Distributed Learning for Vehicle Routing Decision in Software Defined Internet of Vehicles

Kai Lin, *Senior Member, IEEE*, Chensi Li, Yihui Li, Claudio Savaglio, *Member, IEEE*, and Giancarlo Fortino, *Senior Member, IEEE*

***Abstract*— With the increasing number of vehicles, the traffic congestion is becoming more and more serious. In order to alleviate such a problem, this article considers transmission and inference delay of cloud centralized computing in the software defined Internet of Vehicles (SDIoV), and builds a new SDIoV architecture based on edge intelligence, for supporting real-time vehicle routing decision through distributed multi-agent reinforcement learning model. Then, a software defined device collaboration optimization method is designed to improve the efficiency of distributed training. Combined with multi-agent reinforcement learning, a distributed-learning-based vehicle routing decision algorithm (DLRD) is proposed to adaptively adjust vehicle routing online. The performed simulations show that the DLRD can successfully realize real-time routing decision for vehicles and alleviate traffic congestion with the dynamic changes of the road environment.**

***Index Terms*— Distributed deep learning, edge intelligence, vehicle routing decision, software defined Internet of Vehicles.**

I. Introduction

With the development of the social economy, the increasing number of vehicles has led to an urgent need for the innovative solutions provided by the Internet of Vehicles (IoV). Software-defined network (SDN) is an innovative network architecture that separates the data forwarding plane and the network control plane, so that the centralized network controller can flexibly control the basic network facilities and network traffic. The advantages of SDN are introduced into IoV to form a software-defined Internet of Vehicles (SDIoV), which can fully schedule different IoV resources to comprehensively improve the performance of traditional IoV. As an important part of smart city, SDIoV is facing the challenge of vehicle congestion, facing the challenging problem of traffic congestion: this issue is hindering the development of modern society, seriously impacting on traffic efficiency, people quality of life, and road accidents [1]. Fortunately, the advent of intelligent edge computing has further promoted the development of SDIoV, empowering the vehicle routing decision and thus the safety and efficiency of transportation [2], [3].

Vehicle routing decision is the core problem that IoV needs to solve. A better routing decision solution can not only improve the traffic efficiency, but also make the driving environment of the vehicle safer. For example, Fu and Yin [4] proposed a hybrid routing decision algorithm based on simulated annealing algorithm and particle swarm optimization, which has a good global optimization ability and improves the safety of vehicles. Different from the above method, edge intelligent computing adopts distributed machine learning and combines historical and real-time traffic environment information to rationally plan vehicle travel paths and positively impact on traffic congestion and efficiency [5].

In order to achieve vehicle routing decision for SDIoV, traditional SDIoV mainly completes the processing and analysis of massive data in cloud servers, especially the training of machine learning models, which requires a large amount of raw data to be transmitted from vehicles to the cloud [6]. Abdelrahman *et al.* [7] proposed a novel cloud-based route planner that utilizes drivers' individualized risk profiles in suggesting routing options based on drivers' personal skillfulness levels. Adiththan *et al.* [8] proposed an adaptive offloading technique for control computations into the cloud for vehicle routing decision. Although the above methods exploit artificial intelligence technology to improve routing decision, the aggregation process of massive data to the cloud will consume a lot of networking, computation and storage resources. At the same time, artificial intelligence models running on the cloud are prone to form inference delays, which prevent vehicles to promptly receive real-time routing decision. Moreover, with the increasing number of connected vehicle, cloud centralized processing is also facing problems of network congestion and task waiting delays caused by data deluge [9].

This article uses edge intelligent computing to replace the centralized processing of the cloud. We design a SDIoV architecture that supports multi-agent learning [10] to implement edge intelligence decisions and control network

Manuscript received June 24, 2020; revised August 24, 2020; accepted September 8, 2020. Date of publication September 28, 2020; date of current version June 2, 2021. This work was supported in part by the Liaoning Province Higher Education Innovative Talent Support Program, in part by the Fundamental Research Funds for the Central Universities under Grant DUT19JC22 and Grant DUT18RC(3)069, in part by the National Natural Science Foundation of China under Grant U1908214, in part by the Italian MIUR, PRIN 2017 Project Fluidware under Grant CUP H24I17000070001, and in part by the National Key Research and Development Program of China under Grant 2020YFB170458 and Grant 2018YFB1600600. The Associate Editor for this article was Z. Lv. *(Corresponding author: Kai Lin.)*

Kai Lin, Chensi Li, and Yihui Li are with the School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China (e-mail: link@dlut.edu.cn; lichensi@mail.dlut.edu.cn; 928982000@mail.dlut.edu.cn).

Claudio Savaglio and Giancarlo Fortino are with the Department of Informatics, Modeling, Electronics, and Systems (DIMES), University of Calabria, 87036 Rende, Italy (e-mail: g.fortino@unical.it; csavaglio@dimes.unical.it).

Digital Object Identifier 10.1109/TITS.2020.3023958

data deluge, which is also in line with the inevitable trend of SDIoV development [11]. For making full use of edge resources and reducing inference delay, the asynchronous parallel multi-agent deep learning is adopted to design the routing decision algorithm [12].

1) An edge intelligence based SDIoV architecture, which treats vehicles and roadside units (RSUs) as terminals and edge devices, respectively, to avoid the reasoning delay problem in routing decision due to cloud centralized processing and to mitigate network pressure caused by data deluge. Moreover, it supports distributed multi-agent reinforcement learning model for routing decision.
2) A software defined device collaboration optimization method to more efficiently and distributedly execute routing decision tasks under an edge intelligent architecture. It calculates the resource cost overhead according to the task requirements and combines the long-short-term memory neural network (LSTM) to predict the resources cost of routing decision task, which fully considers the sharing and mutual exclusion of the resource requirements to optimize the task set and interaction mode.
3) A distributed-learning-based vehicle routing decision algorithm (DLRD). The edge-side devices cooperate in an asynchronous and parallel manner to solve the routing decision of the autonomous vehicles based on multi-agent deep reinforcement learning to obtain real-time routing decisions.
4) An extensive set of simulations to evaluate the proposed algorithm by comparing with other three algorithms. The results demonstrate that DLRD can achieve high performance for reducing traffic congestion to improve the efficiency of transportation.

The rest of this article is organized as follows. In Section II, we review the related works. The edge intelligence based SDIoV architecture and multi-agent reinforcement learning based routing decision model are presented in Section III. In Section IV, the device interaction optimization method is introduced. In Section V, we describe algorithm DLRD, while the experiment results and analysis are presented in Section VI. Final considerations and future work conclude the paper.

II. Related Work

We review recent literature and related works in three different perspectives that are closely related to our work: *A)* vehicle routing decision, *B)* intelligent edge computing, and *C)* distributed deep learning.

A. Vehicle Routing Decision

As an important part of SDIoV, vehicle routing decision focuses on optimizing route selection for vehicles to improve overall traffic performance. Nha *et al.* [13] used the algorithm Dijkstra for optimal vehicle routing decision, and updated the plan based on real time traffic conditions. Pothan *et al.* [14] adopted state lattice strategy for vehicle routing decision, where a set of paths containing all the geometry, kinematics, and constraints of vehicles are preset, and a rotation matrix is used in this method to describe novelty. Vagner [15] proposed an intelligent route planner system which could avoid the traffic jams of the city in a more easy way. Zhao *et al.* [16] introduced content-data-friendly information-center networking architecture into the internet of vehicles, and achieved efficient routing decision for automatic vehicles through the big data acquisition and analysis. Duan *et al.* [17] proposed the route search algorithm including establishment of environmental models and visualization of digital map for implementing the process of intelligent vehicle routing decision. Rasekhipour *et al.* [18] proposed a model predictive route-planning controller which is capable of treating different obstacles and road structures distinctly while planning the optimal route utilizing vehicle dynamics. However, the above routing decision methods are based on the interests of individual vehicle, without meditating the impact of the vehicle's behavior on the traffic environment. Differently, our proposal focuses on planning the appropriate route for the vehicle with consideration of the overall traffic environment.

