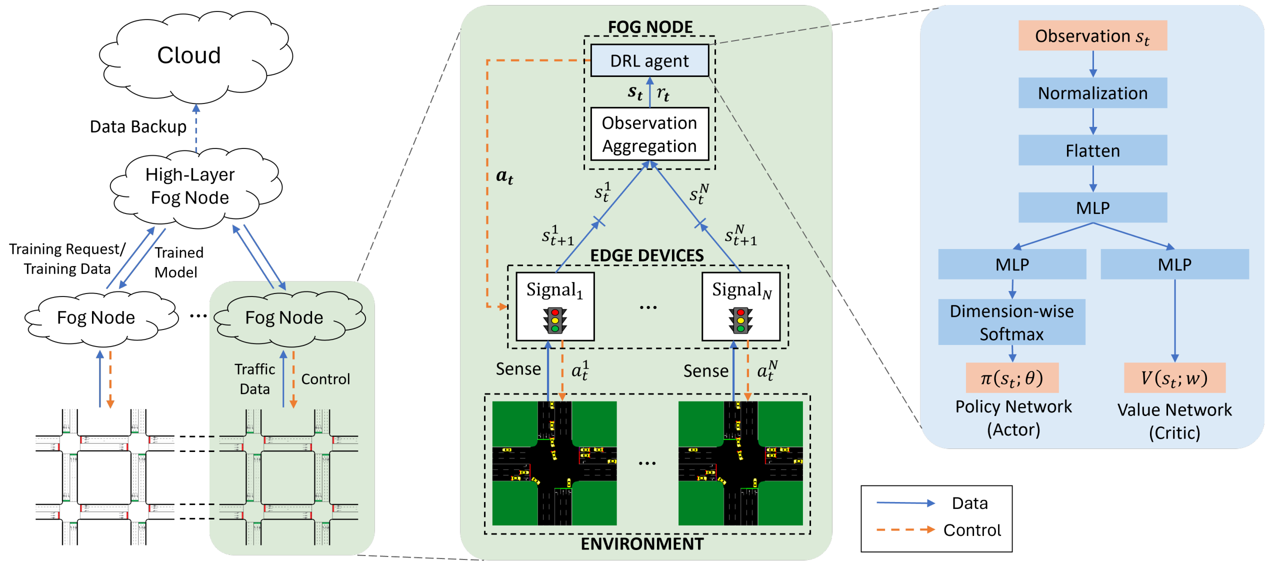


Fig. 1. The overall architecture of the proposed framework. **Left**: the fog computing architecture. **Middle**: The interaction between the RL agent and the environment. **Right**: the neural network architecture of the RL agent

B. Traffic Signal Control Optimization Using Reinforcement Learning

In this sub-section, we describe the traffic signal control within one cluster by formalizing it into an RL problem. The interaction between an RL agent and the environment is shown in the middle part of Fig 1.

1) Observation: The observation space is a matrix consisting of N rows, where N is the number of signalized intersections that the agent controls. Each row corresponds to the observation of one traffic light. The number of columns depends on the number of lanes controlled by the signalized intersection.

The observation of one traffic signal is composed of the following components:

- The number of vehicles in the sensing range of the intersection. We conducted a controlled pilot experiment to compare the performance differences between using the number of detected vehicles and the more commonly used queue length. As shown in Fig. 2, with all other variables kept the same, simply using the number of vehicles yields a higher cumulative reward. Combining both features does not provide extra benefits either.
- The one-hot encoding of the index of the current phase.
- The time that the current signal has elapsed in seconds.

For intersections with fewer lanes or phases, the corresponding segment of the observation vector can be zero-padded. To stabilize the training, each feature is standardized to have zero mean and unit variance.

