Adaptive Traffic Signal Control Using Distributed MARL and Federated Learning

1st Tianyu Wang
Nanjing University of Science and Technology
Nanjing, China
tianyu.wang@njust.edu.cn

2nd Teng Liang
Nanjing University of Science and Technology
Nanjing, China
teng.liang@njust.edu.cn

3rd Jun Li
Nanjing University of Science and Technology
Nanjing, China
jun.li@njust.edu.cn

4th Weibin Zhang
Nanjing University of Science and Technology
Nanjing, China
weibin.zhang@njust.edu.cn

5th Yiji Zhang
Nanjing University of Science and Technology
Nanjing, China
yijin.zhang@gmail.com

6th Yan Lin
Nanjing University of Science and Technology
Nanjing, China
yanlin@njust.edu.cn

***Abstract*—Facing the increasingly serious traffic congestion problem, the existing traffic signal control technology has been unable to meet the demands of urban smart traffic construction. As one of the most promising methods, Reinforcement Learning (RL) has been widely exploited for solving the adaptive traffic signal control (ATSC) problem. However, centralized RL control mode is impractical due to its poor robustness and high computational complexity. In this paper, we propose two distributed Multi-agent Reinforcement Learning (MARL) control modes as well as a Federated Learning (FL) framework to solve the ATSC problem, where the former is based on Advantage Actor-Critic (A2C) algorithm and the latter is based on Federated Averaging (FedAvg) algorithm. By comparing in a small grid of traffic network, the experimental results reveal that our proposed algorithms outperform the centralized solution in terms of both the optimality and the learning efficiency.**

***Index Terms*—Adaptive Traffic Signal Control, Distributed Multi-agent Reinforcement Learning, Federated Learning**

I. Introduction

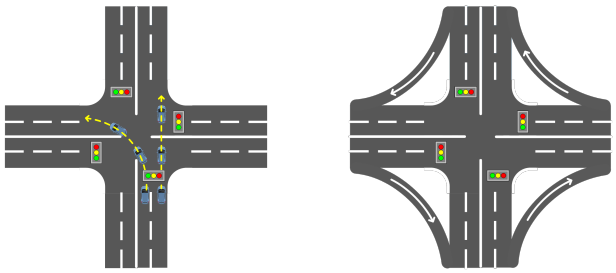
With the acceleration of urbanization, traffic congestion has become a major problem in most cities around the world, which has caused significant negative impacts on the environment, society and economy. According to British statistics, in a city with more than a hundred of intersections, the average annual loss due to traffic delays at each intersection is 1.4 million pounds [1]. In China, the economic losses resulted from traffic congestion each year reach nearly 160 billion yuan, equivalent to 3.2% of national GDP. Since the Internet of Vehicles and driverless technology are not mature yet [2], the most economical and efficient way at present is to optimize the control of traffic signal to improve road patency rate.

Adaptive traffic signal control (ATSC) with Reinforcement Learning (RL) [3] is an emerging concept to compensate for the existing control technology [4]. Its biggest feature is to arrange the signal phase and time according to the road conditions, replacing the scheme of fixed phase sequence. On the other hand, unlike the methods based on mathematical models, RL does not require over-idealized assumptions and mathematical formulas. Therefore, applying RL under the Markov Decision Process (MDP) framework is a viable means to solve ATSC problem in the context of various traffic conditions [5].

In the past, Q-learning, as a value-based and off-policy method, was widely adopted in this research. For instance, system where agent at each traffic intersection could control the signals through Q-learning-based Single Agent Reinforcement Learning (SARL) was designed and tested in Toronto city, which proved to reduce the average vehicle congestion time by 10% [6]. Also, a Q-learning-based Multi-Agent Reinforcement Learning (MARL) solution was proposed to solve the multi-intersection traffic signal control on arterial traffic [7]. However, a recent research has supported that Actor-Critic method has advantages over Q-learning cause it can tolerate more state and action space which is suitable for endless MDP problems [8]. Meanwhile, motivated by the research on networked MARL in continuous spaces [9], [10], we propose two distributed ATSC schemes after combining the Advantage Actor-Critic (A2C) algorithm with MARL.

