Digital Transportation and Safety

ARTICLE

Open Access

https://doi.org/10.48130/DTS-2023-0012

Digital Transportation and Safety **2023**, 2(2):150–163

An integrated and cooperative architecture for multi-intersection traffic signal control

Qiang Wu[1], Jianqing Wu[2], Bojian Kang[3], Bo Du[4*], Jun Shen[5*] and Adriana Simona Mihăiţă[6]

[1] *Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chendu 610054, China*
[2] *School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[3] *School of Management, Lanzhou University, Lanzhou 730000, China*
[4] *SMART Infrastructure Facility, University of Wollongong, Wollongong 2522, Australia*
[5] *School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia*
[6] *Faculty of Engineering and IT, University of Technology Sydney, Sydney 2007, Australia*
* Corresponding authors, E-mail: bdu@uow.edu.au; jshen@uow.edu.au

Abstract

Traffic signal control (TSC) systems are one essential component in intelligent transport systems. However, relevant studies are usually independent of the urban traffic simulation environment, collaborative TSC algorithms and traffic signal communication. In this paper, we propose (1) an integrated and cooperative Internet-of-Things architecture, namely General City Traffic Computing System (GCTCS), which simultaneously leverages an urban traffic simulation environment, TSC algorithms, and traffic signal communication; and (2) a general multi-agent reinforcement learning algorithm, namely General-MARL, considering cooperation and communication between traffic lights for multi-intersection TSC. In experiments, we demonstrate that the integrated and cooperative architecture of GCTCS is much closer to the real-life traffic environment. The General-MARL increases the average movement speed of vehicles in traffic by 23.2% while decreases the network latency by 11.7%.

Keywords: Intelligent transport system, Traffic signal control, Traffic, Deep learning

Citation: Wu Q, Wu J, Kang B, Du B, Shen J, et al. 2023. An integrated and cooperative architecture for multi-intersection traffic signal control. *Digital Transportation and Safety* 2(2):150–163 https://doi.org/10.48130/DTS-2023-0012

Introduction

Urban traffic congestion has become a global issue and caused severe effects, such as increased travel time, fuel consumption, and air pollution[1–3] It has serious implications for the global economy and the environment. For example, in recent years, the USA alone lost $87 billion per year in extra driving time and gasoline due to traffic jams[4,5].

Inadequate traffic infrastructure, the rapid growth of vehicles, and pre-determined traffic signal control (TSC) methods are the leading causes of urban traffic congestion[6]. Addressing urban planning and infrastructure concerns necessitates significant financial and material resources. As a result, improving the existing TSC methods is the most cost-efficient way to relieve traffic congestion.

In recent years, significant progress has been made on reinforcement learning (RL) for solving various sequential decision-making problems in Artificial Intelligence (AI) games, such as Atari[7], Go[8], and Dota2[9,10]. The TSC problem can be regarded as an agent that can make decisions at intersections by interacting with the environment, just like in a game.

It is challenging to alleviate traffic congestion by optimizing and controlling traffic signals only at a single intersection[11]. TSC is being extended from optimization of a single intersection to multiple intersections, which can be formulated as a multi-agent system with cooperation between agents. Hence, multi-agent reinforcement learning (MARL) has been receiving more attention from researchers in these years[12–14]. Furthermore, considering the network transmission and communication between traffic lights, efforts toward efficiently deploying MARL in practice has remained a research challenge.

Additionally, most research focused on lab theories and algorithms with few considerations of industrial-scale deployment issues. With the limited capabilities of the network transmission bandwidth and underlying computing resources, optimizing the deployment structure and algorithmic performances for TSC is essential for intelligence transport systems (ITS).

ITS-related technologies, such as urban traffic simulation environments[15–18], TSC algorithms[19,20], and traffic signal communication[14,21], have considerably enhanced traffic operation and management. To the best of our knowledge, there is rare research considering all aspects above for TSC, and few works can be deployed in the real-world. We summarize several existing challenges in TSC:

1) Although some studies have contributed to multi-intersection TSC[22–24], it lacks an integrated architecture to leverage the traffic simulation environment, cooperative TSC algorithm, and traffic signal communication to achieve optimal multi-intersection TSC;

2) Traditional algorithms in urban TSC rarely consider traffic light cooperation and communication simultaneously;

3) Most studies optimize the algorithms but ignore the network capacity or latency in the urban TSC process.

To address the aforementioned challenges, an integrated and cooperative architecture for TSC across multiple intersections is proposed in this paper. The main contribution is threefold:

1) An integrated architecture, namely General City Traffic Computing System (GCTCS), is proposed, which integrates an urban traffic simulation environment, TSC algorithm, and communication across the traffic signal network simultaneously.

2) A MARL algorithm, namely General-MARL, is developed for TSC based on GCTCS, considering cooperation and communication between traffic lights.

3) Comprehensive experiments have been conducted to validate the proposed architecture and algorithm with promising results. From experimental results, our novel architecture is much closer to the real-life traffic environment. With the proposed algorithm, the average speed of vehicles is increased by 23.2%, and the network latency is reduced by 11.7% compared with baseline algorithms.

The remainder of this paper is organized as follows: Section Related Work introduces related works. Section Preliminary explains the basic concepts and problem definition. Section Methodology describes the architecture of GCTCS and details the General-MARL method for cooperative traffic light control based on GCTCS. Section Experiments conducts experiments to demonstrate the advantage of the General-MARL algorithm. Section Conclusions concludes the paper and discusses future work.

Related Work

In this section, we discuss and introduce studies on the TSC, which can be divided into two typical categories: Conventional approaches and RL-based methods.

Conventional methods for TSC

Conventional TSC methods are classified into four types: fixed-time control[25], actuated control[26], adaptive control[27], and optimization-based control[28]. Fixed-time is a conventional and primary method of urban TSC, benefiting from the simplicity of deployment. It usually consists of a pre-timed cycle length, fixed cycle-based phase sequence, and phase split. While calculating the cycle length and phase split, the traffic flow is assumed to be uniform during a specific period. Since introduction in the 1950s[25], it has been a leading solution to TSC in practice considering that the urban traffic environment is complex and uncertain, and mathematical approaches cannot precisely build a model from internal operational mechanisms of a TSC system. Actuated control[26] decided whether to keep or change the current phase based on the pre-defined rules and real-time traffic data. It could set the green signal for a specific traffic signal phase if the number of approaching vehicles is larger than a threshold. Based on traffic volume data from loop sensors, adaptive control[27] created a set of traffic plans and chose the one that was best for the current traffic situation. Optimization-based control[28] formulated TSC as an optimization problem under a dynamic traffic flow and decided the traffic signal according to the observed traffic information. All of the methods discussed above heavily rely on human-designed traffic signal plans or rules.

RL based methods for TSC

RL-based methods have emerged as a promising TSC solution, which are designed for different application scenarios including single intersection control[11,29], and multi-intersection control[30,31].

Single intersection control

Abdulhai et al. [11] introduced Q-learning for TSC and presented a case study involving application to traffic signal control. Li & Wang[29] proposed the idea to set up a deep neural network (DNN) to learn the Q-function of reinforcement learning from the sampled traffic state/control inputs and the corresponding traffic system performance result. Park et al.[32] developed two traffic signal control models using reinforcement learning and a microscopic simulation-based evaluation for an isolated intersection. Additionally, the models could also be adapted for two coordinated intersections.

Multi intersection control

Multi-agent reinforcement learning (MARL) involves the participation of more than one agent[12]. It can learn through the cooperation of (1) sharing instantaneous information through interaction with the environment and (2) sharing learned policies in episodic experience.

MARL is a suitable method for the TSC problem, which can be solved as a typical MARL system for optimization of all intersections[13,14]. There exist intelligent traffic agents in the environment that can facilitate learning progress in MARL. Co-DQL model[12] used a highly scalable independent double Q-learning method based on double estimators and the upper confidence bound (UCB) policy for multi intersections. Wang et al.[13] proposed two distributed MARL control models 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. El-Tantawy et al.[30] investigated the following dimensions of the control problem: (1) RL learning methods, (2) traffic state representations, (3) action selection methods, (4) traffic signal phasing schemes, (5) reward definitions, and (6) variability of flow arrivals to the intersection. Rasheed et al.[31] designed a multi-agent DQN (MADQN) and investigated its use to further address the curse of dimensionality under traffic network scenarios with high traffic volume and disturbances. El-Tantawy et al.[30] introduced a multi-agent auto communication (MAAC) algorithm, which is an innovative adaptive global traffic light control method based on multi-agent reinforcement learning (MARL) and an auto communication protocol in edge computing architecture. The MAAC model considered traffic communication but did not leverage MARL and traffic simulation environment optimization.

From the literature, we find that most studies attempt to develop an RL or MARL model to address the TSC problem directly, ignoring the traffic simulation environment optimization and traffic communication simultaneously.

Preliminary

Traffic simulation environment

In order to create models for simulating complex traffic dynamics, urban traffic simulation systems typically integrate computing technologies and operational features of traffic flow (as illustrated in Fig. 1). Road network, vehicle description, traffic signal control, and communication between the simulation system and traffic equipment are typically the basic elements of an urban traffic simulation system.