2) Action: To ensure a fast response speed, we allow the agents to make decisions every second. At each time step,

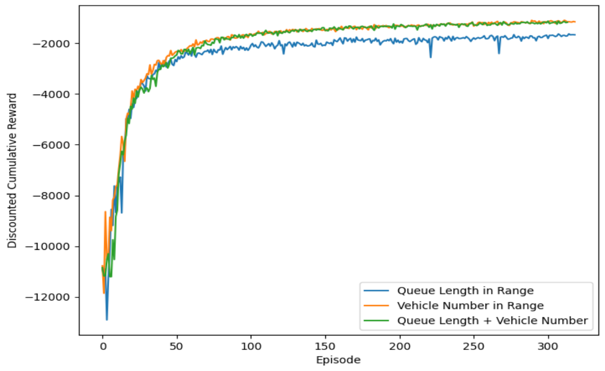


Fig. 2. Preliminary comparison between two features

the agent will choose between keeping the current phase, switching to the subsequent phase in the cycle, and skipping the next phase. More formally, for a cluster with N signalized intersections, the action space is defined as

$$\begin{aligned}\boldsymbol{A} &:= \{0,1,2\}^N \\ &= \{(a_1,a_2,\dots,a_N) \mid a_i \in \{0,1,2\}, \forall i = 1,2,\dots,N\}\end{aligned}$$

For the i-th traffic signal T_i, $a_i = 0$ denotes keeping the current phase for one more time step, $a_i = 1$ denotes switching to the next non-yellow phase, and $a_i = 2$ denotes skipping the next phase. For instance, in a cluster with five traffic signals, an action vector $\boldsymbol{a} = [0,1,0,0,2]$ indicates that traffic signal T_2 will switch to the next phase in its corresponding cycle, and traffic signal T_5 will skip the next phase and switch directly to the one after. All other traffic signals will remain in their current phases.

We also apply action masks for two reasons. First, by default, a vehicle is considered "waiting" when its speed is below 0.1m/s. We observed that the agent might exploit this by "flickering" the signal to keep the vehicles moving just above the speed threshold, thereby reducing the waiting time penalty. This phenomenon was also pointed out in [7]. To prevent this, action masks are applied to enforce minimum and maximum durations for each phase. When the current elapsed time is below the corresponding minimum duration, actions 1 and 2 are marked invalid; if it has reached the maximum duration, action 0 is marked invalid. An obvious and straightforward solution is to simply use longer time steps, such as 3 seconds, but it compromises fine control and responsiveness. Another merit of action masking is that it significantly reduces the exploration space [14].

Since the action space grows exponentially with the number of traffic signals, an issue known as the curse of dimensionality, we factorize the action space into separate dimensions, each corresponding to one intersection. In other words, the output layer of the policy network consists of $3N$ neurons instead of 3^N, significantly reducing the network complexity and improving scalability [15].

3) Reward: The reward r is structured as a penalty, specifically defined as

$$r = w_1 V + w_2 B + w_3 T, w_1, w_2, w_3 \leq 0$$

where V is the number of waiting vehicles; B is the number of vehicles that performed emergency braking; T is the number of vehicles teleported by the simulator due to excessively long wait;w_1, w_2, and w_3 are hyperparameters. This ensures that the agent is incentivized to minimize traffic congestion by reducing the number of waiting vehicles and, consequently, the overall waiting time. In this study, we use $w_1 = -1, w_2 = w_3 = -100$.

4) DRL Algorithm: Proximal Policy Optimization (PPO) [16] is used as our training algorithm. It consists of two trainable neural networks: the policy network with parameters θ denoted by $\pi(s;\theta)$ and the value network with parameters ω denoted by $V(s;\omega)$. The right part of Fig. 1 illustrates the network architecture of our PPO algorithm. The policy network and the value network share the initial layers for feature embedding. In addition, they each have their own hidden layers and output layers to accommodate their specific tasks and output dimensions.

We use the same objective function as [16]:

$$L_t = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{1}$$

where L_t^{CLIP} is the surrogate objective function. L_t^{VF} is the loss of the current value network V. $S[\pi_\theta](s_t)$ is the entropy of the current policy π_θ. c_1 and c_2 are coefficients. We use gradient ascent to train the policy.

Algorithm 1 shows the pseudocode of the overall training process.