B. Intelligent Edge Computing

Intelligent edge computing [11], [19] transfers networking, computation and storage resource and services to the network edge. It greatly decreases the data transmission from massive terminals to the cloud for source data processing, analysis and storage, thereby reducing network load [20], [21]. With the development of intelligent edge computing, many researchers apply it in the field of IoV. Zhang and Letaief [22] outlined the latest developments in edge intelligence for IoV, introduced key design issues, methods and hardware platforms, and typical use cases for intelligent vehicles, including edge-assisted perception, mapping and positioning. Xu *et al.* [23] proposed a computational offload method that uses vehicle-to-all communication (V2X) technology for data transmission in edge computing, combined with related algorithms to generate a balanced offload strategy and find the best on the premise of confirming the calculated route uninstall strategy. Garg *et al.* [24] proposed a data-driven transportation optimization model to complete intelligent vehicles Cyber threat detection, which uses unmanned aerial vehicles as intermediate aerial nodes between vehicles and edge nodes to facilitate the transmission of data from vehicles to the edge for real-time analysis. Luo *et al.* [25] proposed a software-defined collaborative data sharing architecture based on edge computing in 5G Internet of Vehicles, and verified the superiority and scalability of the proposed architecture through simulation experiments. Qi *et al.* [26] proposed a knowledge driven service offloading decision framework for IoV which supports the pre-training at the powerful edge computing node and continually online learning when the vehicular service is executed. Moubayed *et al.* [27] used multiple access/mobile edge computing (MEC) by placing V2X services at edge nodes to satisfy delay/latency requirements of V2X. In this article, intelligent edge computing is further explored to implement distributed deep learning and asynchronous parallel decision-making.

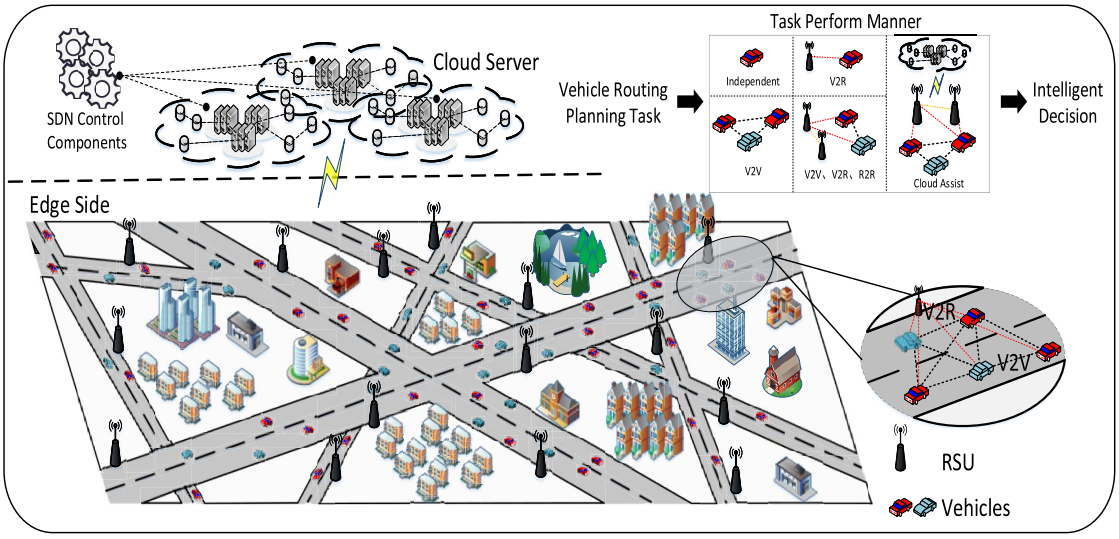


Fig. 1. Edge intelligence based SDIoV architecture.

C. Distributed Deep Learning

The growth of application services has brought a large number of pending data, the learning and processing of these data has complicated the training of deep learning models [28]. Distributed training using multiple devices has become an essential choice. Zhu *et al.* [29] proposed a scalable multiagent learning algorithm in an edge-assisted distributed routing framework to efficiently find the Nash equilibria, enables virtual agents representing vehicles to interact with others to make real-time routing decisions. Mynuddin and Gao [30] proposed a novel distributed predictive cruise control (PCC) algorithm based on reinforcement learning, which can reduces the travel time and fuel consumption rate of vehicles. Ju *et al.* [31] proposed a distributed Kalman filter and a modified generalized likelihood ratio algorithm to detect and estimate the deception attacks of IoV sensors on vehicles. Tian *et al.* [32] proposed a web attack detection system which takes advantage of analyzing URLs based on distributed deep learning, and proved that the system is competitive in detecting Web attacks. Saputra *et al.* [33] proposed a novel framework based on distributed deep learning, which allows mobile edge nodes on the network to cooperate and exchange information for reducing errors in content demand predictions without revealing mobile users private information. Langer *et al.* [34] proposed a method for distributed training of deep learning models on spark, that is specifically designed to run in low-budget environments. In this article, the distributed multi-agent reinforcement learning model is designed and deployed to achieve vehicle routing decision.

III. System Architecture and Problem Formulation

In this section, the edge intelligence based SDIoV architecture is designed, which implements routing decision decisions by supporting distributed multi-agent reinforcement learning.

A. Edge Intelligence Based SDIoV Architecture

A representation of edge intelligence based SDIoV architecture is reported in Fig. 1. The architecture is built for a general traffic environment, which is mainly composed of vehicles, RSUs and cloud servers, denoted by set $V = \{v_1, v_2, \ldots, v_n\}$, $R = \{r_1, r_2, \ldots r_g\}$ and $C = \{c_1, c_2, \ldots c_w\}$ respectively. Vehicles and RSUs jointly participate in learning tasks for routing decision as edge-side devices.Vehicles act as terminal devices to initiate learning requests for routing decision and participate in learning tasks. RSUs work as edge servers to provide a quick response service for vehicles, and at least one RSU is deployed stationary on each road segment. Vehicle-to-Vehicle (V2V) and Vehcile-to-RSU (V2R) communication are used for data transmission between terminals and between terminals and edge servers, respectively. Vehicles and RSUs communicate with the cloud through long-distance wireless communication technologies such as 5G. This article assumes that all vehicles, RSU and cloud servers have data processing functions, which together provide intelligent services for vehicle-initiated routing decision tasks. The terminal-edge-cloud device set can be expressed as $I = \{V, R, C\} = \{u_1, u_2, \ldots u_m\}$.

The real-time position of vehicles can be obtained by GPS or other localization technologies, and shared in SIoV through RSUs and the cloud. The position of vehicles can change with time, while the position of RSUs and cloud servers are guaranteed to remain unchanged. Since the height difference between vehicles, RSUs and cloud servers is usually negligible, the distance between them is calculated from the horizontal Euclidean distance based on longitude and latitude:

$$d(u_p, u_q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \tag{1}$$

where x_p, y_p and x_q, y_q represent the latitude and longitude of device u_p and u_q.

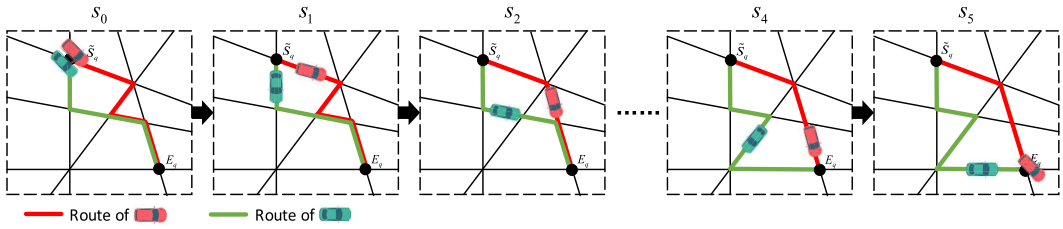


Fig. 2. State update of routing decision by multi-agent reinforcement learning.

The architecture separates data and control by combining with SDN, which is deployed on cloud servers in the form of software as control components for facilitating global logic control and fast data forwarding. In particular, it is responsible for organizing device collaboration during training in this article.