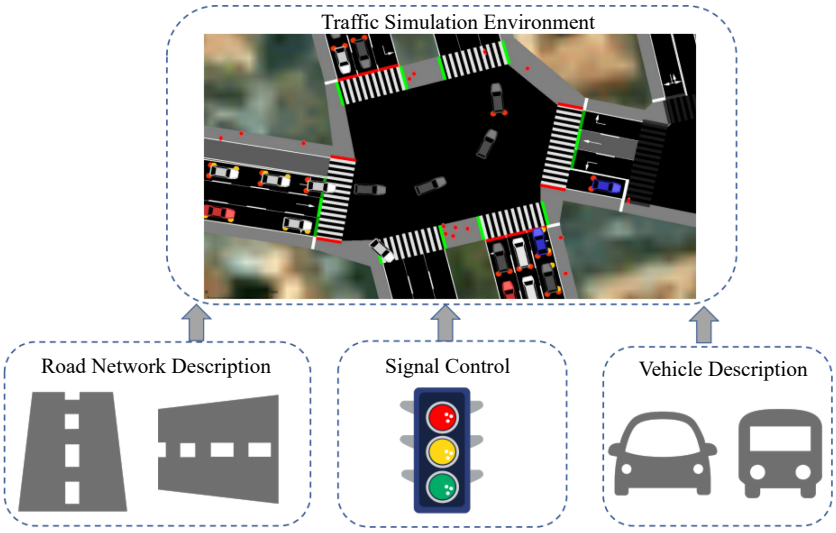

Fig. 1 An urban traffic simulation environment for traffic research.

The traffic simulation system can be divided into two groups: the commercial and open-source system. VISSIM[33] and Paramics[34] are widely used as commercial traffic simulation systems. While, open-source systems, such as MIT Simlab[15], SUMO[16], CityFLow[17] are usually adopted by researchers.

Traffic computing structure

Communication is one of the key functions in cooperation, which usually suffers from inevitable network latency. However, relevant studies on the development and deployment of computing frameworks often ignore the network delay in communication. In general, there are three major IoT computing structures, namely cloud computing[35], fog computing[21], and edge computing[36].

Cloud computing is a ubiquitous, convenient, and on-demand repository of computing resources such as networks, servers, and storage. Fog computing is closer to where data is generated than cloud computing. Edge computing and data processing are performed at the nearest location where terminal devices generate the data.

Problem definition

The TSC problem can be regarded as an agent that can make decisions at intersections by interacting with the environment.

It can be formulated as an Markov Decision Process (MDP)[37] ($\langle S, A, P, R, \pi, \gamma \rangle$), with the traffic state set S, action set A, transition probability P, reward R, control policy π and discount factor $\gamma \in [0, 1]$, as shown in Fig. 2.

In this paper, we formalize the TSC problem as decentralized intersection optimization by MARL for multi-intersection (as shown in Fig. 3). Each intersection I_i is controlled by an algorithm A_i. At time step t, A_i makes an optimal decision to choose proper signal phase after analyzing its observation of the traffic state S_i (vehicle queue, number of vehicle and etc.) at the intersection. In this case, the time T (average travel time) that vehicles spend in the traffic network can be minimized, denoted as:

$$minT_i^{I_i(A_i(S_i))} \tag{1}$$

where $T_i^{I_i(A_i(S_i))}$ is the average time vehicles spending at intersection I_i based on algorithm A_i on state S_i.

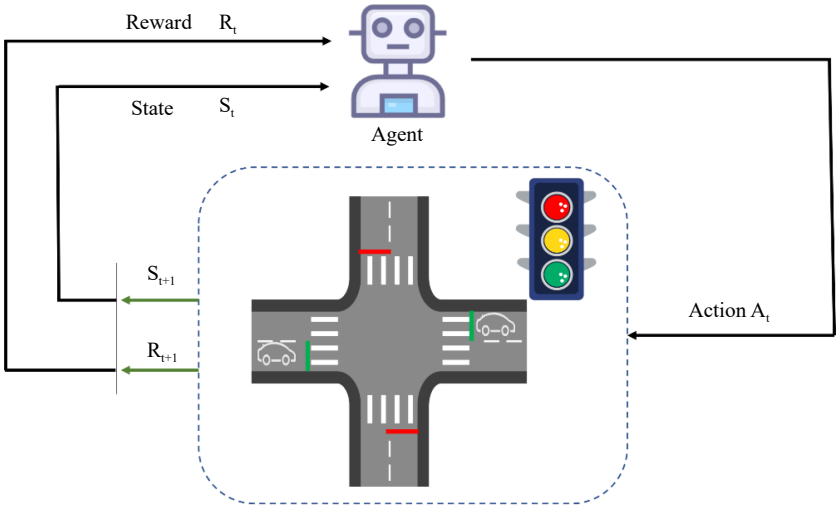

Fig. 2 An illustration of traffic signal control at an intersection. According to the current traffic state S_t and the reward R_t, the agent selects and executes the corresponding action A_t (change or maintain the current traffic light). Then the agent evaluates the effects of the action to obtain a new traffic state S_{t+1} and a new reward R_{t+1}.

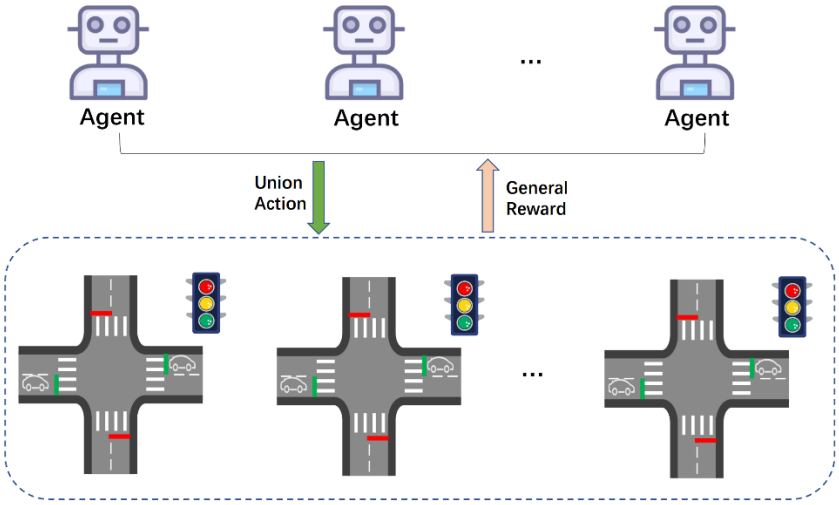

Fig. 3 The MARL structure in urban traffic signal control.

In a multi-agent system, the cooperation between agents is usually based on one computing structure. Thus, an effective and efficient integration solution covering different aspects is needed for TSC in practice.

Methodology

Architecture

In this section, we introduce an integrated and cooperative architecture, i.e., GCTCS, for urban TSC across multiple intersections. It integrates an urban simulation environment, a hybrid computing framework (cloud computing, fog computing, and edge computing), an interface for TSC algorithm deployment, a dynamic prediction module of traffic flow for traffic configuration interface, and an edge computing node of real-time traffic video processing[38].

The development of GCTCS is to support more synthetic and efficient TSC experiments involving multiple intersections and to provide simulation data that are closer to real traffic conditions for industrial deployment, as shown in Fig. 4.

GCTCS dynamically connects the simulation environment with the natural environment of multi-intersection traffic signals; provides urban TSC algorithm interfaces deployed under the hybrid computing architecture of cloud computing, fog computing, and edge computing; and takes complete account of network bandwidth and network delay. GCTCS can break through the barrier between the real traffic flow across multiple intersections and the traffic flow configuration of the simulation environment and realize dynamic synchronization

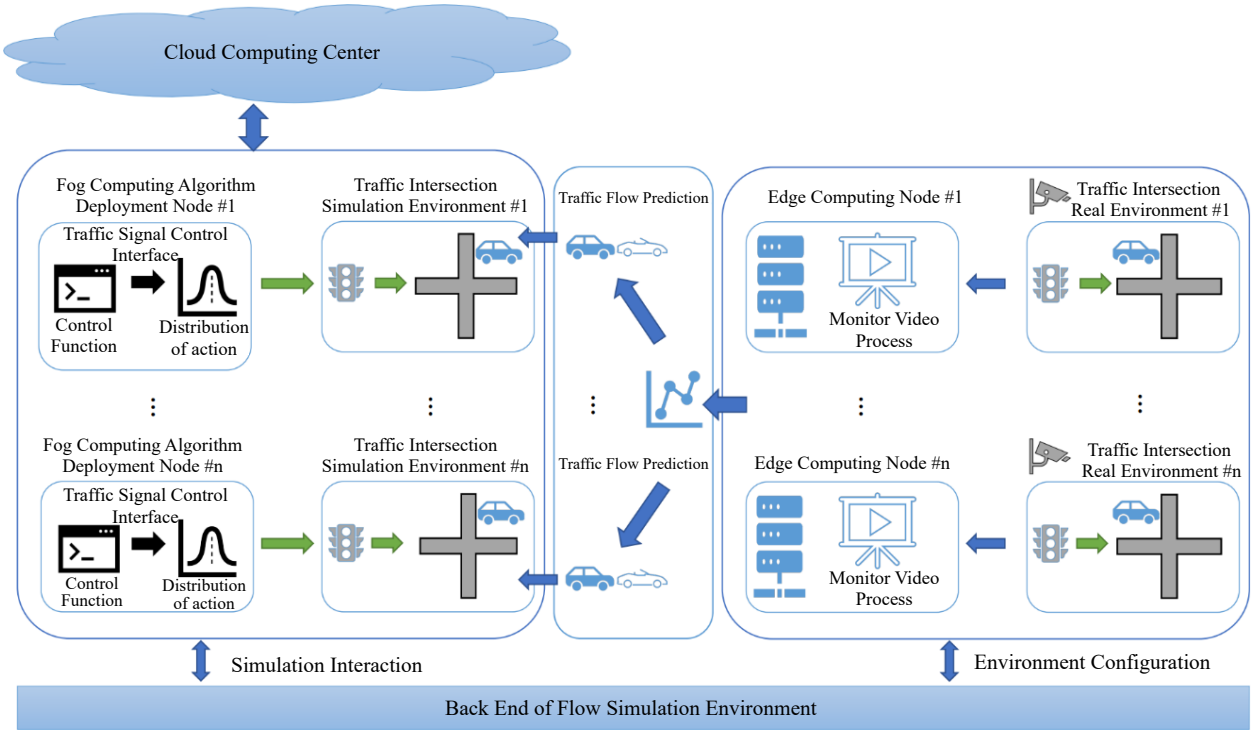

Fig. 4 The General City Traffic Computing System (GCTCS) for urban traffic signal control.