Algorithm 1 Train Fog Agent

```
$T$: batch size
Initialize policy parameters $\theta$ and value function parameters $\omega$
for $iteration = 1, 2, ...$ do
    for $i = 1, 2, ..., T$ do
        Collect and aggregate observations to form $s_t$
        Compute $r_t$ from $s_t$
        Take action $a_t$ according to $\pi_\theta(.|s_t)$
    end for
    Compute loss function $L_t$ using Eq. (1)
    $\theta \leftarrow \theta + \alpha \frac{\partial L_t}{\partial \theta}$
    $\omega \leftarrow \omega - \alpha \frac{\partial L_t}{\partial \omega}$
end for
```

IV. Evaluation

In this section, we describe the experiments conducted and present the results to quantitatively evaluate the proposed method against baseline methods, including state-of-the-art methods.

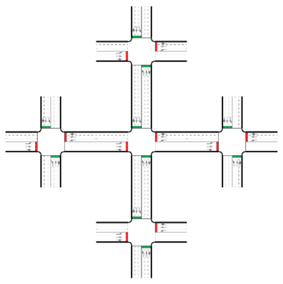
A. Simulation Implementation

The traffic simulation tool we use is **S**imulation of **U**rban **MO**bility (SUMO), a popular open-source microscopic traffic simulation software [17]. Its **Tra**ffic **C**ontrol **I**nterface (TraCI) is used to retrieve traffic data in real time and control traffic signals according to the agent's commands. The first two layers of the PPO model are shared between the policy network and value network, with 512 and 128 neurons, respectively. Subsequently, each network has a separate hidden dense layer with 64 neurons. Each hidden dense layer is followed by a `tanh` activation layer.

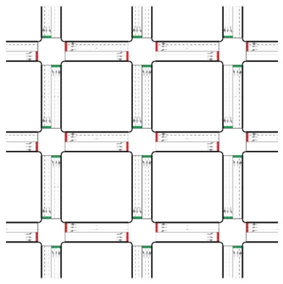
B. Evaluation Scenarios

We test all methods both in synthetic and real-world scenarios. Three synthetic road networks are created, depicted in Fig. 3. They consist of 5, 9, and 18 signalized four-directional intersections, respectively. for the first two scenarios with 5 and 9 signals each, all signals belong to the same cluster. In the third scenario, the network is divided into two clusters of 9 intersections, each controlled by a separate fog node, denoted by the dashed squares in Fig. 3c. To reflect the urban traffic dynamics, fringe edges are assigned with changing probabilities that vehicles depart and leave the map at each edge. For instance, in Fig. 3a, vehicles are twice more likely to enter the road network from the north for time steps $0 \dots 999$ and twice more likely to enter from the south for time steps $1000 \dots 1999$. Each experiment was repeated 10 times, and we calculated their average.

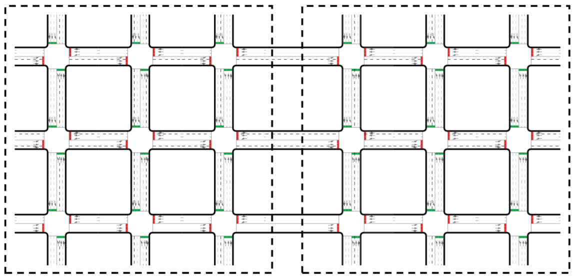
For the real-world scenario, we use the well-established SUMO scenario, namely "TAPAS Cologne" [18], which describes the traffic within the city of Cologne, Germany. The road network is shown in Fig. 4.



(a) 5-Intersection Network



(b) 9-Intersection Network



(c) 18-Intersection Network

Fig. 3. Synthetic Road Networks

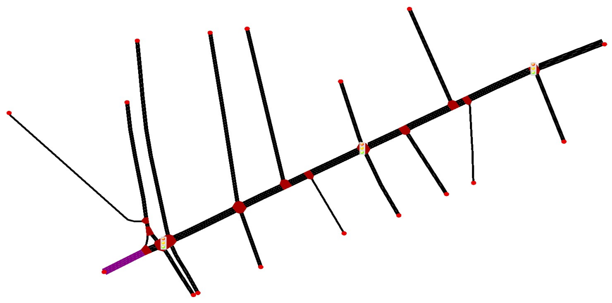
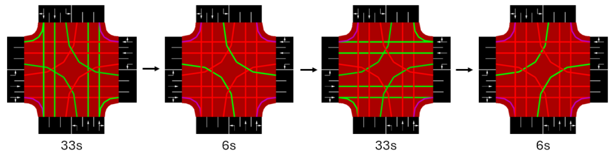