B. Multi-Agent Reinforcement Learning for Routing Decision

As a kind of distributed deep learning, the multi-agent reinforcement learning takes full advantage of its parallel training, and shares one system environment to all agents for collaborative training. In the multi-agent interaction process, decisions from different agents influence each other to collaboratively form better decision results. In this article, the distributed multi-agent reinforcement learning method is utilized to make decisions for routing decision tasks initiated by vehicles. The decision-making process focuses on the actual traffic conditions, and the purpose of alleviating traffic congestion is achieved by adjusting the vehicle's route selection. The multi-agent reinforcement learning model for routing decision is defined as follows:

$$\langle \widetilde{V}, S, \{A_p\}_{p\in\widetilde{V}}, f, \{R_p\}_{p\in\widetilde{V}} \rangle \tag{2}$$

where $\widetilde{V} = \{\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_h\}$ represents the set of vehicles that initial planning request at a certain moment, i.e. h multi-agent participating in reinforcement learning. S represents the state set, which is specifically represented by driving routes of vehicles, $s_k \in S$ represents the state of the agents at step k. $\{A_p\}_{p\in\widetilde{V}}$ represents the action space of multi-agent. Assuming that the maximum number of intersections in the environment is N_a, the direction that the vehicle v_p can choose is $A_p = \{a_p^1, a_p^2, \ldots, a_p^{N_a}\}$. $f : S \times A \times S \longrightarrow [0, 1]$ is the state transition probability of the agent, and $R_p : S \times A \times S \longrightarrow \mathbb{R}$ represents the reward function.

Fig. 2 shows the state update of routing decision taking two agents as an example, each disjoint line segment in the figure represents a road segment. The starting and ending points of route that two agents need to plan are $\widetilde{S}_q$ and E_q, respectively. The red line and the green line represent the driving routes of two vehicles, which will change according to the actions selected by vehicles. Before reaching the ending point, vehicles can change their driving direction at any time to choose a different route.

The setting of the overall-reward function is related to the traffic flow of each road state, which affects the actions chosen by multi-agent. Assuming that the distribution density of vehicles on the road has not changed in a short time, this is defined as the time required for an iterative training, and each adjustment of the routing decision will naturally cause the distribution density of vehicles to change. In order to alleviate traffic congestion, this article uses the difference between traffic conditions of state s_k and s_{k+1}, i.e. before and after route adjustment to form a overall-reward function:

$$R_i(s_k) = \sum_{y=1}^{N_r} \frac{x_k^y}{\overline{\omega_y} l_y} - \sum_{y=1}^{N_r} \frac{x_{k+1}^y}{\overline{\omega_y} l_y} \tag{3}$$

where N_r represents the number of road segments, x_k^y is the number of vehicles on the road segment y in the state s_k, $\overline{\omega_y}$ represents the average width of the road segment y, l_y represents the length of the road segment y, $\frac{x_k^y}{\overline{\omega_y} l_y}$ represents the distribution density of vehicles that can reflecting traffic congestion, a larger value means more serious traffic congestion. Vehicles get rewarded by avoiding congested paths.

In order to realize dynamic and real-time routing decision, it is necessary to perform the task of reinforcement learning through device cooperation based on intelligent edge computing. In this article, vehicle routing decision with the same starting point and ending point is regarded as the same multi-agent reinforcement learning task, and the vehicles that initiated the planning constitute the agent set of the task. Define the set of tasks that need to be learned as $M = \{m_1, m_2, \ldots, m_d\}$. Considering the mobility of vehicles and the communication distance limitations of V2V and V2R, the cooperative devices that perform learning tasks are neither fixed nor easily determined. In each time period t, the following judgment matrix of whether the device can participate in the collaboration is defined as:

$$W(t) = [w_{pq}(t)]_{m\times m} \tag{4}$$

where $w_{pq}(t) = \rho_{pq} \times (D_{pq} - d_{pq}(t))$ represents whether devices u_p and u_q can cooperate without considering resource constraints during time period t, named as collaborative feasibility. D_{pq} represents the maximum distance that can be communicated between devices, the specific value is determined by the devices' types. $d_{pq}(t)$ is the maximum distance between devices u_p and u_q during time period t. ρ_{pq} is the correlation between devices to measure the possibility of potential stable cooperation of devices which is calculated as follows:

$$\rho_{pq} = \psi_{pq} \times \xi_{pq} \tag{5}$$

where $\psi_{pq} = \frac{2n_{pq}}{n_p+n_q}$ if both devices are vehicles, and n_{pq} counts the number of road segments where vehicles v_p and v_q

traveled together, n_p and n_q respectively represent the number of road segments that vehicles v_p and v_q traveled. Otherwise, $\psi_{pq} = 1$. $\xi_{pq} = \frac{\sum_{i=1}^{d} \xi_{pq}^i}{d}$ represents the feasibility of devices u_p and u_q cooperating to execute the same task and ξ_{pq}^i is defined as follows:

$$\xi_{pq}^i = \begin{cases} \dfrac{(Y_p^i + Y_q^i) \times (1 - y_{pq}^i)}{Y_i}, & p \neq q \\ Y_p^i / Y_i, & else \end{cases} \tag{6}$$

where Y_q^i and Y_q^i respectively represent the data that devices v_p and v_q can provide to participate in performing task m_i, respectively. Y_i is the data required by task m_i. y_{pq}^i represents the proportion of duplicate data related to task m_i in devices v_p and v_q. When $w_{pq}(t) > 0$, devices u_p and u_q can cooperate to perform tasks, and the larger value means more stable cooperation. Conversely, these two devices cannot collaborate.

IV. Software Defined Device Collaboration Optimization Method

Routing decision decisions consume corresponding computing, storage, and communication resources. Since most terminals have limited computing and storage capabilities, it is difficult to perform learning tasks independently without the assistance of other devices. Meanwhile, the cooperative reinforcement learning of multi-agent requires the sharing of information between devices to achieve the overall optimization goal. Therefore, it is necessary to effectively organize the devices that participates in the execution of routing decision tasks, and whether these devices can obtain the various resources required to perform the tasks in time is critical to real-time decision performance. By considering the resource overhead of tasks under the edge intelligence architecture, a software defined device collaborative organization optimization method is designed which focuses on device set selection and interaction mode adoption for completing routing decision based on reinforcement learning. In this method, the initial set of collaborative devices and the interaction modes used are formed according to the resource overhead of task execution, and the long-short-term memory neural network is used to predict the future resource overhead of the task, and further optimization is used to reduce the decision delay of routing decision.

A. Resource Overhead Measurement for Routing Decision Tasks

For a certain learning task m_i, the set of all interaction modes completing by collaborate of terminal-edge-cloud devices (may not include these three types of devices at the same time) is expressed as set ξ_i, the set of terminal-edge-cloud devices set involved in interaction mode $\omega_i^\varepsilon \in \xi_i$ is $I_i^\varepsilon = \{u_1^\varepsilon, u_2^\varepsilon, \ldots u_l^\varepsilon\}$.

Each routing decision task can be performed by different sets of devices through different interaction modes, which results in different resource requirements and makes the online resource overhead measurement problem complicated. In order to adapt to this situation, no matter what device set and interaction mode is used for the routing decision task, the corresponding resource overhead needs to be calculated in time. In general, the computing overhead of devices to perform a task is mainly reflected in their CPU usage, so it is expressed in terms of these CPUs' computing power and usage rate. The storage overhead of task is expressed by the storage occupation of the corresponding devices, and the communication overhead is calculated by the channel bandwidth between the devices occupied by the task. To simplify the description, this article assumes that the channel bandwidth required for device communication remains unchanged when performing routing decision learning tasks. The calculation of the resource overhead also needs to consider the changes in resource requirements caused by different device sets and interaction modes under multi-task parallelism. The calculation process is divided into two steps: Single routing decision task overhead calculation and Overall routing decision overhead calculation.

1) Single Vehicle Routing Decision Task Overhead Calculation: Due to the different measurement pattern involved in computing, storage, and communication overheads, the overhead of vehicle routing decision task m_i is defined as a three tuple $\varphi_i = (r_i, g_i, p_i)$, where r_i, g_i and p_i represent the computing, storage, and communication overhead required for task execution, respectively. When calculating the overhead of task m_i in interactive mode ω_i^ε, the CPU usage and storage occupation of the task on each device, as well as the communication requirements between devices can be obtained. Define the required CPU usage rate and storage occupation of device u_p^ε are ς_i^p and τ_i^p, the communication bandwidth occupied between device u_p^ε and u_q^ε is B_i^{pq} under interaction mode ω_i^ε. The resource overhead of task m_i is:

$$\varphi_i = (r_i, g_i, p_i) = \left(\frac{\sum_{p=1}^{l} \varsigma_i^p \times W_p}{\sum_{p=1}^{l} W_p}, \sum_{p=1}^{l} \tau_i^p, \sum_{p=1}^{l} \sum_{q=1}^{l} B_i^{pq}\right) \tag{7}$$

where W_p represents the cpu capability of the device u_p^ε, which is calculated according to the cpu core number, main frequency and transistor number. Routing decision tasks must ensure the real-time nature of decision-making, and the overhead of resources exceeding the burden of the device set will result in the inability to perform the task within the specified time. In this case, device sets and interaction modes need to be adjusted.