Additionally, a novel deep reinforcement learning framework namely Federated Learning (FL) which was established by Google in 2019 for distributed learning [11], [12]. It has lower requirements for data sharing, especially real-time communication, which gave us a new inspiration in neural network training. In this paper, we try to apply the FL to the distributed MARL while dealing with the ATSC problem.

978-1-7281-8141-7/20/$31.00 ©2020 IEEE



(a) (b)

Figure 1. Sketch of traffic intersection

II. System Model

This section gives a standard scenario to describe traffic intersections. Then the MDP method is applied to build a mathematical model of the complex and indescribable traffic dynamics.

A. Traffic Intersection Modeling

At present, the traffic intersections in most cities adopt a configuration of three incoming lanes. For unifying the system model, we consider the intersection shown in Figure 1(a) including only the straight lanes and left-turn lanes. Note that, in this paper, the right-turn lanes of each entrance are not taken into consideration in our model. The main reasons are as follows:

(1) Most urban intersections have no signal restrictions on right-turn lanes in the countries where the vehicle has a left rudder [13].

(2) Some cities in China adopt the "Separate Right-turn Lane" design mode in Figure1(b). The right-turn lane is separated from the main road about 50m before the intersection [14].

B. MDP Setting

In order to apply reinforcement learning to the practical environment, it is the premise to describe the dynamic changes of the traffic environment with the aids of the MDP which contains action, state and reward.

1) Action definition: In the ATSC problem, the use of a fixed sequence of signal phases is outdated because it can no longer fit for the adaptive control in the dynamic situation. In this paper, we set the phase duration to a short interval and free the phase selection form the limitation of the fixed sequence. Therefore, it is a reliable and convenient way to define the phase of the signal light as the action u_t in the MDP. Table I illustrates the four actions which agent can take.

TABLE I. Traffic Signal Phases

p_1	p_2	p_3	p_4
N ↕ S	W N S E	W ⇄ E	W N E S

When the phase is p_1, the lane in the north-south direction is open to allow vehicle's travelling while other lanes are still closed. Similarly, p_3 is for vehicle's travelling in the west-east direction. p_2 is for vehicle's travelling from north to east and south to west. Conversely, p_4 is for vehicles travelling from west to north and east to south.

2) State definition : We define the local state as

$$s_{t,i} = \{len_t[l], p_t\}_{l \in L_i}, \tag{1}$$

where $len_t[l]$ presents the queue length of each incoming lane l, L_i is the set of incoming lanes at intersection i, p_t is the current phase.

3) Reward definition : We define the step reward as

$$r_{t,i} = -\sum_{l \in L_i} (queue[l])/|L_i|, \tag{2}$$

where $queue[l]$ represents the amount of vehicles in each incoming lanes, $|L_i|$ represents the number of the elements in set L_i.

III. Problem Formulation

In the ATSC problem, agents are installed at intersections to observe the local traffic information and control the signals. Since the traffic flow at different intersections is interactive, it is necessary to coordinate the various agents in the network [15]. Centralized control mode is the most direct but the most complicated control method. However, as the number of agents increases, its computational complexity will increase exponentially. Distributed control mode tends to be practical. So in this section, we propose three decentralized control modes as well as the centralized control mode for comparison.

- Centralized RL based Control Mode
- Independent MARL based Control Mode
- Joint MARL based Control Mode
- FL based Control Mode

In a multi-agent environment, we can use the principle of graph theory to define a graph $G(\nu, \varepsilon)$ to represent a network composed of multiple agents. Notably, ν is the set of agents as each node, and ε is the set of edges between different nodes. Besides, N_i is defined as the set of associated nodes for agent i. Also, the minimum distance between agent i and agent j is described as $d_{i,j}$.

A. Centralized RL Control Mode

In this solution, a CC (Central Controller) is set to conduct RL and control all the agents in the system. As shown in Figure 2(a), the agent i at each intersection plays a role of the agent to observe the local traffic information and sends the data to the CC in time [16]. After sharing the global state in the trained model, CC will give the feedback of action selection to every agent.

Although this method seems to be comprehensive, it still has the following problems:

(1) Processing large amount of data would cause decision-making lag.

(2) If one or several agents accidentally lose communication, incorrect decision-making could occur. In brief, this method has high learning efficiency but poor robustness.