between the real environment and the simulation environment. In addition, the overall computing architecture of GCTCS is based on a hybrid computing framework, which fully considers and simulates network delay. It provides interfaces for urban TSC and traffic flow prediction algorithms.

The functions of each layer

The cloud computing layer is located at the top layer of the architecture. It is responsible for interacting with each fog computing node in the fog layer, collecting information at the fog node, and sending control instructions from the cloud computing layer.

The fog computing layer is located between cloud computing and edge computing layers and consists of various powered servers, routers, and controllers that can bear heavy processing[39]. The primary function is to compute and output the control signal instructions.

The edge computing nodes are located at the bottom layer, mainly processing surveillance videos and extracting status information at the current traffic intersection.

The module of traffic simulation environment

We designed the module of urban traffic simulation environment based on the urban traffic simulation platform described above, and it flexibly establishes the traffic flow forecasting interface for multiple intersections[40]. The module can be configured by dynamic traffic flow based on the current simulation environment, making the environment more realistic.

This module employs the traffic flow forecasting method's external interface to dynamically construct traffic flow at each intersection, allowing the traffic simulation system to assimilate the current condition and provide an algorithm interface for TSC. The simulation environment FLOW framework[18] is to optimize the urban traffic simulation environment benefiting from the forecasting capabilities based on data sets and time series models.

The traffic environments in different cities differ, and it is necessary to establish a unified method to obtain traffic circulating situations to construct the urban traffic environment closer to the real world (as shown in Fig. 5). The methodology (1) uses edge computing capabilities to convert the traffic flow captured in the urban traffic monitoring videos into text content; (2) applies a traffic flow prediction algorithm to predict relatively accurate traffic flow; and (3) dynamically interacts with the simulation environment's traffic flow configuration interface. The urban traffic environment is not built in isolation. It interacts with the simulation environment. The process is as follows:

(1) Establish a data set of urban traffic flow;

(2) Design a forecasting model for traffic flow;

(3) Develop an interactive interface with the simulation environment.

The simulation environment interacts with the traffic flow anticipated by the traffic flow prediction model, allowing the simulation environment to configure real-time traffic flow.

General-MARL

In this section, we design a TSC method, named General-MARL, based on the GCTCS.

The design of the General-MARL algorithm is according to different layers of the GCTCS architecture. Thus, the General-MARL is composed of three sub-algorithms: the algorithm at the edge computing layer (Edge-General-control), the algorithm at the fog computing layer (Fog-General-control), and the algorithm at the cloud computing layer (Cloud-General-control). Additionally, the GCTCS with a layer-to-layer connection, which has a clear structure and lowers complexity compared to point-to-point connections, making feasible for the General-MARL with the increasing computation ability for edge and fog devices.

The Cloud-General-control produces the global control signal for each fog node's parameter updating based on the joint state (all traffic states from each node) and playback mechanism (historical interactions from fog nodes). The Fog-General-control generates a TSC signal according to the traffic

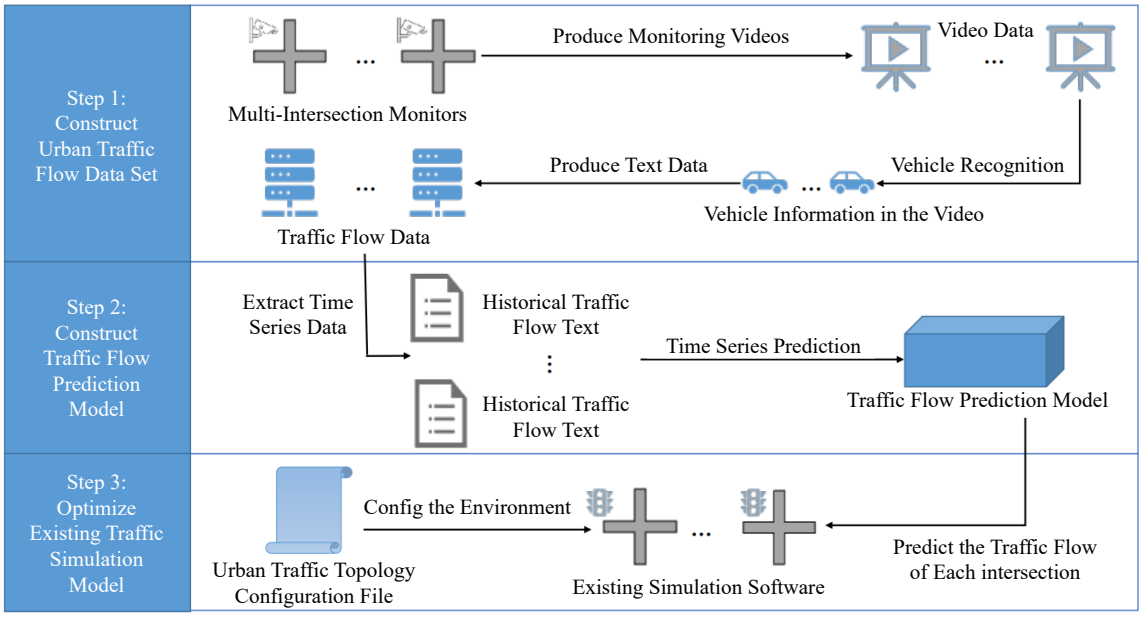

Fig. 5 The processes of construction of the urban traffic real environment.

state abstraction from Edge-General-control, the intelligent communication between agents allows an agent to share the learned strategies and the parameters from the cloud layer. Deep neural networks[41] are the basic component in General-MARL for generating control actions, communication information, and so on. The structure of the algorithm architecture is illustrated in Fig. 6.

Edge-General-control

The Edge-General-control algorithm processes the traffic surveillance video based on the target detection algorithm YOLO[38], detecting the vehicles' location at the intersection.

As shown in Fig. 7, we apply an open-source vehicle image dataset, i.e., BIT-Vehicle[42], to fine-tune the YOLO algorithm for vehicle detection precisely. The center of each intersection is the center point (0,0). We get the vehicle position from the YOLO-based model and add two fully-connection (FC) layers for fine-tuning. Then, we obtain the direction information according to the vehicle's location and center position. Hence, traffic state text is abstracted from the traffic surveillance video, including the vehicle-id, type, direction, and timestamp for passing vehicles.

Finally, we introduce the traffic flow prediction algorithm GCN-GAN[40], which can predict the traffic flow and synchronizes with the simulation environment in real-time (Algorithm 1).

Fog-General-control

The Fog-General-control algorithm includes the Nash-MARL module and the communication module to generate control strategies for urban traffic signals.

Nash-MARL Module: The Nash-MARL Module is to obtain Nash equilibrium for TSC without prior knowledge dynamically[43]. It defines that X is the traffic environment state space for multi-intersection; U is the joint action from agents to tune transfer signal; A is the action space of one agent $a \in A$; joint action is $u^a \in U$. R is the reward or objective function. The object of $Agent^i$ is to choose a proper strategy π_i and to maximize the objective function R_i.

Herein, the Bellman equation with Nash Equilibrium: firstly, fix other agents' policies π^*_{-i}; optimize the objective function of $Agent^{-i}$: $R_i\left(x;\pi_i;\pi^*_{-i}\right)$:

$$R = \max_{u\in U_i}\left\{r_i\left(x,u,\pi^*_{-i}(x)\right)+\gamma_i \mathop{\mathrm{E}}_{x'\sim p(\cdot|x,u)}\left[R_i\left(x';\pi^*_{i,1};\pi^*_{-i,1}\right)\right]\right\} \quad (2)$$

Where π^*_i is $Agent^i$' policy according to other agents' $Agent^{-i}$ action selection policies. π^*_i and $\gamma \in [0,1]$. The result of the objective (reward) function is based on the policy selection for participated each agent. It automatically generates its action policy by considering the behavior of other agents. When the behavior of other agents is stable, $Agent^i$ tries to optimize the behavior of the objective function. All agents' policies could achieve Nash equilibrium after iteration update.

Algorithm 1. Edge-General-control algorithm.