2) Overall Vehicle Routing Decision Overhead Calculation: The change of interaction mode has an independent impact on each kind of resource overhead, and the degree of impact is restricted by objective factors such as data distribution and network environment. When measuring the overall overhead of routing decision in SDIoV, the parallel execution of multiple tasks creates resource sharing (including storage data) and competition conflicts between devices. The combined effect of multi-dimensional resource sharing and mutual exclusion leads to the calculation of the overall overhead cannot be equal to the sum of individual tasks. Therefore, this article draws sharing effect between tasks into the solution of calculating the overall resource overhead. The overall overhead of routing decision tasks is also defined as a three tuple $\varphi_M = (r_M, g_M, p_M)$,

where r_M, g_M and p_M represent the computing, storage, and communication overhead required for execution of all tasks. The resource sharing between the tasks due to mutual influence is expressed by a matrix as follows:

$$H = [\varphi_a, N_a]_{y\times 2} \tag{8}$$

where a represents resources that have been jointly used by parallel execution of multiple tasks, and there are y in total, $\varphi_a = (r_a, g_a, p_a)$ stands for the overhead of resource a, and N_a indicates the number of tasks using resource a. The overhead required for resources shared by multiple tasks only needs to be calculated once, so the overall overhead for routing decision in SDIoV is calculated as follows:

$$\varphi_M = \sum_{i=1}^{d} \varphi_i - \sum_{a=1}^{y} \varphi_a \times (N_a - 1) \tag{9}$$

B. Initialing Device Set and Interaction Mode

In the initial division of collaborative devices for a task, priority is given to the interest of the vehicles that initiated the task. The multi-agent routing decision task can be regarded as a collaboration of single-agent tasks, while independent single-agent training is used to make decisions when target vehicles have different starting or ending points. The device set must contains the vehicles that initiate the routing decision request. If these vehicles have sufficient computing and storage capacity, no additional devices are needed to perform the distributed learning process, and communication with other devices is limited to information sharing during reinforcement learning. Otherwise, continuously selecting idle RSUs or vehicles with the greatest collaborative feasibility $w_{pq}(t)$ by matrix $W(t)$, and add them into the collaborative device set to jointly perform the routing decision task through information exchange. Although adding participating devices provide more information and resources for task execution, it is also reduces the efficiency of task execution caused by increasing the complexity of the collaboration process. Therefore, when the storage and computing capacity meet tasks, adding collaboration devices is stopped.

The initial selection of the task cooperation equipment set is based on a single task without considering it from the overall perspective of SDIoV. In fact, a device may perform multiple tasks at the same time. When the device's own capacity is not enough to perform multiple tasks at the same time, problems such as resource waiting and repeated calculation will occur. It is necessary to adaptively adjust the device sets and interaction modes that perform tasks, and improve resource utilization and reduce task waiting time through resource sharing. In order to share partially overlapping resources in the execution of multiple routing decision tasks, the formation of a collaborative device set needs to consider the correlation between tasks. Since the routing decision task is initiated by the vehicles, the relevance of tasks is related to the social correlation of the vehicles, and is also related to the planning route. By comprehensively considering the correlation between the vehicle sets $\widetilde{V}_i$ and $\widetilde{V}_j$ that initiate tasks m_i and m_j, the correlation between the tasks and is calculated as follows:

$$\vartheta_{ij} = \overline{\rho_{ij}} \frac{2N_{ij}}{N_i + N_j} \frac{1}{\sum_{k=1}^{m} \widetilde{W_{ij}^k} + 1} \tag{10}$$

where $\overline{\rho_{ij}}$ is the correlation between vehicle sets $\widetilde{V}_i$ and $\widetilde{V}_j$, which is equal to the average social correlation between vehicles. N_i and N_j represent the number of states input for reinforcement learning training of tasks m_i and m_j, while N_{ij} is the number of coincident input states of tasks m_i and m_j.

$$\widetilde{W_{ij}^k} = \left|\frac{\sum_{u_a\in\widetilde{V}_i} w_{ka}(t)}{\widetilde{N}_i} - \frac{\sum_{u_b\in\widetilde{V}_j} w_{kb}(t)}{\widetilde{N}_j}\right| \tag{11}$$

where u_a and u_b respectively belong to different sets of vehicles that initiated tasks m_i and m_j. $\widetilde{N}_i$ and $\widetilde{N}_j$ represent the number of vehicles that initiated tasks m_i and m_j. $w_{ka}(t)$ is obtained according to equation 4, indicating the collaborative feasibility of device u_k and u_a. $\sum_{u_a\in\widetilde{V}_i} w_{ka}(t)$ uses summation to measure the overall collaborative feasibility between device u_k and vehicles which initiate task m_i. $w_{kb}(t)$ and $\sum_{u_b\in\widetilde{V}_j} w_{kb}(t)$ are defined in the same way. $\widetilde{W_{ij}^k}$ represents the difference of average collaborative feasibility between vehicles which initiate task m_i and m_j who cooperate with device u_k. The smaller the value of $\widetilde{W_{ij}^k}$W means the greater the feasibility of vehicles initiating the task m_i or m_j to perform the task in cooperation with the same device u_k. In other words, $\sum_{k=1}^{m} \widetilde{W_{ij}^k}$ reflects the difference between these two device sets participating in the collaborative execution of task m_i and m_j.

After obtaining the correlation between tasks, the adjustments is made to the tasks' device sets and interaction modes, so that the similar data processing part of the tasks with higher correlation is performed in the same device, and the resource sharing between the tasks is realized. That is, for tasks whose task relevance is great, analyze the problems that the task needs to process on each device, move similar processing to the device side with a smaller amount of tasks, and share device resources to perform the task. Through preliminary division and adjustment, the task device set and interactive mode is formed.

C. Device Set and Interaction Mode Optimization

1) Resource Overhead Prediction for Routing Decision Task: The alternation of execution way and target of routing decision task leads to dynamic changes in resource requirements. If the trend can be predicted, the required resources can be scheduled in advance to ensure the completion of the task [35]. LSTM as a time-loop neural network is suitable for predicting time-related events. Its input contains useful output results of the previous time period. Each repeated training will add a new input, and finally the result of the next time period to be predicted is output through the previous comprehensive training. To this end, this article uses LSTM to predict resource overhead based on changes in execution interaction mode of task.

Fig. 3 shows the single task resource prediction process. The input function contains historical information about whether

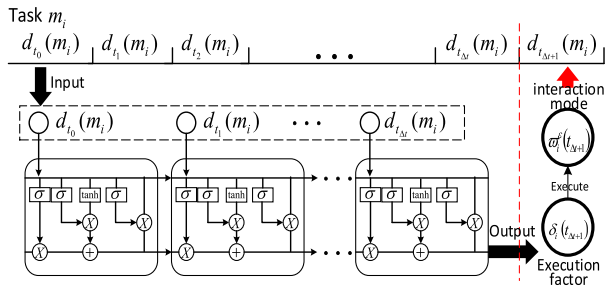


Fig. 3. Single task resource prediction process.

the task is executed and what interaction mode is used:

$$d_{t_x}(m_i) = [\delta_i(t_x), \varpi_i^{\varepsilon}(t_x), I_i^{\varepsilon}(t_x)] \tag{12}$$

where $\delta_i(t_x)$ is a coefficient factor which judges whether task m_i is executed, $\varpi_i^{\varepsilon}(t_x)$ and $I_i^{\varepsilon}(t_x)$ respectively represent the set of devices that execute the task in time period t_x and the corresponding interaction mode. As for task m_i, the task execution matrix $d_{t_{\Delta t+1}}(m_i)$ in the future time period $t_{\Delta t+1}$ can be obtained by inputting the task execution matrix $d_{t_x}(m_i)$ of the known time periods t_0 to $t_{\Delta t}$. First, whether the task is executed is judged in the time period $d_{t_{\Delta t+1}}(m_i)$ according to the obtained coefficient factor $\delta_i(t_{\Delta t+1})$. If it is executed, the resource overhead of the task in the future time period $t_{\Delta t+1}$ is calculated by Equation 6. Furthermore, the obtained $d_{t_{\Delta t+1}}(m_i)$ can be used as a known condition to predict the task execution status of the time period $t_{\Delta t+2}$, and then the process is repeated continuously to obtain long-term resource overhead prediction.