B. Distributed RL Control Mode

1) Independent MARL Control Mode: In this scheme, each agent performs RL independently on the assumption that it can obtain global information. Under the conditions of cooperation, the overall system could be optimized when agent $i \in \nu$ adjusts its own policy to maximize the global reward [17]. As shown in Figure 2(b), each agent uploads the local information to the server. Then, the server integrates the data and then broadcasts it to each agent to share the global information which is used in their learning process.

The distributed RL of each agent effectively reduces the calculation cost of the central processor. Nevertheless, the requirement of sharing global information in this system is still necessary which is challenging for the constantly changing traffic situation.

2) Joint MARL Control Mode: Figure 2(c) presents the mechanism of the Joint MARL Control Mode among multiple agents. The agent i exchanges information (including state, step reward, etc.) with related nodes $\{j|j \in N_i\}$ within a certain range. At the same time, we can also weight this information to imply its degree of influence with space, which will be discussed in detail in section 4. To sum up, the method not only reduces the communication volume, but also reduces the computational cost.

3) Federated Learning Control Mode: FL allows agents to collaboratively train models without data sharing. In essence, the main task of the FL system is the interaction and optimization of agents' neural network parameters[18]. Unlike the traditional distributed learning system, FL does not have high requirements for real-time communication. As shown in Figure 2(d), agents $\{i|i \in \nu\}$ perform RL locally without intercommunication. They would upload their model parameters when free and the server collects and processes these model parameters to generate new parameters which will be broadcast to agents for updating.

TABLE II. THE COMPARISON OF CONTROL MODES

control mode	real-time requirement	calculation expense	robustness
Centralized RL Control Mode	high	high	poor
Independent MARL Control Mode	high	high	poor
Joint MARL Control Mode	high	medium	general
Federated Learning Control Mode	low	low	good

IV. PROPOSED METHODS

In order to adopt RL methods in the above ATSC problem, we consider the following algorithms:

- Centralized A2C Algorithm(CA2C)
- Independent A2C Algorithm(IA2C)
- Joint A2C Algorithm(JA2C)
- Federated Learning-A2C Algorithm

In this section, we briefly introduce the principle of A2C algorithm at the beginning. Accordingly, the approach to realizing the proposed algorithms is discussed in detail. Furthermore, the procedure of FL algorithm in ATSC problem is demonstrated additionally.

A. Centralized A2C Algorithm

The A2C algorithm not only has the characteristics of Q-learning (based on state value learning), but also has the features of Policy Gradient (based on the probability of action selection).

Suppose the agent has a small set of experiences $B = \{(s_t, u_t, s'_t, r_t)\}$ through sampling, so the original value function is

$$\hat{R}_t = \sum_{\tau=t}^{t_B-1} \gamma^{\tau-t} r_\tau, \tag{3}$$

where t_B is the last step of this batch, r is the single step reward, γ is the discount factor which can also be regarded as learning rate.

Based on the AC algorithm, the A2C algorithm introduces a value benchmark V_w to estimate the expected return $\mathbb{E}\left[R_t^\pi | s_t = s\right]$. Therefore, Eq 3 is revised to

$$\tilde{R}_t = \hat{R}_t + \gamma^{t_B-t} V_w\left(s_{t_B}\right). \tag{4}$$

Also, the Advantage function is defined as $A_t = \tilde{R}_t - V_w\left(s_t\right)$ which is used in updating θ and w which are neural network parameters of Actor and Critic respectively. Clearly, the advantage function can effectively reduce the variance between different returns during sampling which shows a better performance.

It is necessary for CC to collect the information of which global state $\mathbf{s}_{t,\nu}$ is defined as a multi-dimensional vector $\mathbf{s}_{t,\nu} = (s_{t,1}, \cdots, s_{t,i}, \cdots, s_{t,N})$, containing the states obtained by each agent at timestamp t. Similarly, the corresponding action space $\mathbf{u}_{t,\nu} = (u_{t,1}, \cdots, u_{t,i}, \cdots, u_{t,N})$ is also a multi-dimensional vector. Meanwhile, we assume that the global reward can be decomposed into the sum of the single-step rewards of each agent. The global reward can be illustrated as