1: Input the video information of the traffic situation.
2: Capture one frame from the video: G
3: Use and fine-tune the YOLO to recognize all the vehicle's position (x,y) and type in G.
4: **for** vehicle in G **do**
5: Obtain the direction of vehicles by judging the (x,y) from the regions according the three regions predefined.
6: Record traffic state text information (vehicle -id, types, direction, and the timestamp) into traffic state text T.
7: **end for**
8: Use GCN-GAN for traffic flow prediction to T.
9: Connect traffic flow prediction capability to the urban simulation environment.

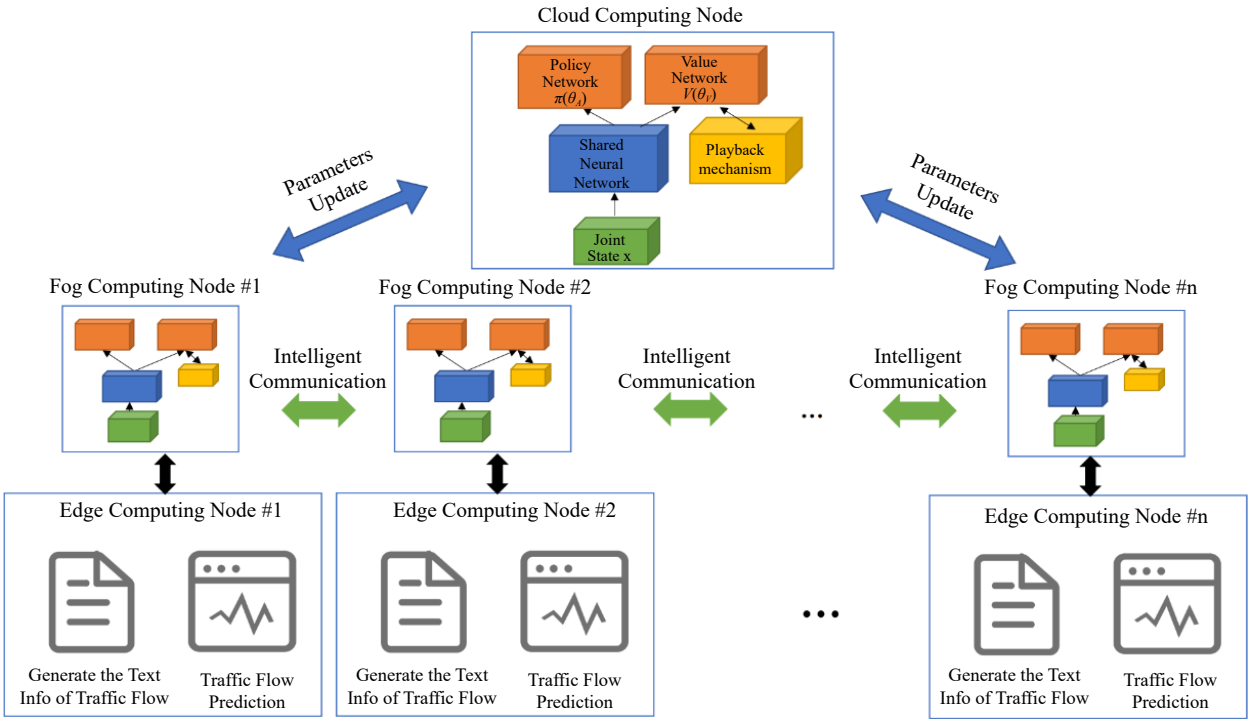

Fig. 6 The General-MARL is composed of three sub-algorithms based on different layers of the GCTCS architecture.

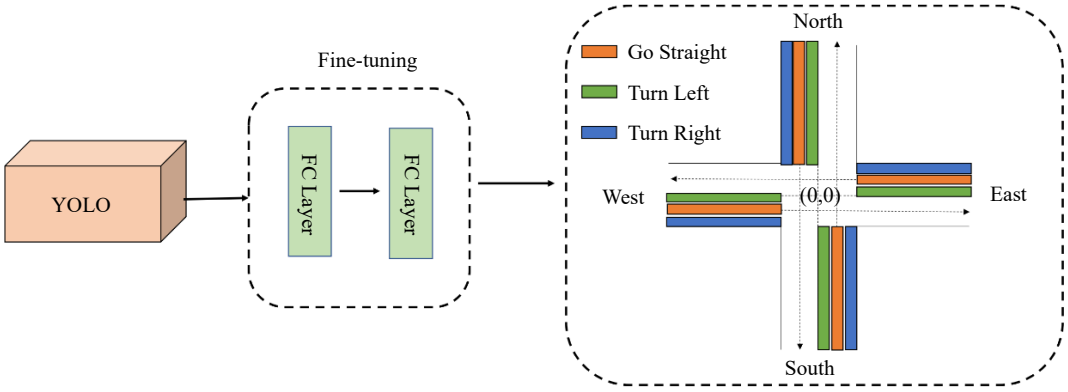

Fig. 7 The process of abstracting traffic information from the video to generate the text info from traffic flow. Detection of vehicles in the three regions stands for different directions of passing vehicles

In the design of the Nash-MARL Module, the computation of the critic function (advantage) in deep reinforcement learning might have positive (good) and negative (bad) reward results, making the learning process efficiently. Thus, the Nash-MARL module defines the Q function of an $Agent^{-i}$ as $\left(\hat{Q}_i^{\theta}(x;u)\right)_{i\in N}$; the estimate value function of $Agent^{-i}$ is $\left(\hat{V}_i^{\theta}(x;u)\right)_{i\in N}$, wherein u is the union action, and x is the union state.

$$\hat{A}^{\theta}(x;u)=\hat{Q}^{\theta}(x;u)-\hat{V}^{\theta}(x;u) \tag{3}$$

It employs the actor-critic model[44] as a framework. In the actor-critic model, the actor is a policy function and the critic is a value function. The parameter set θ into the value function parameter set and the policy function parameter set $\theta=(\theta_V,\theta_A)$, where θ_V represents the model of the value function $\hat{V}^{\theta_V}$; θ_A represents the parameters of the agent (participant) action selection policy $\hat{\pi}^{\theta_A}$. $x_m^{`}$ is the next state of x_m. The model samples M epochs. The object of the algorithm is to minimize the loss of the sampled data and the Nash-Bellman equation $L_m(\theta)$:

$$L_m(\theta)=\frac{1}{M}\sum_{m=1}^{M}\left\|\hat{V}^{\theta}(x_m)+\hat{A}^{\theta}(x_m;u_m)-r(x_m;u_m)-\gamma_i\hat{V}^{\theta}(x_m')\right\| \tag{4}$$

The module defines $y_m=\hat{L}(y,\theta_V,\theta_A)$ for simplifying the above expression.

$$y_m=\left\|\hat{V}^{\theta_V}(x_m)+\hat{A}^{\theta_A}(x_m;u_m)-r(x_m;u_m)-\gamma_i\hat{V}^{\theta_V}(x_m')\right\|^2 \tag{5}$$

Inspired by the deep Q-networks[7], the module also introduce a memory buffer (replay buffer) to store triples $y_t=(x_{t-1},u,x_t)$, which represents the previous state of the environment x_{t-1}; the union operation performed u; the next state of the environment x_t; and the reward y_t that passes through this state. The module can use stochastic gradient descent (SGD) to update the parameters after randomly selecting a piece of memory data from the replay buffer. To improve the action plan, the algorithm also employs ϵ-greedy exploration.

Additionally, the idea of parallel *space-time* is incorporated to enable the simultaneous execution in various environments. To achieve a stable learning process, multiple explorations in multiple environments would explore different policies and accelerate the learning speed. The overall algorithm structure is shown in Fig. 6. Further, the Nash-MARL algorithm process is detailed in Algorithm 2.

Communication Module: The communication module between agents allows an agent to share the learned strategies with others based on the attention mechanism[45].

The calculation steps in the communication module[14] at time t are shown in Fig. 8. At a time t in the communication module, the environment input is $X_t=\left(x_t^1,\cdots,x_t^N\right)$ and the corresponding communication information input is $C_t=\left(c_t^1,\cdots,c_t^N\right)$. Multi-agents $(Agent_1,\cdots,Agent_N)$ are able to interact with each other. Each agent receives information from receivers and transmitters internally. The receiver receives its own environmental information x_t and communication

Algorithm 2. Nash-MARL Module.

Init Episode $B>0$, $b=1$; Minibatch Size $\hat{M}>0$, Game Step N
Init Replay Buffer D, θ_V and θ_A
repeat
  Reset, go to the x_0
  repeat
    Select $u\leftarrow\pi^{\theta_A}(x)$ or select u randomly (e.g., ϵ-greedy)
    Observe $y_t=(x_{t-1},u,x_t)$
    Store y_t to Reply Buffer D
    Sampling from Replay Buffer: $Y=\{y_i\}_{i=1}^{\hat{M}}$
    Optimize $\frac{1}{M+1}\sum_{y\in Y\cup\{y_t\}}\hat{L}(y,\theta_V,\theta_A)$ fix θ_A update θ_V
    Optimize $\frac{1}{M+1}\sum_{y\in Y\cup\{y_t\}}\hat{L}(y,\theta_V,\theta_A)$ fix θ_V update θ_A
  Until $t>N$
Until $b>B$
return θ_V and θ_A

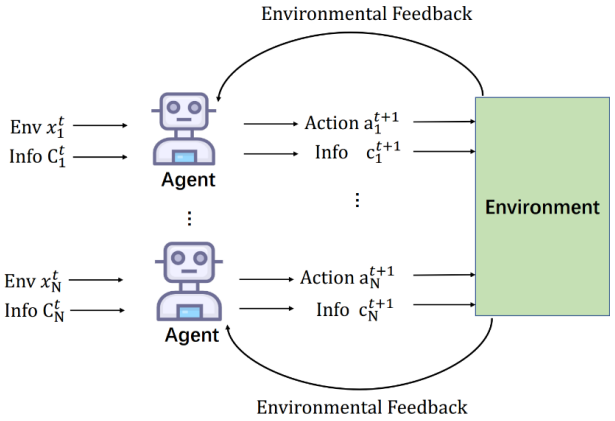

Fig. 8 The communication process between agents in the communication module.