2) Device Collaborative Organization Optimization: The parallel multi-agent routing decision task based on edge intelligent computing needs to fully consider the sharing and mutual exclusion of the resource requirements of execution tasks. Therefore, it is necessary to design a device collaborative organization optimization method that can be used in environments with limited network resources.

$X_t^{|M|\times|I|}$ is used to denote the resource allocation matrix at time period t, where the value x_{ip}^t is a three tuple $(r_{ip}^t, g_{ip}^t, p_{ip}^t)$ represents the computing, storage and communication overhead caused by device u_p performing task m_i. According to the above overhead prediction and preliminary adjustment of the device set, an initialized matrix X_t can be obtained. In order to improve the execution performance of all routing decision tasks in SDIoV, it is necessary to avoid mutually exclusive tasks from being executed by the same device and deploy different tasks with shared data on same devices when multiple tasks are executed in parallel. In addition, considering that under the condition of limited resources, some tasks can not be completed within the time limit, the dynamic task prioritization is introduced for task deployment.

Let the dynamic priority of task m_i be $\zeta_i^t = f_{\zeta}(\zeta_i^0, T(x_i^t, w_i^t, c_i^t))$, where ζ_i^0 represents the initial priority of task. $T(x_i^t, w_i^t, c_i^t)$ represents the total waiting time of task form the task generated to time period t, including resource allocation mode x_i^t, resources waiting time w_i^t and training time c_i^t. In the process of device collaborative optimization, the task priority needs to be adjusted adaptively. The trend of adjustment is to increase the priority of tasks with higher initial priority and longer waiting time. In order to reduce the training and reasoning delay of task execution for real-time decision-making, dynamic programming for solving Markov decision is adopted to optimize device collaboration.

$$\langle S', A_s, R(s,a), P\rangle \tag{13}$$

The state space S' is the set of tasks' volume in different time period t which need to be completed. In each state, the selectable resource allocation way set A_s is the possible set of all X_t, P is the state transition probability, and the reward $R(s,a)$ depends on the state and decision of each step. In this article, P is set to 1, and T is defined as the maximum total execution time, then the iteration number of each Markov decision process is not more than $\frac{T}{|t|}$. Define $\tilde{T}$ as the sum of task execution and waiting time at a certain state, and the priority sum of all completed tasks at this state is ζ_t. The reward is defined as:

$$R(s,a) = -(\tilde{T} + \frac{1}{\zeta_t}) \tag{14}$$

The set of all strategies defined in the system forms a strategy space Π. The goal of dynamic programming is to find the strategy π^* with the highest benefit:

$$max\, E\{\sum_{t=0}^{T/|t|} \gamma\, R(s,a)\} \tag{15}$$

where γ is the discount factor. In order to find the optimal solution, we introduce the Bellman equation [36] and define the value function of state s' as $\kappa^{\pi}(s)$, then the expectation that strategy π can obtain is:

$$\kappa^{\pi}(s) = E_{\pi}\{\sum_{t=0}^{T/|t|} \gamma\, R_{k+t+1} | s = s_k\} \tag{16}$$

The strategy π^* needs to be found to make $\kappa^{\pi^*}(s) = max\kappa^{\pi}(s)$. According to the Bellman equation, and writing the value function of all states a vector κ, we can get:

$$\kappa(s) = \max_{a\in A_s}\{R(s,a) + \frac{R(s,a)}{\gamma + R(s,a)} \sum_{z\in S'} \kappa(z)\} \tag{17}$$

The reward function is normalized as follows:

$$\tilde{R}(s,a) = R(s,a)\frac{R(s,a)+\gamma}{\eta+\gamma} \tag{18}$$

where η is the normalized parameter, and the Bellman equation is deformed as follows:

$$\tilde{\kappa}(s) = \max_{a\in A_s}\{\tilde{R}(s,a) + \frac{\tilde{R}(s,a)}{\gamma + \tilde{R}(s,a)} \sum_{z\in S'} \tilde{\kappa}(z)\} \tag{19}$$

Finally, the value iteration method is used to continuously increase the reward until the reward value cannot be increased, and the obtained is the final resource allocation matrix, i.e. the device cooperation mode.

V. Distributed-Learning-Based Vehicle Routing Decision Algorithm

A. Distributed Deep Q Learning Based Routing Decision

In this article, the goal of vehicle routing decision is to improve traffic efficiency by reducing congestion, which requires collaboration between vehicles. Each multi-agent reinforcement learning task plans paths for vehicles with the same start and end points, and the vehicles that initiate the request form the agent set of the task. Vehicles with computing, communication, and storage capabilities act as terminals to perform distributed reinforcement training. Here, distributed collaborative learning requires each vehicle v_p to maintain its own local Q value $Q_p^k(s_k, a_k)$ and strategy $h_p(s_k)$ through its own action decision.

The update of $Q_k(s_k, a_k)$ at the $k+1$ step is as follows:

$$Q_p^{k+1}(s_k, a_p^k) = max\{Q_i^k(s_k, a_p^k), r_p^{k+1} + \gamma\, max\, Q_p^k(s_{k+1}, a_p^k)\} \tag{20}$$

where a_p^k and s_k denote the selected action and the state that represents routes updated by all vehicles that initiate routing decisions in step k. γ is the discount factor. The action selection of vehicle in each step will change its routing. Besides, the travel time of vehicles is also considered to evaluate the single-reward as one of factors. Defining $r_p^{k+1} = R_i(s_{k+1}) + \chi$ as the single-reward of vehicle v_p in state s_{k+1}, where $\chi = \frac{T^k - T^{k+1}}{T^k}$ represents the time feedback, T^k is the time vehicle v_p needs to travel on the selected routing to the ending point in step k, the shorter travel time means the greater reward value. The wrong driving action chosen by vehicles is lead to addition of driving time, which makes the single-reward reduce caused by negative time feedback.

The strategy $h_p(s_k)$ is updated as follows:

$$h_p^{k+1}(s_k) = \begin{cases} a_p^k, & \max\limits_{a_p^{k+1}} Q_p^{k+1}(s_{k+1}, a_p^{k+1}) \\ & \neq \max\limits_{a_p^{k+1}} Q_p^k(s_{k+1}, a_p^{k+1}) \\ h_p^k(s_k), & else \end{cases} \tag{21}$$

which means that for the state s_k, when the value Q increases, the new action is selected to increase the reward, otherwise the action selected at the state is unchanged.

All agents calculate independently and greedily obtain their own optimal solution. Furthermore, we add collaboration between agents for distributed multi-agent learning to find the group optimal solution. In this article, agents make their own decisions while considering each other's influence, so that multi-agents eventually form a consistent strategy.

In order to facilitate vehicle collaboration, course-based learning method is combined to achieve collaborative distributed reinforcement learning [37]. In the process of machine learning, the course-based learning method trains samples in a certain order. Simple samples are trained first, and the next sample is added for training when the current training process tends to be stable. In this way, samples are added into training from simple to complex until all samples are trained, which can get a better local optimal solution and improve training speed. In the multi-agent reinforcement learning in this article, due to the learning objectives of all agents are the same, the learning of multi-agent can be divided into multiple parts. The training of each agents part is a sub-learning process, we start training from the agents with shorter planning route. When the training reaches a certain level, we add the next part of agents in training until all agents are joined.