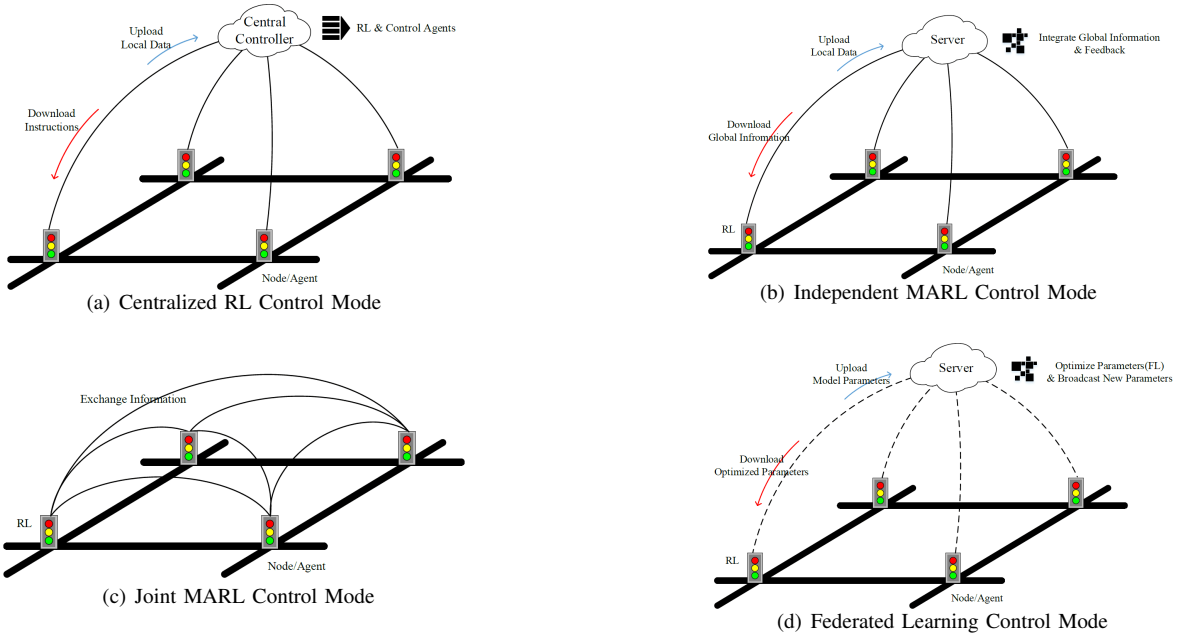
$$r_t = \sum_{i \in \nu} r_{t,i}. \tag{5}$$

Then we have the common global value function for each return as

$$\hat{R}_{t,\nu} = \sum_{i \in \nu} \sum_{\tau=t}^{t_B-1} \gamma^{\tau-t} r_\tau, \tag{6}$$

where, t_B is the last step in minibatch. However, according to A2C algorithm, a value regressor V_w need to be introduced to estimate it and the global return changes into

$$\tilde{R}_{t,\nu} = \hat{R}_{t,\nu} + \gamma^{t_B-t} V_w\left(s_{t_B,\nu} | \pi_\theta\right). \tag{7}$$



(a) Centralized RL Control Mode

(b) Independent MARL Control Mode

(c) Joint MARL Control Mode

(d) Federated Learning Control Mode

Figure 2. Diagrams of different control modes

As we input the global state and return into neural network, the value loss function is defined as

$$\mathcal{L}(w) = \frac{1}{2|B|}\sum_{t=0}^{t_B}\left(\tilde{R}_{t,\nu} - V_w\left(s_{t_B,\nu}|\pi_\theta\right)\right)^2. \tag{8}$$

Similarly, the policy loss function is

$$\mathcal{L}(\theta) = -\frac{1}{|B|}\sum_{t=0}^{t_B}\log\pi_\theta\left(u_{t,\nu}|s_{t,\nu}\right)A_{t,\nu}, \tag{9}$$

where $A_{t,\nu} = \tilde{R}_{t,\nu} - V_w\left(s_{t_B,\nu}|\pi_\theta\right)$.