Fig. 4. Road network in Cologne, Germany [18]

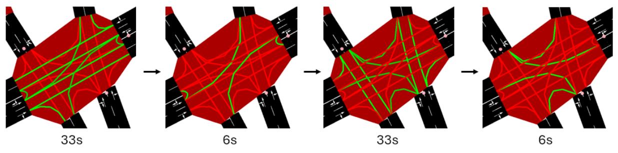
C. Baselines

The following four models are included for comparison:

- **Fixed Time Control (FTC)**.
- **Max-Pressure Control**. A greedy actuated method that assigns green signals to movements with the highest pressure.
- **MPLight** [4]. A well-received and highly scalable Deep Q-Network (DQN)-based method with the addition of pressure in observation and reward. All agents share the same neural network parameters. It uses the pressure of traffic movements as both the observation and the reward.
- **Independent DQN (IDQN)** [19]. Each traffic signal is controlled by a separate Deep Q-Network agent. The observation space consists of the current phase, the number and average speed of approaching vehicles, the number of waiting vehicles, and their accumulated waiting time. According to [20], it outperforms all other studied methods.



(a) Synthetic scenarios



(b) Real-world scenario

Fig. 5. Phase scheduling of FTC

The FTC's phases and their corresponding durations are shown in Fig 5. The remaining models are implemented based on RESCO [20], an all-in-one implementation of multiple signal control models. We use the default 10 seconds as the value of Δt for IDQN and MPLight.

D. Evaluation Metrics

The studied models are evaluated in the following metrics:

- **Average Waiting Time**. The average total time of each vehicle when it is driving under 0.1 m/s.
- **Average Time Loss**. The time lost due to driving below ideal speed, including slowdowns due to intersections.

E. Experimental Results

The experimental results in average waiting time and average time loss are shown in TABLE I and TABLE II, respectively. In the parentheses are the percentage of reduction of FogLight compared to the corresponding value. The average waiting time and average time loss of the experiments conducted in the Cologne scenario are shown in TABLE III. Our framework is referred to as "FogLight".

With the availability of broader collective data and faster response speed, our method yields noticeably lower average waiting times and average time loss than the baseline methods in both synthetic and real-world scenarios. Among the baseline methods, the closest performances come from IDQN, which tails by less than three seconds in most cases because the absolute improvement becomes less significant as the performance approaches the theoretical minimum in demanding scenarios, in which case the percentage improvement is more representative.

V. Conclusions and Future Work

In this study, we have presented a DRL-based traffic signal control framework that leverages the fog computing architecture, and the simulation results indicate that the proposed

TABLE I
AVERAGE WAITING TIMES IN SYNTHETIC SCENARIOS

Model	**5-inter**	**9-inter**	**18-inter**
FTC	26.0 (83.5%)	36.7 (83.1%)	60.7 (88.3%)
Max-Pressure	9.0 (52.2%)	10.8 (42.6%)	17.3 (59.0%)
IDQN	5.8 (25.9%)	7.3 (15.1%)	8.9 (20.2%)
MPLight	8.3 (48.2%)	10.0 (38.0%)	14.9 (52.4%)
FogLight	**4.3**	**6.2**	**7.1**

TABLE II
AVERAGE TIME LOSSES IN SYNTHETIC SCENARIOS

Model	**5-inter**	**9-inter**	**18-inter**
FTC	43.7 (56.8%)	56.0 (54.6%)	85.5 (60.0%)
Max-Pressure	25.1 (24.7%)	29.6 (14.2%)	44.9 (23.8%)
IDQN	20.6 (8.3%)	25.5 (0.4%)	34.8 (1.7%)
MPLight	24.1 (21.6%)	29.4 (13.6%)	43.4 (21.2%)
FogLight	**18.9**	**25.4**	**34.2**

model outperforms the baseline models in terms of the average waiting times and time losses, which implies its efficacy. It is also worth noting that the proposed method only relies on the number of vehicles in each lane and the current phase. With the prevalence of Vehicle-to-Infrastructure (V2I) technology, richer data such as routing, speed, acceleration, and waiting time will become easier and cheaper to obtain, which can further improve performance.