The join time of next agents part needs to be determined by the current learning progress of training, that is, only when the current training results are relatively stable. ϕ_k is defined to represent the learning situation of the system in state s_k:

$$\phi_k = Q_k(s_k, a_k) - Q_{k-1}(s_{k-1}, a_{k-1}) \tag{22}$$

where $Q_k(s_k, a_k) = \sum_{i=1}^{h'} Q_i^k(s_k, a_i^k)$. h' is the number of agents currently joining the training. If ϕ_k is very small, i.e. the rewards obtained by changing the action no longer increase significantly, the next part of agents can join for training.

In SDIoV, if there are many vehicles that initiate the routing decision task or the planned route is long, traditional reinforcement learning method are difficult to obtain effective results in time due to the excessive computational cost for such problems. the state space of reinforcement learning is very complex and huge. Therefore, this article uses a deep Q-learning (DQN) algorithm that combines deep neural networks and reinforcement learning for online inference. The $\epsilon - greedy$ strategy is used to determine the next action taken by the agent, the action that maximizes the Q value is selected with the probability of $1-\epsilon$, or the action is selected randomly with the probability of ϵ [38].

In this article, the states are abstracted as a one-dimensional vector input to the DQN, and the next action of multi-agent is decided according to the network output $Q(s, a; \theta)$, where parameter θ represents the deep reinforcement learning network parameter. The loss function $L(\theta)$ for DQN training of one single-agent is as follows:

$$L(\theta_p) = E_{s_k, a_p^k, r_p^{k+1}, s_{k+1}}[(r_p^{k+1} + \gamma \max_{a'} Q_p(s_{k+1}, a'; \theta_p) - Q_p(s_k, a_p^k; \theta_p))^2] \tag{23}$$

The partial derivative of the loss function is utilized to calculate the update gradient of parameter θ:

$$\begin{aligned} \nabla_\theta L(\theta_p) &= \frac{\partial L(\theta_p)}{\partial \theta_p} \\ &= E_{s_k, a_p^k, r_p^{k+1}, s_{k+1}}[(r_p^{k+1} + \gamma \max_{a'} Q_p(s_{k+1}, a'; \theta_p) \\ &\quad - Q_p(s_k, a_p^k; \theta_p)) \frac{\partial Q_p(s_k, a_p^k; \theta_p)}{\partial \theta_p}] \end{aligned} \tag{24}$$

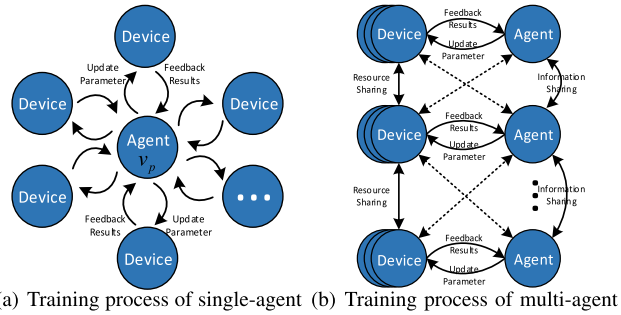
The update of the parameter θ_p is:

$$\theta_p = \theta_p - \alpha \nabla_\theta L(\theta_p) \tag{25}$$

where α is the learning rate.

B. Distributed-Learning-Based Vehicle Routing Decision

Through the software defined device collaborative organization optimization method described above, the set of devices that perform the routing decision task can be determined. For any multi-agent learning routing decision task m_i, the training



(a) Training process of single-agent (b) Training process of multi-agent

Fig. 4. Training process of single-agent and multi-agent based on distributed learning.

and calculation of the task can perform through the device set. Assuming that the multi-agent task has h agents, then each task can be divided into h single agent learning tasks. These agents use V2V or V2R communication to realize the information sharing between them, and then cooperate to maximize the sum of their rewards for achieving the final goal.

As shown in Fig. 4(a), for training process of one single-agent of task m_i initiated by vehicle v_p, the system takes the state of single vehicle or multiple vehicles with different starting points or ending points as input for training, and independently outputs the most suitable planned route for each vehicle. A set of collaborative devices I_p (which may include vehicles, RSUs, or cloud servers) provides resources required for training, these devices performs distributed training in an asynchronous and parallel manner. Each device participating in the learning performs the parameter training in certain specific layers during the training, as well as the communication transmission and update parameters between the devices, and then feeds the final result back to the agent.

The process of multi-agent training is show in Fig. 4(b), the system takes the state of multiple vehicles with the same starting point and ending point as input training and outputs the appropriate route planning for all vehicles, agent with these vehicles constitute the multi-agent for the same task. These agents share state information, but each agent trains its own parameters independently. The device participates in training is responsible for at least one single-agent training belong to the task. After each step of training, agents share the impact of their actions on routing to update the state. Then their rewards are calculated according to the state. Finally, the task is trained iteratively until the reward value no longer increases significantly. The perform detail of DLRD is shown in Algorithm 1.

The initial learning model and training parameters required for vehicle routing decision can be obtained by training all different route plans in advance, and updated through online distributed collaborative learning during the real-time planning process.

VI. Performance Analysis

This section evaluates the proposed DLRD algorithm by using VISSIM. 100 road segments are set in the simulation. The number of vehicles change from 200 to 2500, the routing decision is performed for 70% of these vehicles. The main

Algorithm 1 Distributed-Learning-Based Vehicle Routing Decision Algorithm

Initialization: Initialize $Q_p(s,a)$ with random weight θ_p of all agents, the state $s = s_0$ where all vehicles that initiated the task at the starting point;
for k in $(0, \ldots, \widetilde{N_i})$ **do**
  for v_p in $\widetilde{V}$ **do**
    Select a random action a_p^k with probability ϵ, otherwise $a_p^k = \max_{a_p} Q_p(s_k, a_p; \theta_p)$;
  end for
  Observe the system reward $R_i(s_{k+1})$ and new state s_{k+1};
  for v_p in $\widetilde{V}$ **do**
    Minimize the loss $L(\theta_p)$ according to equation 23 and update θ_p;
  end for
end for

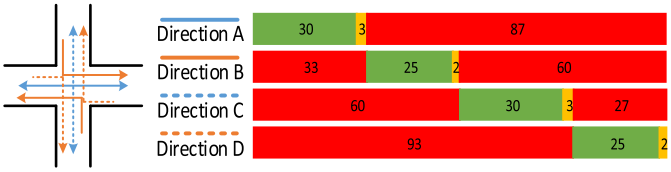


Fig. 5. Traffic light setting at cross-shaped intersections.

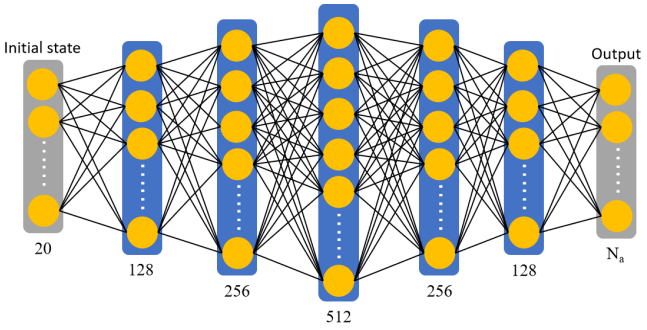


Fig. 6. Network structure for training.

purpose of the experiment is to alleviate the further traffic congestion caused by the random driving of 2500 vehicles by planning the route of 2500 vehicles. Vehicle routing decision with the same starting point and ending point is a multi-agent reinforcement learning task. During the learning process, the movement of the vehicle leads to real-time adjustment of the routing decision requirements, and each agent learning device set is obtained based on software defined device collaborative organization optimization method. Driving speed of the vehicle is set between 35-60km/h. The parameter $\mu = N_y/S_y$ is used to judge whether the road is congested, where N_y and S_y represent the number of vehicles on and the area of road segment y. The road intersections are all cross-shaped intersections. The traffic light setting include four phases is shown in Fig. 5.