B. Independent A2C Algorithm

IA2C algorithm can be regarded as a derivative of CA2C in which global information is also required. For agent i, the input state information has not changed compared with the CA2C algorithm which is still a multi-dimensional matrix $\mathbf{s}_{t,i} = \mathbf{s}_{t,\nu}$ containing all nodes' state information. Whereas, the action space $\mathbf{u}_{t,i}$ is limited to the number of selectable actions in local. Each agent optimizes its own action policy by observing the global state and estimating global return as

$$\hat{R}_{t,i} = \sum_{i\in\nu_i}\sum_{\tau=t}^{t_B-1}\gamma^{\tau-t}r_\tau, \tag{10}$$

$$\tilde{R}_{t,i} = \hat{R}_{t,i} + \gamma^{t_B-t}V_w\left(s_{t_B,i}|\pi_\theta\right). \tag{11}$$

Then the value loss function can be given as

$$\mathcal{L}(w) = \frac{1}{2|B|}\sum_{t=0}^{t_B}\left(\tilde{R}_{t,i} - V_w\left(s_{t_B,i}|\pi_\theta\right)\right)^2. \tag{12}$$

Equally, the policy loss function becomes

$$\mathcal{L}(\theta) = -\frac{1}{|B|}\sum_{t=0}^{t_B}\log\pi_\theta\left(u_{t,i}|s_{t,i}\right)A_{t,i}, \tag{13}$$

where $A_{t,i} = \tilde{R}_{t,i} - V_w\left(s_{t_B,i}|\pi_\theta\right)$.

Algorithm 1: Independent A2C Algorithm

Parameters: $\gamma, T, |B|, \lambda^w, \lambda^\theta$
Result: $\{w_i\}_{i\in\nu}, \{\theta_i\}_{i\in\nu}$
initialize $s_0, \pi_{-1}, t \leftarrow 0, B = \emptyset$
repeat
  /*Agent*/
  for $i \in \nu$ **do**
    obtain $s_{t,i}$ from server
    choose $u_{t,i}$ from $\pi_{t,i}$
    observe $\tilde{r}_{t,i}$, $s_{t+1,i}$ and upload data
  /*Server*/
  collect data and generate $s_{t,i} \leftarrow s_{t,\nu}$
  $B \leftarrow B \cup \{(t, s_{t,i}, \pi_{t,i}, u_{t,i}, \tilde{r}_{t,i}, s_{t+1,i})\}_{i\in\nu}$
  $t \leftarrow t+1$
  if $t = T$ **then**
    for $i \in \nu$ **do**
      estimate $\hat{R}_{\tau,i}, \tilde{R}_{\tau,i} \forall \tau \in B$
      update w_i with $\lambda^w \nabla\mathcal{L}(w_i)$
      update θ_i with $\lambda^\theta \nabla\mathcal{L}(\theta_i)$
    reset $s_0, \pi_{-1}, t \leftarrow 0, B \leftarrow \emptyset$
until *Stop condition reached*;

C. Joint A2C Algorithm

The JA2C algorithm we proposed aims to conduct MARL through small-scale clustering to make the convergence more stable. In other words, each agent communicates with its associated nodes in a certain grid, shares information and performs RL using the local neural network.

A spatial discount factor $\beta \in [0,1]$ is added to the JA2C algorithm to trade off the influence of the associated nodes $\{j|j \in N_i\}$ of agent i which varies with distance. So the total reward for agent i need to be adjusted as

$$\tilde{r}_{t,i} = \sum_{d=0}^{D_i} \left(\sum_{j \in N_i | d(i,j)=d} \beta^d r_{t,j} \right) + r_{t,i}, \tag{14}$$

where D_i is the maximum distance between agent i and the element which belongs to set N_i. Accordingly, the total return becomes

$$\hat{R}_{t,i} = \sum_{\tau=t}^{t_B-1} \gamma^{\tau-t} \tilde{r}_\tau, \tag{15}$$

$$\tilde{R}_{t,i} = \hat{R}_{t,i} + \gamma^{t_B-t} V_{w_i}\left(\tilde{s}_{t_B,i}|\pi_\theta\right). \tag{16}$$

Moreover, we continue to use β to discount the state of the neighbor node, the agent's state is expressed as

$$\tilde{s}_{t,i} = [s_{t,i}] \cup \beta[s_{t,j}]_{j \in N_i}. \tag{17}$$

In this case, the value loss function becomes

$$\mathcal{L}(w_i) = \frac{1}{2|B|} \sum_{t=0}^{t_B} \left(\tilde{R}_{t,i} - V_{w,i}\left(\tilde{s}_{t_B,i}|\pi_\theta\right) \right)^2. \tag{18}$$

The policy loss function becomes

$$\mathcal{L}(\theta) = -\frac{1}{|B|} \sum_{t=0}^{t_B} \log \pi_{\theta_i}\left(u_{t,i}|\tilde{s}_{t,i}\right) A_{t,i}, \tag{19}$$

where $A_{t,i} = \tilde{R}_{t,i} - V_{w,i}\left(\tilde{s}_{t_B,i}|\pi_\theta\right)$.