Algorithm 3. The communication module.

1: Initialize the communication matrix of all agents C_0

2: Initialize the parameters of agent θ^i_{Sender} and $\theta^i_{Receiver}$
3: **repeat**

4: Receiver of $Agent^i$: use attention mechanism to generate communication matric $\hat{C}_t$

5: Sender of $Agent^i$: chooses an action a^i_{t+1} from policy selection network, or randomly chooses an action (e.g., ϵ-greedy exploration)

6: Sender of $Agent^i$: generate its own information through the receiver's communication matrix $\hat{C}_t : c^i_{t+1}$
7: Collect all the joint actions of Agent and execute the actions $a^1_{t+1}, \cdots, a^N_{t+1}$, get the reward from the environment R_{t+1} and next state X_{t+1}
8: **until** End of Round Episode

9: return θ^i_{Sender} and $\theta^i_{Receiver}$ for each agent

information c_t, and generates action and external interaction information group (a_{t+1}, c_{t+1}) at $t+1$.

In the communication module (as shown in Algorithm 3), the parameter set of $Agent^i$ for each agent is θ^i. Furthermore, θ^i is divided into the sender θ^i_{Sender} and receiver $\theta^i_{Receiver}$. The parameters of the sending side and the receiving end are optimized by the overall multi-agent objective function. The parameter sets of the receiver and the sender are updated iteratively for each agent.

The Fog-General-control algorithm is illustrated as below (Algorithm 4):

Cloud-General-control

The Cloud-General-control algorithm deploys the Nash-MARL module, the 'parallel universe' of the Nash-MARL algorithm are fog nodes.

The Cloud-General-control Algorithm 5 is:

Algorithm 4. Fog-General-control.

1: Apply the communication module:

2: Initialize the communication matric C_0 of all fog computing node Agents

3: Initialize the parameters θ^i_{Sender} and $\theta^i_{Receiver}$ of the fog computing node Agents

4: Receive the global parameter sets θ_V and θ_A distributed by the cloud computing node and initialize the parameter sets θ^i_V and θ^i_A

5: Initialize the Episode $B > 0, b = 1$; the minimum batch size Minibatch Size $\bar{M} > 0$, the number of game steps N
6: Apply the Nash-MARL Module:
7: Initialize the memory record Replay Buffer D
8: **repeat**
9: Reset the environment and enter the initial state x_0
10: **repeat**

11: Choose joint action $u \leftarrow \pi^{\theta_A}(x)$ or randomly choose joint action u (e.g., ϵ-greedy exploration)

12: Observe the state-action-state triplet $y_t = (x_{t-1}, u, x_t)$
13: Store triples in the Replay Buffer D

14: Extract data $Y = \{y_i\}^M_{i=1}$ from the Replay Buffer

15: $Agent^i$ receiver uses Attention mechanism to generate communication matrix $\hat{C}_t$

16: The strategy choice network of the Agent t^i sender chooses an action a^i_{t+1}, or randomly chooses action a (e.g., ϵ-greedy exploration)

17: The $Agent^i$ sender generates its own information c^i_{t+1} through the communication matrix $\hat{C}_t$ at the receiving end

18: Collect the joint actions of all Agents, execute an action $a^i_{t+1}, \cdots, a^N_{t+1}$, get rewards R_{t+1} and the next state X_{t+1} from the environment

19: Optimization step $\frac{1}{M+1}\sum_{y \in Y \cup \{y_t\}} \hat{L}\left(y, \theta^i_V, \theta^i_A, \hat{C}_t\right)$, fixes θ^i_A updates θ^i_V

20: Optimization step $\frac{1}{M+1}\sum_{y \in Y \cup \{y_t\}} \hat{L}\left(y, \theta^i_V, \theta^i_A, \hat{C}_t\right)$, fixes θ^i_V updates θ^i_A

21: **until** $> N$
22: **until** $b > B$

23: Return θ^i_V and θ^i_A

Algorithm 5. Cloud-General-control.

1: Apply the Nash-MARL module:
2: Initialize the global parameter sets θ_V and θ_A of the cloud computing center and the global counter T
3: **repeat**

4: Distributer global parameters to fog computing nodes $\theta^i_V = \theta_V, \theta^i_A = \theta_A$
5: **repeat**

6: Update global parameters $\theta_V = \theta_V + d\theta^i_V, \theta_A = \theta_A + d\theta^i_A$
7: **until** all fog computing nodes are traversed and collected
8: $T \leftarrow T + 1$
9: **until** $T > T_{max}$

Experiments

In this section, we apply General-MARL and baseline methods to the integrated and cooperative architecture GCTCS for multi-intersection TSC. The experimental process ignores or considers the situation of network bandwidth and network delay, respectively.

Dataset

The dataset from Lanzhou in China consists of two parts: (1) the traffic network; and (2) the traffic flow. The traffic network describes the traffic network, including lanes, roads, intersections, and signal phases. As shown in Fig. 9, there are 27 intersections connecting 45 roads. All the traffic networks are simulated in the simulation environment from our GCTCS. Various and flexible TSC algorithms control each intersection's signal. The distance between two intersections is two to four kilometers (km). The speed limitation is 60 (km/h).

The initial flow of incoming and outgoing vehicles at each intersection is configured based on the real traffic flow data. The traffic flow dataset contains vehicles travel information, which is described as (t, u), where t is the time that each vehicle starts entering the traffic network and u is the pre-planned route from its original location to its destination.

Parameter settings of General-MARL

Cloud computing center

We deploy the Cloud-General-control algorithm in a Docker environment[46], logically away from the intersections, and set the communication delay from the cloud center to the fog computing node to x seconds in the simulation code (set $x = 0$ second to sleep to ignore network delay; set $x = 10$ seconds to consider network delay). Then we set four hidden layers in the network for policy network generation $\pi(x)$ in the cloud computing node with 40, 80, 80 and 80 nodes in each layer, respectively. The network layers are activated and connected through the activation function ReLU[47]; the main neural network has three hidden layers, each containing 40 nodes. The layers are activated and connected through ReLU. The learning rate is set to 0.001.

Fog computing node

We deploy the Fog-General-control algorithm in the Docker environment and set it to be logically close to the intersection. The delay from the intersection to the fog computing node is set to x second (set $x = 0$ second to sleep to ignore network delay; set $x = 1$ second to consider network delay). Twenty seven fog computing nodes are deployed in the environment. The network for policy network generation $\pi(x)$ in each node is set with four hidden layers, with 40, 80, 80, and 40 nodes, respectively. The layers are connected through ReLU activation, and the main neural network has three hidden layers, each containing 40 nodes. The layers are connected through ReLU activation. The learning rate is set to 0.001.

Edge computing node

We deploy the Edge-General-Control algorithm in the Docker environment at one intersection. The network delay from the edge computing nodes to fog computing and vehicles are both set to x seconds (set $x = 0$ second to sleep to ignore network delay; set $x = 1$ second to consider network delay).

Initial traffic light period setting

The traffic control period is set to 60 s initially. The intervals of the green, red and yellow lights are set as $g_t = 27$ s, $r_t = 27$ s, and $y_t = 27$ s, respectively.

Vehicle simulation

According to the actual traffic flow forecasting algorithm, the traffic flow is predicted every 15 min, and the vehicle simulation is conducted at each intersection.

Evaluation mechanism

In the simulation environment, the passing time of all vehicles at each intersection in an Episode is recorded as $T_e = \sum_{i=1}^{I} \sum_{m=1}^{M} t_{mi}$ wherein $e \in E$ is the number of Episodes, M is the number of vehicles, and I is the number of intersections. In this configuration, we set $E = 1{,}000$; $I = 27$.