The network structure used by training for one of the agents of multi-agent learning in routing decision is shown in Fig. 6. The input is a one-dimensional state vector of length 100,

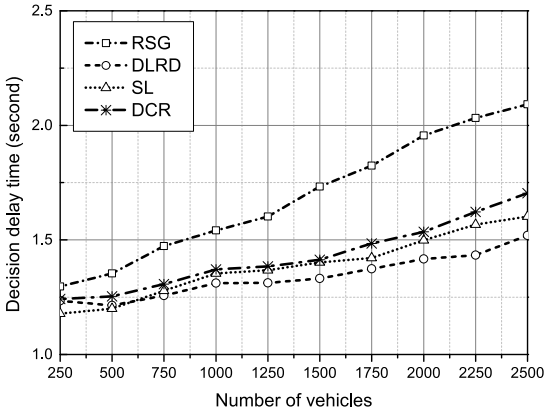


Fig. 7. The decision delay time of four algorithms.

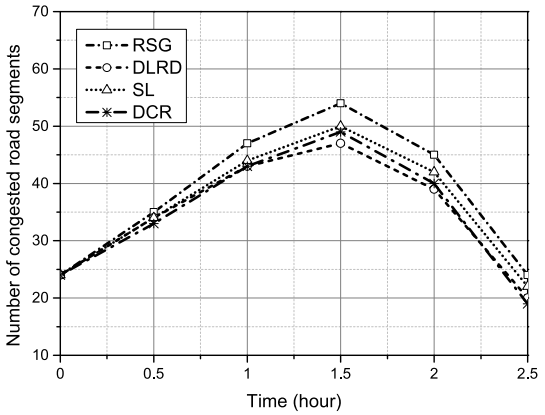


Fig. 8. The number of congested road segments.

representing the distribution of vehicles on 100 road segments. The middle layer of the network include five fully connected layers, and the number of neurons is 128, 256, 512, 256 and 128, respectively. Each fully connected layer is followed by an activation function ReLU. The output is a vector of length N_a, representing N_a Q value results. γ and α are all set to 0.9. The initial value of ϵ in the $\epsilon - greedy$ strategy is 0.5, and it decreases by 0.01 for every 1000 training steps until it drops to 0.1.

This article evaluates the proposed DLRD algorithm in terms of decision delay time and congestion control capabilities. In the experiment, the DLRD is compared with reverse Stackelberg approach algorithm (RSG) [39], stage learning algorithm (SL) [29] and distributed cooperative routing algorithm (DCR) [40]. In RSG, the reverse Stackelberg game is utilized to model vehicle routing, and the systematic optimal vehicle routing is achieved according to the predicted traffic conditions. The SL performs vehicle routing decision by quantifying the bounded rationality of the driver as a parameter for multi-agent learning in an edge-assisted distributed routing framework. The DCR models cooperative vehicle routing decision as a population game and then use evolutionary game theory combined with edge computing to solve the problem.

Fig. 7 shows the decision delay time of four algorithms, which is the average waiting time of the vehicle from initiating the request to obtaining the decision result. It should be noted that this online decision delay is based on previous training results instead of a complete training process from scratch. The comparison is done by changing the number of vehicles, while the routing decision is performed for 70% of them. It can be found that with the increase number of agents (vehicles that initiate the task), although the training time of all algorithms increases, the decision time of DLRD is significantly less than others. Moreover, the growth rate of time caused by the increase number of vehicles in DLRD is also the smallest.

In order to evaluate the control ability of traffic conditions through different algorithms, traffic congestion and the average waiting time are used as measurement standards. Assuming that when $\mu > 0.6$, the road is congested. The traffic congestion is tested by performing routing decision for 2500 vehicles, while adding vehicles to the road at a flow of 40 vehicles/min. In addition, a 2.5-hour simulation test is conducted. In the

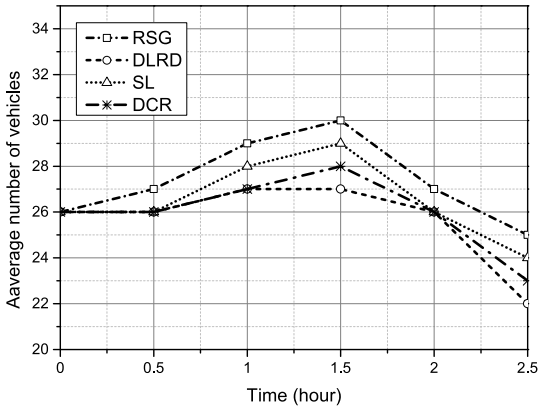


Fig. 9. The average number of vehicles on the congested road segments.

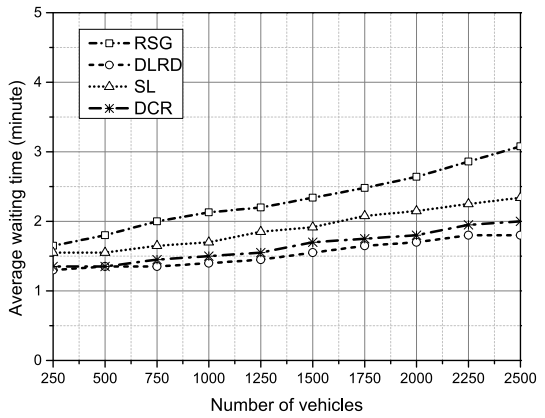


Fig. 10. The average awaiting time at intersection.

first hour, all test vehicles are on the road, and the peak of congestion reaches around 1.5 hours. The number of congested road segments is shown in Fig. 8. It can be seen that the DLRD algorithm has the least number of congested road segments compared to the other two algorithms. In addition, Fig. 9 shows the average number of vehicles on the congested road segments of these algorithms, and DLRD still has the smallest number of vehicles, which means that even on congested routes, DLRD can reduce the degree of congestion.

Fig. 10 shows the vehicles' average waiting time at intersection with the number of vehicles change from 200 to 2500.

It can be seen that DLRD has the least waiting time. This means that the DLRD can effectively relieve traffic congestion.

VII. Conclusion

This article studied the vehicle routing decision in SDIoV to relieve traffic congestion, recently intensified by the rapid growth of vehicles number. In detail, we first proposed the edge intelligence based SDIoV architecture to alleviate the pressure of cloud centralized computing and to reduce the delay caused by transferring data to the remote cloud. First, the edge intelligence based SDIoV architecture is proposed to alleviate the pressure of cloud centralized computing and reduce the delay caused by transferring data to the remote cloud. Then the multi-agent deep reinforcement learning for vehicle routing decision is introduced, which treats the vehicle that initiates the routing decision as an agent and vehicles requests with the same starting point and ending point constitute the multi-agent of the same task. Then, a software defined collaboration optimization method is designed to speed up the distributed training process. Finally, the DLRD algorithm is designed with combination of edge based deep Q-learning for real-time routing decision. The simulation results show that the proposed DLRD has excellent performance in ease traffic congestion and can effectively reduce the training time. Future work will consider how to accelerate distributed learning to get a faster vehicle planning decision response.

References

[1] A. Thakur and R. Malekian, "Fog computing for detecting vehicular congestion, an Internet of vehicles based approach: A review," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 2, pp. 8–16, Summer 2019.

[2] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive Internet of vehicles," *Comput. Commun.*, vol. 120, pp. 58–70, May 2018.

[3] K. Lin, J. Luo, L. Hu, M. S. Hossain, and A. Ghoneim, "Localization based on social big data analysis in the vehicular networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1932–1940, Aug. 2017.

[4] J. Yin and W. Fu, "A hybrid path planning algorithm based on simulated annealing particle swarm for the self-driving car," in *Proc. Int. Comput., Signals Syst. Conf. (ICOMSSC)*, Sep. 2018, pp. 696–700.

[5] M. Chen and Y. Hao, "Label-less learning for emotion cognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2430–2440, Jul. 2020.

[6] K. Lin, C. Li, G. Fortino, and J. J. P. C. Rodrigues, "Vehicle route selection based on game evolution in social Internet of vehicles," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2423–2430, Aug. 2018.

[7] A. Abdelrahman, H. S. Hassanein, and N. Abu-Ali, "IRouteSafe: Personalized cloud-based route planning based on risk profiles of drivers," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[8] A. Adiththan, S. Ramesh, and S. Samii, "Cloud-assisted control of ground vehicles using adaptive computation offloading techniques," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 589–592.

[9] K. Lin, M. Chen, J. Deng, M. M. Hassan, and G. Fortino, "Enhanced fingerprinting and trajectory prediction for IoT localization in smart buildings," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1294–1307, Jul. 2016.