Algorithm 2: Joint A2C Algorithm

Parameters: $\beta, \gamma, T, |B|, \lambda^w, \lambda^\theta$
Result: $\{w_i\}_{i\in\nu}, \{\theta_i\}_{i\in\nu}$
initialize $s_0, \pi_{-1}, t \leftarrow 0, B = \emptyset$
repeat
  for $i \in \nu$ **do**
    choose $u_{t,i}$ from $\pi_{t,i}$
    observe $\tilde{r}_{t,i}$ and $\tilde{s}_{t,i}$
  $B \leftarrow B \cup \{(t, \tilde{s}_{t,i}, \pi_{t,i}, u_{t,i}, \tilde{r}_{t,i}, \tilde{s}_{t+1,i})\}_{i\in\nu}$
  $t \leftarrow t+1$
  if $t = T$ **then**
    for $i \in \nu$ **do**
      estimate $\hat{R}_{\tau,i}, \tilde{R}_{\tau,i} \forall \tau \in B$
      update w_i with $\lambda^w \nabla \mathcal{L}(w_i)$
      update θ_i with $\lambda^\theta \nabla \mathcal{L}(\theta_i)$
    reset $s_0, \pi_{-1}, t \leftarrow 0, B \leftarrow \emptyset$
until *Stop condition reached*;

D. Federated Learning-A2C Algorithm

The Federated Learning-Advantage Actor-Critic algorithm we proposed is based on FedAvg algorithm, which is the most widely used FL algorithm. Agents distributed in the system use their own sampled data for model training in parallel, and the server averages the model parameters of each agent to optimize. In particular, this algorithm is not only proved to have good convergence on mathematical principles[19], but also shows good convex optimization in practical applications.

Assuming there are M agents, the server is responsible to receive all agents' neural network parameters and process the data as follows.

Algorithm 3: FL-A2C Algorithm

Parameters: $\beta, \gamma, t, episode, T, E, |B|, \lambda^w, \lambda^\theta$
Result: $\{w_i\}_{i\in\nu}, \{\theta_i\}_{i\in\nu}$
Initialize $s_0, \pi_{-1}, t \leftarrow 0, B = \emptyset$
repeat
  for $i \in \nu$ **do**
    choose $u_{t,i}$ from $\pi_{t,i}$
    observe $\tilde{r}_{t,i}$ and $\tilde{s}_{t,i}$
  $B \leftarrow B \cup \{(t, \tilde{s}_{t,i}, \pi_{t,i}, u_{t,i}, \tilde{r}_{t,i}, \tilde{s}_{t+1,i})\}_{i\in\nu}$
  $t \leftarrow t+1$
  if $t = T$ **then**
    for $i \in \nu$ **do**
      estimate $\hat{R}_{\tau,i}, \tilde{R}_{\tau,i} \forall \tau \in B$
      update w_i with $\lambda^w \nabla \mathcal{L}(w_i)$
      update θ_i with $\lambda^\theta \nabla \mathcal{L}(\theta_i)$
    reset $s_0, \pi_{-1}, t \leftarrow 0, B \leftarrow \emptyset$
  $episode \leftarrow episode + 1$
  if $episode = E$ **then**
    update $\mathbf{w_i} \leftarrow \sum_{k=1}^{N} p_k \cdot \mathbf{w}_k$
    update $\theta_\mathbf{i} \leftarrow \sum_{k=1}^{N} p_k \cdot \theta_k$
    $episode \leftarrow 0$
until *Stop condition reached*;

where $\mathbf{w}_k, \boldsymbol{\theta}_k$ is the tensors of the Actor and Critic neural networks' parameter of the k^{th} agent. p_k is the weight of the k^{th} agent, satisfying $p_k \geq 0$ and $\sum_{k=1}^{M} p_k = 1$.