Methods for comparison

Fixed-Time[25]: a policy gives a fixed cycle length with predefined green time among all phases. The intervals of the green, red and yellow signals are fixed as green and red are 27 s, and

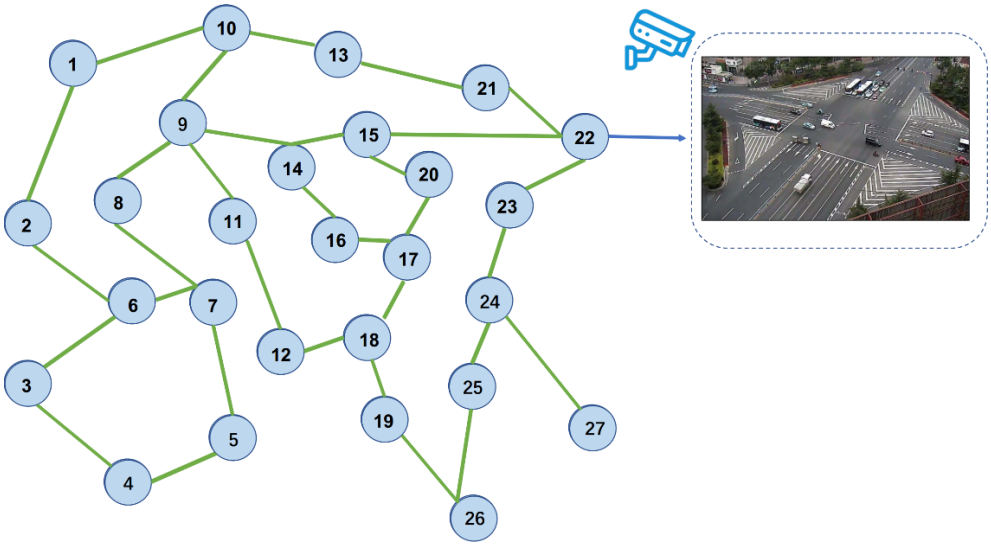

Fig. 9 The illustration of topology map of real traffic intersection.

the yellow is 6 s, which are the same as the initial intervals for other methods for a fair comparison.

Q-learning[32]: a model-free reinforcement learning algorithm to learn the value of an action in a particular traffic state based on Q-learning[48], which leverages the advantage deep neural network for addressing TSC problem. The algorithm can be deployed on the Fog or Edge node with the same parameters as shown in Table 1.

Nash-Q[49]: a model integrates Q-learning[48] and Nash Equilibrium[50] for making agents learn a better cooperative strategy, which can be deployed on the Fog node.

Nash-DQN[51]: a Deep-Q-learning methodology for model-free learning of Nash Equilibrium for general-sum stochastic games.

MAAC[14]: a multi-agent auto communication (MAAC) algorithm is an adaptive global TSC method based on MARL. MAAC combines multi-agent auto communication protocol with MARL, allowing an agent to communicate the learned strategies with others for achieving global optimization in traffic signal control. The intervals of the green, red and yellow signals are fixed as the experiment settings.

Table 1. List of parameters in this paper.

Module	Parameters	Description
Cloud Computing Center	$x = 0 \mid x = 10$	The delay from the cloud to the fog node
	20, 60, 60, 20	The hidden layers in the network
	0.001	The learning rate
Fog Computing Node	$x = 0 \mid x = 1$	The delay from intersection to the fog node
	20, 60, 60, 20	The hidden layers in the network
	0.001	The learning rate
Edge Computing Node	$x = 0 \mid x = 1$	The delay from edge nodes to fog node
Experiment Settings	$g_t = r_t = 27, y_t = 6$	The initial intervals of the green, red and yellow
	15	The traffic flow prediction period
	$E = 1{,}000$	The number of Episodes
	$I = 27$	The number of intersections
	$l = 0.001$	The learning rate
	$\gamma = 0.982$	The discount rate

Experimental process

Each experiment consists of two parts: the first part ignores the network delay and compares results with other TSC algorithms; in the other part, network delay is considered.

Ignoring network delay

The delays of edge computing, fog computing, and cloud computing nodes in General-MARL are all set to 0 without considering network latency. In the urban simulation environment, we input actual traffic data before configuring various TSC algorithm models *via* the algorithm interface. Baseline methods and the General-MARL algorithm are run in the simulation environment. The traffic flow at each intersection is updated every 15 min. We record the waiting time of vehicles at each traffic intersection every minute (considered as an environmental feedback reward) and conduct 1000 Episode (round) training.

As shown in Fig. 10, the training convergence effect of the General-MARL algorithm is as good as the Nash-DQN algorithm but does not perform the best.

Considering network delay

In the case of considering network delay, the delays of the edge computing, fog computing, and cloud computing nodes in General-MARL are all set parameters according to the previous section. Then we configure the fixed duration as the traffic signal time series as the previous setting. Q-learning (edge) deploys the Q-learning algorithm on the fog computing node to control the traffic lights, and the delay from the controlling agent to the intersection is 1 s. There are 27 docker containers deployed at 27 traffic intersections with Q-learning capabilities to generate traffic flow controlling signals, respectively.

Q-learning (center) and DQN only use the cloud computing center to collect the information at each intersection. They are applied for overall control and generation of control

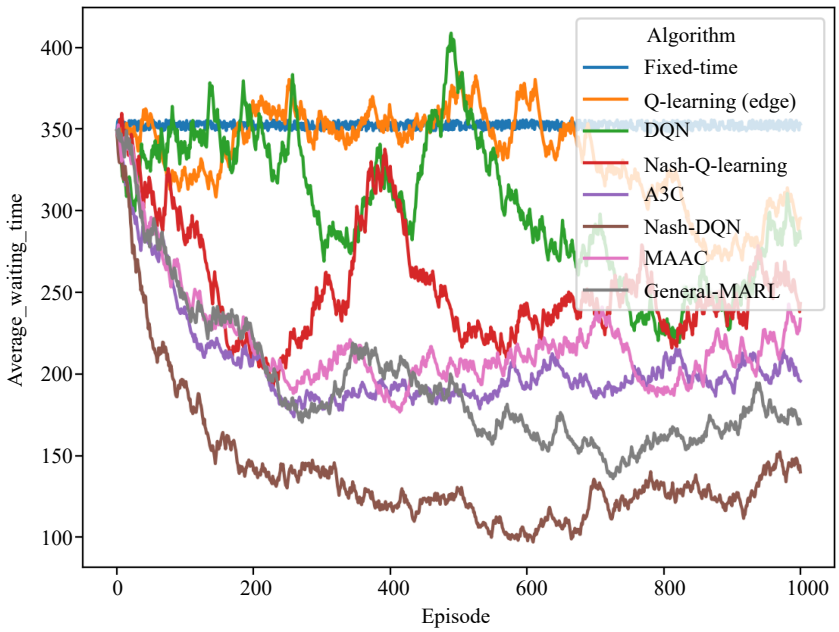

Fig. 10 Multi-intersection traffic signal control training process (without network delay).

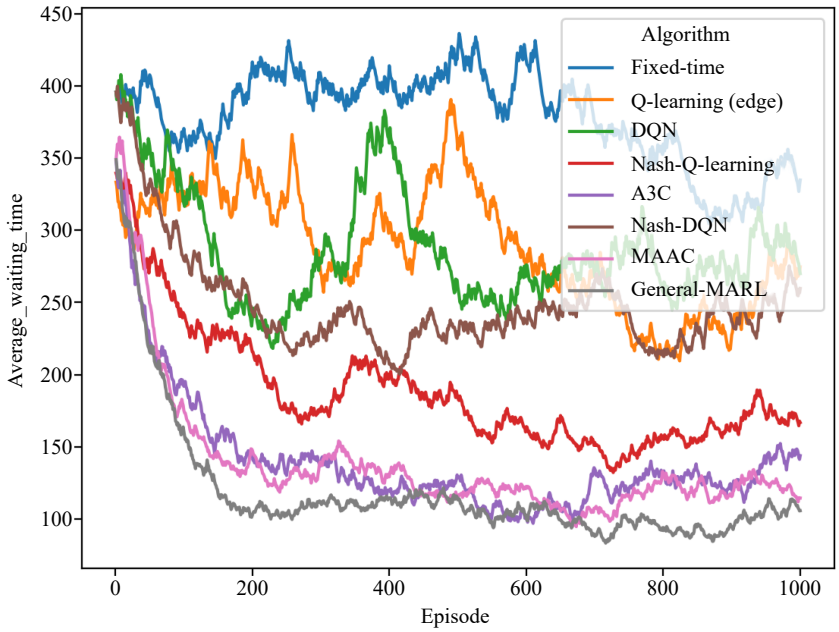

Fig. 11 Multi-intersection traffic signal control training process (with network delay).

commands for each intersection. The traffic flow prediction algorithm then updates the traffic flow at each intersection every 15 min (GCN-GAN). GCTCS records the waiting time of vehicles at each intersection every minute (as an environmental feedback reward) and conducts the training with 1000 Episodes.

As shown in Fig. 11, the convergence speed of the General-MARL is faster than Nash-DQN, and the training results outperform other baseline algorithms.

Experimental results

The results of the experiment are also split into two parts: the first part ignores network delay, and the second part considers network delay.

Ignoring network delay

After training in the simulation experiment in the previous section, 500 episode tests were performed to generate the experimental results, as shown in Table 2.

The experimental results show that, regarding the average speed and average waiting time of vehicles at 27 intersections, the General-MARL algorithm leads a similar performance as the Nash-DQN algorithm. The performance of General-MARL is not outstanding when network delay is ignored, but the overall performance of General-MARL is better than other baseline algorithms.

Table 2. Results of General-MARL and other algorithms (ignoring network delay).