[10] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, "Agent-oriented cooperative smart objects: From IoT system design to implementation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 11, pp. 1939–1956, Nov. 2018.

[11] R. Casadei, G. Fortino, D. Pianini, W. Russo, C. Savaglio, and M. Viroli, "Modelling and simulation of opportunistic IoT services with aggregate computing," *Future Gener. Comput. Syst.*, vol. 91, pp. 252–262, Feb. 2019.

[12] Y. Zhang, R. Wang, M. S. Hossain, M. F. Alhamid, and M. Guizani, "Heterogeneous information network-based content caching in the Internet of vehicles," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10216–10226, Oct. 2019.

[13] V. T. N. Nha, S. Djahel, and J. Murphy, "A comparative study of vehicles' routing algorithms for route planning in smart cities," in *Proc. 1st Int. Workshop Veh. Traffic Manage. Smart Cities (VTM)*, Nov. 2012, pp. 1–6.

[14] S. Pothan, J. L. Nandagopal, and G. Selvaraj, "Path planning using state lattice for autonomous vehicle," in *Proc. Int. Conf. Technol. Advancements Power Energy (TAP Energy)*, Dec. 2017, pp. 1–5.

[15] A. Vagner, "Intelligent route planning system for car drivers in a city," in *Proc. 6th IEEE Int. Conf. Cognit. Infocommun. (CogInfoCom)*, Oct. 2015, pp. 551–555.

[16] C. Zhao, M. Dong, K. Ota, J. Li, and J. Wu, "Edge-mapreduce-based intelligent information-centric IoV: Cognitive route planning," *IEEE Access*, vol. 7, pp. 50549–50560, 2019.

[17] J. Duan, H. Ge, and J. Yao, "Implementation of path planning method of intelligent vehicle based on MapX," in *Proc. 26th Chin. Control Decis. Conf. (CCDC)*, May 2014, pp. 4720–4724.

[18] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[19] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications," *Future Gener. Comput. Syst.*, vol. 90, pp. 149–157, Jan. 2019.

[20] M. Chen, Y. Miao, H. Gharavi, L. Hu, and I. Humar, "Intelligent traffic adaptive resource allocation for edge computing-based 5G networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 2, pp. 499–508, Jun. 2020.

[21] Y. Zhang, X. Ma, J. Zhang, M. S. Hossain, G. Muhammad, and S. U. Amin, "Edge intelligence in the cognitive Internet of Things: Improving sensitivity and interactivity," *IEEE Netw.*, vol. 33, no. 3, pp. 58–64, May 2019.

[22] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.

[23] X. Xu, Y. Xue, X. Li, L. Qi, and S. Wan, "A computation offloading method for edge computing with vehicle-to-everything," *IEEE Access*, vol. 7, pp. 131068–131077, 2019.

[24] S. Garg, A. Singh, S. Batra, N. Kumar, and L. T. Yang, "UAV-empowered edge computing environment for cyber-threat detection in smart vehicles," *IEEE Netw.*, vol. 32, no. 3, pp. 42–51, May 2018.

[25] G. Luo *et al.*, "Software defined cooperative data sharing in edge computing assisted 5G-VANET," *IEEE Trans. Mobile Comput.*, early access, Nov. 12, 2020, doi: 10.1109/TMC.2019.2953163.

[26] Q. Qi *et al.*, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.

[27] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, early access, Jan. 10, 2020, doi: 10.1109/TMC.2020.2965929.

[28] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani, "Deep features learning for medical image analysis with convolutional autoencoder neural network," *IEEE Trans. Big Data*, early access, Jun. 20, 2017, doi: 10.1109/TBDATA.2017.2717439.

[29] B. Zhu, J. Li, Q. Yuan, J. Lu, and S. Yang, "An edge-assisted vehicle routing method based on game-theoretic multiagent learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 462–469.

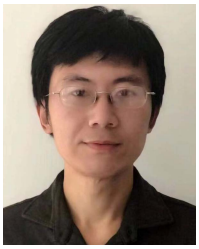
[30] M. Mynuddin and W. Gao, "Distributed predictive cruise control based on reinforcement learning and validation on microscopic traffic simulation," *IET Intell. Transp. Syst.*, vol. 14, no. 5, pp. 270–277, May 2020.

[31] Z. Ju, H. Zhang, and Y. Tan, "Distributed deception attack detection in platoon-based connected vehicle systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4609–4620, May 2020.

[32] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for Web attack detection on edge devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1963–1971, Mar. 2020.

[33] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, D. Niyato, and D. I. Kim, "Distributed deep learning at the edge: A novel proactive and cooperative caching framework for mobile edge networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1220–1223, Aug. 2019.

[34] M. Langer, A. Hall, Z. He, and W. Rahayu, "MPCA SGD—A method for distributed training of deep learning models on spark," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2540–2556, Nov. 2018.
[35] K. Lin, Y. Li, J. Deng, P. Pace, and G. Fortino, "Clustering-learning-based long-term predictive localization in 5G-envisioned Internet of connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 5, 2020, doi: 10.1109/TITS.2020.2997472.
[36] M. L. Puterman, "Markov decision processes: Discrete stochastic dynamic programming," *J. Oper. Res. Soc.*, vol. 46, no. 6, p. 792, 1995.
[37] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.
[38] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: http://arxiv.org/abs/1701.07274
[39] N. Groot, G. Zaccour, and B. D. Schutter, "Hierarchical game theory for system-optimal control: Applications of reverse Stackelberg games in regulating marketing channels and traffic routing," *IEEE Control Syst. Mag.*, vol. 37, no. 2, pp. 129–152, Apr. 2017.
[40] J. Lu, J. Li, Q. Yuan, and B. Chen, "A multi-vehicle cooperative routing method based on evolutionary game theory," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 987–994.



Kai Lin (Senior Member, IEEE) received the M.S. and Ph.D. degrees in communication engineering from Northeastern University, China. He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology. His research interests include wireless communications, big data analysis, and the Internet of Things (IoT). He is an Associate Editor of *Recent Advances in Communications and Networking Technology* and editor of several journals.

Chensi Li received the B.S. degree in software engineering from Dalian Maritime University, China. She is currently pursuing the M.S. degree in computer application technology with the School of Computer Science and Technology, Dalian University of Technology, China. Her research interests include social networks and vehicular networks.

Yihui Li received the B.S. degree in the Internet of Things engineering from Xidian University, China, in 2018. He is currently pursuing the M.S. degree in computer software and theory with the School of Computer Science and Technology, Dalian University of Technology. His current research interests include big data analysis, 5G, the Internet of Things (IoT), and network spectrum allocation.

Claudio Savaglio (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Calabria in 2010, 2013, and 2018, respectively. In 2013, he was a Visiting Researcher with The University of Texas at Dallas, TX, USA, the New Jersey Institute of Technology, NJ, USA, in 2016, and the Universitat Politecnica de Valencia, Valencia, Spain, in 2017. He is currently a Post-Doctoral Researcher. His research interests include the Internet of Things, edge computing, network simulation, and agent-oriented middleware and development methodologies.

Giancarlo Fortino (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Calabria (Unical), Italy, in 2000. He is currently a Full Professor of computer engineering with the Department of Informatics, Modeling, Electronics, and Systems, Unical. He is also a Guest Professor with the Wuhan University of Technology, China, a High-End Expert with HUST, China, and a Senior Research Fellow with the ICAR-CNR Institute. He is the Director of the SPEME Lab, Unical, and the Co-Chair of Joint labs on the IoT established between Unical and WUT and SMU Chinese universities. His research interests include agent-based computing, wireless (body) sensor networks, and the IoT. He is the author of more than 400 articles in international journals, conferences, and books. He is a member of the IEEE SMCS BoG and the IEEE Press BoG and the Chair of the IEEE SMCS Italian Chapter. He is the (Founding) Series Editor of the IEEE Press Book Series on Human-Machine Systems, the EiC of Springer Book Series on Internet of Things, and AE of many international journals, such as the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, the IEEE INTERNET OF THINGS JOURNAL, the IEEE SYSTEMS JOURNAL, the IEEE SMCM, *Information Fusion*, JNCA, EAAI, and so on. He is the Co-Founder and the CEO of SenSysCal S.r.l., a Unical spinoff focused on the innovative IoT systems.