V. Simulation and Result

In this section, a 2×2 traffic network is built to simulate the above algorithm as shown in Figure 3. According to the idea of black box test, the specific measures taken by each agent are not necessary to figure out, while the average episode reward per agent and the average episode waiting time per vehicle are significant indicators to prove their capability.

A. Experiment Settings

Random traffic flows are set to flow in from the external ports of the network $x1 - x8$, and the vehicles inside the network move followed by the ATSC algorithm. During the

experiments, the initial number of vehicles inside the grid and the traffic flow setting are kept same among different groups.

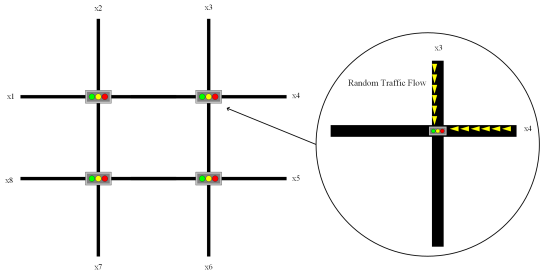


Figure 3. A traffic grid of 4 intersections

As for each intersection, we assume that the agent can observe the environment within 50m of each incoming lane[20]. Actually, we regard the 50m long road as the maximum queue length and divide it into ten units(Δl). Besides, the signal light will maintain a unit time $\Delta t = 12s$ after selecting an action. When the green light is on, four units ($4\Delta l$)of vehicles on that lane are permitted to pass the intersection during the phase duration.

B. Result Analysis

We set up two sets of control groups, in which CA2C is used to test the performance of IA2C and JA2C. On the other hand, in order to compare the performance before and after FL optimization, Fully distributed A2C is applied in four intersections to control their local traffic, which means agent is isolated to perform SARL without any communication.

During the simulation, algorithms are trained in 1,000 episodes and 50 steps ($|B| = 50$) in each episode. Moreover, we built two neural networks to implement the above algorithms, and both neural networks have one hidden layer containing 20 neural units, which all use ReLU as the activation function. The output layer for the actor network is softmax, and that for the critic network is linear.

In addition, we set $\lambda^w = 0.01, \lambda^\theta = 0.001, \gamma = 0.8$ and $\beta = 0.2$ particularly in JA2C. For the implementation of the FedAvg algorithm, the neural network parameters are averaged and updated every 200 episodes.

Figure 4 and 5 plot the curves of the average episode reward (per agent) and the average episode delay (per intersection) of these algorithms and the convergence values of three MARL algorithms are listed in Table III.

TABLE III. THE DATA OF SIMULATION RESULT

Data	Average Episode Reward [Δl]	Average Episode Delay [Δt]
Centralized A2C	-3.5	3.8
Joint A2C	-3.6	3.7
Independent A2C	-4.3	5.1
FL	-4.7	8.7
Fully distributed A2C	-5.5	10.5

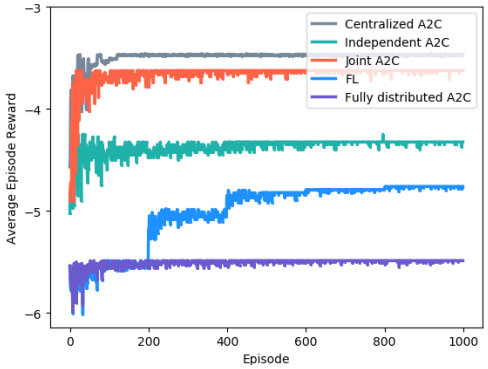


Figure 4. Comparison of average episode reward

By visualizing the characteristics of each algorithm, their performance could be compared more directly. First of all, CA2C's average episode reward arrives at a high and stable value, and the average episode delay oscillates around 4. Likewise, MA2C's average episode reward can also achieve a good effect, but its value will still fluctuate slightly at -3.6. On the other hand, the average episode delay of IA2C is slightly higher than 5 which indicates it could not realize the optimization as well as other methods. The average episode reward of the FL system experiences a great lift from -5.5 to -4.7 after optimization.