Method	Average speed (km/h)	Average waiting time (s)
Fixed-time	10.17	166.70
Q learning	18.43	135.62
DQN	20.10	112.24
A3C	24.12	90.73
Nash-Q	29.70	70.14
Nash-DQN	**33.81**	**61.21**
MAAC	27.39	80.21
General-MARL	**31.22**	**62.87**

Considering network delay

We conduct 500 episodes in the tests after training in the simulation and collected the experimental results, as shown in Table 3. Regarding the average waiting time at intersections, the overall performance of General-MARL is the best, surpassing all the other algorithms, and it shows an average decrease of 18.3% waiting time compared to the baseline algorithms. As for the average speed of vehicles at intersections, the General-MARL algorithm increases the average speed by 10.2% when network delay is considered. As shown in Table 4, the General-

Table 3. Results of the average speed and waiting time in each episode (consider network delay).

Method	Average speed (km/h)	Average waiting time (s)
Fixed-time	10.15	166.71
Q learning(center)	21.69	182.75
Q learning(edge)	23.47	155.64
DQN	24.94	132.70
A3C	26.12	108.65
Nash-Q	28.32	105.55
Nash-DQN	30.86	106.37
MAAC	27.61	116.81
General-MARL	**30.78**	**92.48**

Table 4. Results of accumulated time and network delay in each episode (consider network delay).

Method	Accumulated time (s)	Network delay (s)	Delay rate
Fixed-time	38827.5	0.0	0.0%
Q learning(center)	45448.0	10809.7	23.8%
Q learning(edge)	35789.3	6522.9	18.2%
DQN	33789.9	8997.2	26.6%
A3C	31340.4	7792.1	24.9%
Nash-Q	30940.5	7536.9	24.4%
Nash-DQN	31994.6	7937.7	24.8%
MAAC	28940.7	6552.1	22.6%
General-MARL	**26912.7**	**3264.5**	**12.2%**

Table 5. Results of average waiting time at each intersection in each episode (considering network delay).

ID	Fixed-time	Q-edge	DQN	A3C	Nash-Q	Nash-DQN	MAAC	General
1	175.27	161.96	136.93	110.89	109.34	111.21	124.69	**96.81**
2	188.35	172.68	145.58	116.43	115.88	119.02	130.73	**105.46**
3	155.28	145.58	123.71	102.43	99.35	99.27	120.99	**83.59**
4	197.72	180.36	151.78	120.39	120.57	124.61	133.34	**111.66**
5	155.23	145.54	123.68	102.41	99.32	99.24	116.97	**83.56**
6	168.97	156.8	132.76	108.22	106.19	107.45	122.26	**92.64**
7	157.68	147.55	125.30	103.44	100.55	100.7	107.91	**85.18**
8	161.32	150.53	127.70	104.98	102.37	102.88	119.31	**87.58**
9	185.23	170.13	143.52	115.11	114.32	109.88	128.53	**103.40**
10	176.47	162.95	137.72	111.40	109.94	111.92	115.15	**97.60**
11	161.21	150.44	127.63	104.94	102.31	102.81	119.28	**87.51**
12	125.96	121.55	104.32	90.01	104.30	81.77	105.69	**64.20**
13	131.62	126.19	108.06	92.41	87.52	85.15	117.87	**67.94**
14	169.15	156.95	132.88	108.30	106.28	107.55	122.33	**92.76**
15	175.52	180.84	152.16	120.64	109.47	124.96	40.06	**112.04**
16	132.47	126.89	108.62	92.77	87.94	85.65	108.20	**68.50**
17	164.12	152.83	129.56	87.56	103.77	104.55	113.46	**89.44**
18	166.87	155.08	131.38	107.33	105.14	106.19	121.45	**91.26**
19	150.39	141.57	120.48	100.36	96.90	96.35	115.11	**80.36**
20	177.77	164.01	138.58	111.95	110.59	112.70	125.65	**98.46**
21	153.63	144.23	122.62	101.73	98.52	98.29	96.35	**82.50**
22	167.54	155.63	131.82	107.62	105.48	106.59	121.71	**91.70**
23	177.62	163.89	162.05	157.98	110.52	112.61	139.54	**121.93**
24	186.84	171.45	144.58	115.79	115.13	118.11	129.15	**104.46**
25	175.36	162.04	136.99	110.93	109.39	111.26	128.73	**96.87**
26	175.23	161.93	136.90	110.87	109.32	111.18	124.67	**96.78**
27	188.35	172.68	145.58	116.43	115.88	119.02	104.64	**102.77**

MARL algorithm fully exceeds the baseline algorithms and other algorithms in terms of the accumulated waiting time of vehicles, network delay time, and delay rate indicators. The delay rate is reduced by 7.4%. Furthermore, the General- MARL algorithm optimizes each intersection to a certain extent, as shown in Table 5.

Overall analysis

The analysis of experimental results reveals that network delay has a significant impact on the actual application effect of the TSC algorithm. Although General-MARL showed good performance even when network delay was ignored, implementation of the method in urban TSC should take into account the influence of network latency for industrial deployment.

With the consideration of network delay, the General-MARL fully surpasses the baseline algorithms on average vehicle speed at intersection, average vehicle waiting time, vehicle cumulative waiting time, network delay time, and the indicators of delay rate. The average speed of vehicles increases by 23.2%, and the network latency decreases by 11.7%, as shown in Fig.12.

Conclusions

In this paper, we proposed an integrated and cooperative IoT architecture GCTCS and the General-MARL algorithm for multi-intersection traffic signal control. The results from the proposed framework and algorithm showed that the average speed of vehicles was increased by 23.2%, and the network latency was

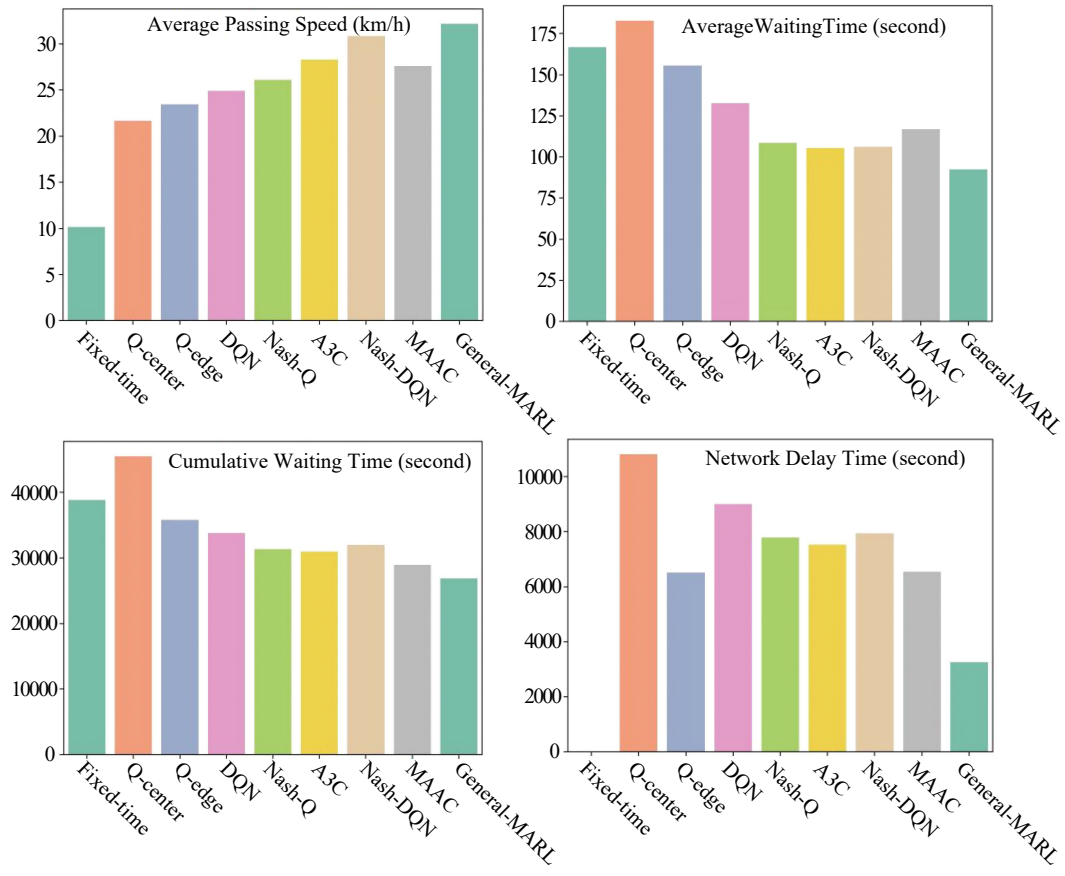

Fig. 12 Comparative results among different algorithms.

reduced by 11.7%, when compared with baseline algorithms. Our results proved that the application of MARL in urban traffic signal control needs to consider multiple factors, including the simulation environment, algorithm process, deployment architecture, and network delay. The results also validated the best performance of the proposed General-MARL. Therefore, GCTCS and General-MARL showed great potential in practical applications and theoretical contributions for multi-intersection traffic signal control with large-scale deployment in real road networks.

In traffic signal control, not only the vehicles' but also the pedestrians' behaviours[52], such as crossing times and waiting times are affected. In future studies, pedestrians will be included to cover all the road users at intersections. We will be able to further validate our proposed methodology based on real-life case studies because we have been invited by one city from China to deploy our GCTCS and algorithm in a test area.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61673150, 11622538). We acknowledge the Science Strength Promotion Programme of UESTC, Chengdu, China.

Conflict of interest

The authors declare that they have no conflict of interest. Bo Du and Jun Shen are the Editorial Board members of *Digital Transportation and Safety*. They were blinded from reviewing or making decisions on the manuscript. The article was subject to the journal's standard procedures, with peer-review handled independently of these Editorial Board members and their research groups.