The one-size-fits-all baseline methods are unable to adapt to varying traffic situations (e.g., the orientation of arterial and side roads, a variety of traffic densities, and different numbers and connections of lanes). On the other hand, FogLight is scalable and can be re-trained for various road networks, observation and action shapes, and changes to road networks.

Currently, the clustering is manual. Therefore, finding an efficient and effective clustering algorithm, such as geo-based KNN, remains an interesting problem.

TABLE III
AVERAGE WAITING TIMES AND TIME LOSSES IN REAL-WORLD SCENARIO

Models	**Avg. Waiting Time (sec.)**	**Avg. Time Loss (sec.)**
FTC	45.9 (66.2%)	63.9 (52.0%)
Max-Pressure	34.6 (55.2%)	50.4 (39.1%)
IDQN	18.3 (15.3%)	33.1 (7.3%)
MPLight	21.0 (26.2%)	36.8 (16.6%)
FogLight	**15.5**	**30.7**

REFERENCES

[1] G. Weisbrod, D. Vary, and G. I. Treyz, "Economic implications of congestion," *NCHRP Report*, 2001.

[2] K. Zhang and S. Batterman, "Air pollution and health risks due to vehicle traffic," *Science of The Total Environment*, vol. 450-451, pp. 307–316, 4 2013.

[3] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in *Advances in dynamic network modeling in complex transportation systems*. Springer, 2013, pp. 27–66.

[4] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3414–3421.

[5] H. Wei, C. Chen, G. Zheng, K. Wu, V. Gayah, K. Xu, and 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*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1290–1298.

[6] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, "Stmarl: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2228–2242, 2022.

[7] E. V. der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, vol. 1, 2016.

[8] P. Y. J. Ha, S. Chen, R. Du, and S. Labi, "Scalable traffic signal controls using fog-cloud based multiagent reinforcement learning," *Computers*, vol. 11, no. 3, 2022.

[9] H. Ezawa and N. Mukai, "Adaptive traffic signal control based on vehicle route sharing by wireless communication," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2010, pp. 280–289.

[10] B. Zhou, J. Cao, and H. Wu, "Adaptive traffic light control of multiple intersections in wsn-based its," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2011, pp. 1–5.

[11] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, "Traffic signal control based on reinforcement learning with graph convolutional neural nets," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-November, pp. 877–883, 12 2018.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *MCC'12 - Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, pp. 13–15, 2012.

[13] C. Tang, S. Xia, C. Zhu, and X. Wei, "Phase timing optimization for smart traffic control based on fog computing," *IEEE Access*, vol. 7, pp. 84 217–84 228, 2019.

[14] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *The International FLAIRS Conference Proceedings*, vol. 35, May 2022.

[15] S. Sharma, A. S. Lakshminarayanan, and B. Ravindran, "Learning to repeat: Fine grained action repetition for deep reinforcement learning," in *International Conference on Learning Representations*, 2017.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[17] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[18] C. Varschen and P. Wagner, "Mikroskopische modellierung der personenverkehrsnachfrage auf basis von zeitverwendungstagebüchern," in *Integrierte Mikro-Simulation von Raum- und Verkehrsentwicklung. Theorie, Konzepte, Modelle, Praxis*, ser. Stadt Region Land, K. J. Beckmann, Ed., vol. 81. Institut für Stadtbauwesen und Stadtverkehr, RWTH Aachen, 2006, pp. 63–69.

[19] J. Ault, J. P. Hanna, and G. Sharon, "Learning an interpretable traffic signal control policy," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 88–96.

[20] J. Ault and G. Sharon, "Reinforcement learning benchmarks for traffic signal control," in *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*, December 2021.