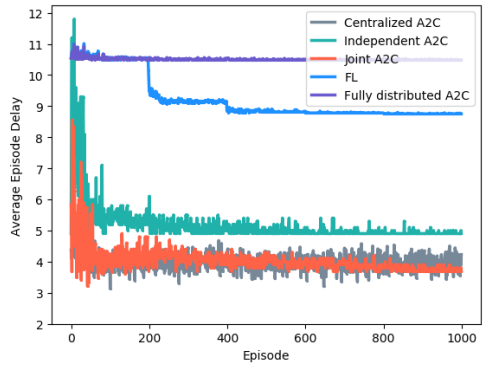


Figure 5. Comparison of average episode delay

In conclusion, the experimental results reveal that in a small-scale area, the performance of JA2C is close to that of CA2C, which is more competitive than IA2C. Also, FedAvg algorithm can effectively enhance the capability of the system.

VI. CONCLUSION

This paper has proposed two A2C-based MARL algorithms and a FL-based SARL algorithm to solve ATSC problems. Several conclusions could be drawn after the experiments which compare the convergence and capability of each algorithm. First of all, JA2C is suggested to work in the environment of dense traffic network and good communication system, compared to IA2C or CA2C whose communication cost for data sharing is extremely high. By comparison, FL

is an appropriate method in sparse traffic network and poor communication system.

Our future works plan to deploy our scheme in the real urban transportation. 1)Real-world traffic flow data are required for training and verifying our algorithms. 2)Communication delay and system robustness should be further considered in our proposed algorithms.

REFERENCES

[1] Y. H. Chen, C. J. Chang, and C. Y. Huang, "Fuzzy q-learning admission control for WCDMA/WLAN heterogeneous networks with multimedia traffic," *IEEE Transactions on Mobile Computing*, vol. 8, no. 11, pp. 1469–1479, 2009.

[2] J. Li, Z. Xing, W. Zhang, Y. Lin, and F. Shu, "Vehicle tracking in wireless sensor networks via deep reinforcement learning," *arXiv*, 2020.

[3] R. Sutton and A. Barto, *Reinforcement Learning:An Introduction*. MIT Press, 1998.

[4] C. Cai, C. K. Wong, and B. G. Heydecker, "Emerging technologies : Adaptive traffic signal control using approximate dynamic programming," 2009.

[5] J. D. Johnson, J. Li, and Z. Chen, "Reinforcement Learning: An Introduction," *Neurocomputing*, vol. 35, no. 1-4, pp. 205–206, 2000.

[6] 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, 2013.

[7] H. Wei, C. Chen, K. Wu, G. Zheng, Z. Yu, V. Gayah, and Z. Li, "Deep Reinforcement Learning for Traffic Signal Control along Arterials," 2019.

[8] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transportation Research Part C: Emerging Technologies*, vol. 85, no. September, pp. 732–752, 2017.

[9] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2020.

[10] K. Zhang, Z. Yang, and T. Basar, "Networked Multi-Agent Reinforcement Learning in Continuous Spaces," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2018-Decem, no. Cdc, pp. 2771–2776, 2019.

[11] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," 2019.

[12] C. Ma, J. Li, M. Ding, H. H. Yang, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, vol. PP, no. 99, pp. 1–7, 2020.

[13] Y. W. Lai, J. Rong, and X. M. Liu, "Capacity model of the right-turn lane at signalized intersection with channelized islands," *Journal of Beijing University of Technology*, vol. 38, no. 6, pp. 865–869, 2012.

[14] I. Potts, K. Bauer, D. Torbic, and J. Ringert, "Safety of channelized right-turn lanes for motor vehicles and pedestrians," *Transportation Research Record Journal of the Transportation Research Board*, vol. 2398, pp. 93–100, 2013.

[15] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for uav-mounted mobile edge computing with deep reinforcement learning," 2020.

[16] P. Wu, J. Li, L. Shi, M. Ding, K. Cai, and F. Yang, "Dynamic content update for wireless edge caching via deep reinforcement learning," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1773–1777, 2019.

[17] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative Multi-agent Control Using Deep Reinforcement Learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10642 LNAI, pp. 66–83, 2017.

[18] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," 2019.

[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," no. 2019, pp. 1–26, 2019. [Online]. Available: http://arxiv.org/abs/1907.02189

[20] J. Li, Z. Xing, W. Zhang, Y. Lin, and F. Shu, "Vehicle tracking in wireless sensor networks via deep reinforcement learning," *arXiv*, 2020.