Dates

Received 15 April 2023; Accepted 16 June 2023; Published online 29 June 2023

References

1. Zhao D, Dai Y, Zhang Z. 2011. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42:485–94
2. Ng V, Kim HM. 2021. Autonomous vehicles and smart cities: A case study of Singapore. In Smart cities for technological and social innovation, eds. Kim HM, Sabri S, Kent A. USA: Academic Press, Elsevier. pp. 265–287. https://doi.org/10.1016/B978-0-12-818886-6.00014-9
3. Sheng MS, Sreenivasan AV, Sharp B, Du B. 2021. Well-to-wheel analysis of greenhouse gas emissions and energy consumption for electric vehicles: A comparative study in Oceania. *Energy Policy* 158:112552
4. Harris N, Shealy T, Klotz L. 2016. Choice architecture as a way to encourage a whole systems design perspective for more sustainable infrastructure. *Sustainability* 9(1):54
5. Afrin T, Yodo N. 2020. A survey of road traffic congestion measures towards a sustainable and resilient transportation system. *Sustainability* 12(11):4660
6. Lee WH, Chiu CY. 2020. Design and implementation of a smart traffic signal control system for smart city applications. *Sensors* 20(2):508
7. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, et al. 2013. Playing atari with deep reinforcement learning. *arXiv* Preprint
8. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550:354–59
9. Berner C, Brockman G, Chan B, Cheung V, Dębiak P, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv* Preprint
10. Telikani A, Tahmassebi A, Banzhaf W, Gandomi AH. 2022. Evolutionary machine learning: A survey. *ACM Computing Surveys (CSUR)* 54(8):1–35
11. Abdulhai B, Pringle R, Karakoulas GJ. 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129(3):278–85
12. Wang X, Ke L, Qiao Z, Chai X. 2020. Large-scale traffic signal control using a novel multiagent reinforcement learning. *IEEE Transactions on Cybernetics* 51(1):174–87
13. Wang T, Liang T, Li J, Zhang W, Zhang Y, et al. 2020. Adaptive traffic signal control using distributed MARL and federated learning. *2020 IEEE 20th International Conference on Communication Technology (ICCT), Nanning, China, 28-31 October 2020*. USA: IEEE. pp. 1242–48. https://doi.org/10.1109/ICCT50939.2020.9295660
14. Wu Q, Wu J, Shen J, Yong B, Zhou Q. 2020. An edge based multi-agent auto communication method for traffic light control. *Sensors* 20(15):4291
15. Ben-Akiva M, Koutsopoulos HN, Toledo T, Yang Q, Choudhury CF, et al. 2010. Traffic simulation with MITSIMLab. In *Fundamentals of traffic simulation*, ed. Barceló J. New York: Springer. pp. 233–68. https://doi.org/10.1007/978-1-4419-6142-6_6
16. Krajzewicz D. 2010. Traffic simulation with SUMO – simulation of urban mobility. In *Fundamentals of traffic simulation*, ed. Barceló J. New York: Springer. pp. 269–93. https://doi.org/10.1007/978-1-4419-6142-6_7
17. Zhang H, Feng S, Liu C, Ding Y, Zhu Y, et al. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. *WWW '19: The world wide web conference, San Francisco, CA, USA, 2019*. New York, NY, USA: Association for Computing Machinery. pp. 3620–24. https://doi.org/10.1145/3308558.3314139
18. Jang K, Vinitsky E, Chalaki B, Remer B, Beaver L, et al. 2019. Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles. *ICCPS '19: Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, Montreal Quebec Canada, April 16–18, 2019*. pp. 291–300. c>. pp. 291–300. https://doi.org/10.1145/3302509.3313784
19. Wei H, Zheng G, Yao H, Li Z. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. *IKDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London United Kingdom, August 1923, 2018*. New York, United States: Association for Computing Machinery. pp. 2496–505.ry. pp. 2496–505. https://doi.org/10.1145/3219819.3220096
20. Liang X, Du X, Wang G, Han Z. 2019. A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology* 68(2):1243–53
21. Wu Q, Shen J, Yong B, Wu J, Li F, et al. 2019. Smart fog based workflow for traffic control networks. *Future Generation Computer Systems* 97:825–35
22. Huo Y, Tao Q, Hu J. 2020. Cooperative control for multi-intersection traffic signal based on deep reinforcement learning and imitation learning. *IEEE Access* 8:199573–85
23. Yang S, Yang B. 2021. A semi-decentralized feudal multi-agent learned-goal algorithm for multi-intersection traffic signal control. *Knowledge-Based Systems* 213:106708

24. Yang S, Yang B, Kang Z, Deng L. 2021. IHG-MA: Inductive heterogeneous graph multi-agent reinforcement learning for multi-intersection traffic signal control. *Neural networks* 139:265–77
25. Webster FV. 1958. Traffic signal settings. *Technical report. Road Research Technique Paper No. 39*. Road Research Laboratory, London.
26. Cools, S. B. ; Gershenson, C. and D'Hooghe, B. 2013. Self-organizing traffic lights: A realistic simulation. In Advances in applied self-organizing systems, ed. Prokopenko M. London: Springer. pp. 45–55. https://doi.org/10.1007/978-1-4471-5113-5_3
27. Hunt PB, Robertson DI, Bretherton RD, Royle MC. 1982. The SCOOT on-line traffic signal optimisation technique. *Traffic Engineering & Control* 23(4):190–92
28. Sun X, Yin Y. 2018. A simulation study on max pressure control of signalized intersections. *Transportation research record* 2672(18):117–27
29. Li L, Lv Y, Wang F. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 3(3):247–54
30. El-Tantawy S, Abdulhai B, Abdelgawad H. 2014. Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems* 18(3):227–45
31. Rasheed F, Yau KLA, Low YC. 2020. Deep reinforcement learning for traffic signal control under disturbances: A case study on Sunway city, Malaysia. *Future Generation Computer Systems* 109:431–45
32. Park S, Han E, Park S, Jeong H, Yun I. 2021. Deep Q-network-based traffic signal control models. *Plos One* 16(9):e0256405
33. Lownes NE, Machemehl RB. 2006. VISSIM: a multi-parameter sensitivity analysis. *Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, December 3-6, 2006*. pp. 1406-13. IEEE. https://doi.org/10.1109/WSC.2006.323241
34. Cameron GDB, Duncan GID. 1996. PARAMICS—Parallel microscopic simulation of road traffic. *The Journal of Supercomputing* 10:25–53
35. Fox A, Griffith R, Joseph A, Katz R, Konwinski A, et al. 2009. Above the clouds: A berkeley view of cloud computing. *Technical Report No. UCB/EECS-2009-28*. University of California at Berkeley, USA. www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
36. Bagchi S, Siddiqui MB, Wood P, Zhang H. 2019. Dependability in edge computing. *Communications of the ACM* 63(1):58–66
37. Sutton RS, Barto AG. 2018. *Reinforcement learning: An introduction*. Cambridge, MA: MIT press.
38. Bochkovskiy A, Wang CY, Liao HYM. 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv* Preprint
39. Telikani A, Shen J, Yang J, Wang P. 2022. Industrial IoT intrusion detection via evolutionary cost-sensitive learning and fog computing. *IEEE Internet of Things Journal* 9(22):23260–71
40. Zhang L, Wu J, Shen J, Chen M, Wang R, et al. 2021. SATP-GAN: Self-attention based generative adversarial network for traffic flow prediction. *Transportmetrica B: Transport Dynamics* 9(1):552–68
41. Goodfellow I, Bengio Y, Courville A. 2016. *Deep learning*. Cambridge, Massachusetts (MA): MIT press.
42. Dong Z, Wu Y, Pei M, Jia Y. 2015. Vehicle type classification using a semisupervised convolutional neural network. *IEEE transactions on intelligent transportation systems* 16(4):2247–56
43. Wu Q, Wu J, Shen J, Du B, Telikani A, et al. 2022. Distributed agent-based deep reinforcement learning for large scale traffic signal control. *Knowledge-Based Systems* 241:108304
44. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, et al. 2016. Asynchronous methods for deep reinforcement learning. *Proceedings of The 33rd International Conference on Machine Learning (ICML), New York, USA, 2016*. New York, USA: PMLR. pp. 1928–37.
45. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. 2017. Attention is all you need. Advances in neural information processing systems. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*. pp.6000–10
46. Merkel D. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal* 239(2):2
47. Agarap AF. 2018. Deep learning using rectified linear units (relu). *arXiv* Preprint
48. Watkins CJCH. 1989. *Learning from delayed rewards*. PhD Thesis. University of Cambridge, England
49. Hu J, Wellman MP. 2003. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4(Nov):1039–69
50. Nash JF Jr. 1950. Equilibrium points in n-person games. *PNAS* 36(1):48–49
51. Casgrain, P.; Ning, B. and Jaimungal, S. 2022. Deep Q-learning for Nash equilibria: Nash-DQN. *Applied Mathematical Finance* 29(1):62–78
52. Du B, Zhang C, Shen J, Zheng Z. 2021. A dynamic sensitivity model for unidirectional pedestrian flow with overtaking behaviour and its application on social distancing's impact during COVID-19. *IEEE Transactions on Intelligent Transportation Systems* 23(8):10404–17