A Joint Service Migration and Mobility Optimization Approach for Vehicular Edge Computing

Quan Yuan, *Member, IEEE*, Jinglin Li, *Member, IEEE*, Haibo Zhou, *Senior Member, IEEE*, Tao Lin, Guiyang Luo, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—The vehicular edge computing is considered an enabling technology for intelligent and connected vehicles since the optimization of communication and computing on edge has a significant impact on driving safety and efficiency. In this paper, with the road traffic assignment to "proactively" reshape the spatiotemporal distribution of resource demands, we investigate the joint service migration and mobility optimization problem for vehicular edge computing. The goal is to meet the service delay requirements of vehicular edge computing with minimum migration cost and travel time. As service migration and mobility optimization are coupled, the joint scheduling problem suffers from the curse of dimensionality, which cannot be solved in real time by centralized algorithms. To this end, a multi-agent deep reinforcement learning (MADRL) algorithm is proposed to maximize the composite utility of communication, computing, and route planning in a distributed way. In the MADRL algorithm, a two-branch convolution based deep *Q*-network is constructed to coordinate migration action and routing action. Extensive experimental results show that the proposed algorithm is scalable and substantially reduces service delay, migration cost and travel time as compared with the existing baselines.**

***Index Terms*—Vehicular edge computing, service migration, mobility optimization, multi-agent deep reinforcement learning.**

I. Introduction

INTELLIGENT and connected vehicles (ICVs) are playing an important role in realizing safe, efficient, comfortable, and energy-saving driving. With the fifth generation (5G) and beyond wireless networks coming up, vehicles can take advantage of their on-board intelligence as well as the edge intelligence to make real-time driving decisions [1], [2]. By offloading compute-intensive tasks (e.g., camera data processing, and dynamic route planning) from vehicles to edge servers, the edge

Manuscript received February 9, 2020; revised April 23, 2020; accepted May 31, 2020. Date of publication June 3, 2020; date of current version August 13, 2020. This work was supported in part by the Natural Science Foundation of China under Grants 61902035, 61871211, and 61876023, in part by the Natural Science Foundation of Beijing under Grant 4181002, and in part by China Postdoctoral Science Foundation under Grant 2019M660565. The review of this article was coordinated by Dr. P. Lin. *(Corresponding author: Jinglin Li.)*

Quan Yuan, Jinglin Li, Tao Lin, and Guiyang Luo are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: yuanquan@bupt.edu.cn; jlli@bupt.edu.cn; lintaomffl@gmail.com; luoguiyang@bupt.edu.cn).

Haibo Zhou is with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China (e-mail: haibozhou@nju.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2020.2999617

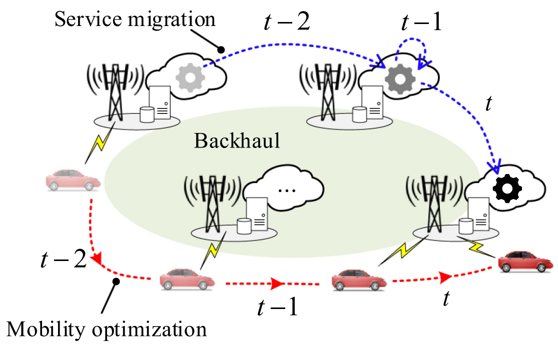

Fig. 1. Joint service migration and mobility optimization.

computing will largely enhance the computational capacity of vehicles [3]. Due to an ever-increasing quantity of ICVs that scramble for edge computing, the sophisticated scheduling of communication and computing resources becomes critical to driving safety and efficiency [4].

Considering the high mobility of vehicles, service migration [5], [6] performs multi-dimensional resource scheduling across edge clouds to ensure service delay and maintain service continuity. However, these existing methods "reactively" schedule edge computing resources based on predicted mobility patterns, which can hardly meet the quality of service (QoS) requirements of intelligent vehicles. On one hand, the already deployed base stations (BSs) and edge servers cannot always accommodate the offloaded tasks of associated vehicles. The service delay cannot be guaranteed when the demand for edge computing exceeds the capacity of nearby edge resources. On the other hand, the dynamic cooperative route planning of intelligent vehicles may break the predicted mobility patterns, which disrupts the scheduling of edge resources. Therefore, the service migration and mobility optimization are coupled for edge-assisted intelligent driving: 1) The road traffic assignment has a strong influence on the spatiotemporal distribution of offloaded tasks; and 2) Conversely, the resource scheduling of service migration determines the correctness and timeliness of vehicles' driving and routing[1] decisions. Only by jointly optimizing the *communication*, *computing*, and *routing*, can the quality of intelligent driving be largely improved.

As shown in Fig. 1, we jointly consider the service migration and mobility optimization for edge-assisted intelligent driving,

[1]In this paper, routing refers to the route planning of vehicles.

which can "proactively" reshape the distribution of resource demands and effectively schedule edge resources. Service migration and mobility optimization, which involve the resource scheduling in both information domain and transportation domain, have been studied separately in existing works [7], [8]. However, the joint optimization of these cross-domain resources will pose new challenges. First, the joint optimization suffers immensely from the curse of dimensionality. Specifically, the state space represents both the loads of edge clouds and the status of road traffic, while the action space increases exponentially with the number of vehicles. It is difficult to find a natural decomposition of this joint optimization problem. Second, the actions of service migration and vehicle mobility have long-term consequences, so the snapshot-based scheduling cannot achieve an optimal solution. To deal with these challenges, by decomposing the state observations and actions of a monolithic centralized controller into multiple simpler agents (i.e., the resource scheduler for each vehicle), we exploit multi-agent deep reinforcement learning (MADRL) to maximize the long-term composite utility of communication, computing, and routing. With MADRL, distributed agents can cooperatively make service migration and mobility optimization decisions based on individual observation and learned policy. The contributions of this paper are summarized as follows:

- A multi-agent-based framework is proposed for vehicular edge computing. Within the framework, the cooperative agents can help their respective vehicles to perform joint service migration and mobility optimization.
- The joint optimization of communication, computing, and routing is formulated to minimize the system cost while satisfying the service delay requirements of vehicles, under the constraints of limited communication resource, computing power, and road capacity.
- We propose an MADRL algorithm for the joint optimization problem to deal with the curse of dimensionality. A two-branch convolution based deep Q-network, which serves as the agent network, is constructed to coordinate the migration and routing actions.

The remainder of this paper is organized as follows. In Section II, recent research works on edge resource scheduling and road traffic optimization are reviewed. The system model and problem formulation are presented in Section III. Then, Section IV gives the overview of MADRL. Next, the MADRL-based joint service migration and mobility optimization approach is elaborated in Section V. The performance of the proposed method is evaluated in Section VI. Finally, the work is concluded in Section VII.

II. Related Work

A. Low-Dimensional Resource Scheduling

In this subsection, we review recent studies that scheduled no more than two kinds of resources in edge cloud. Vehicle-to-Everything (V2X) [9], [10] communication is a key enabling technology for intelligent driving, so resource allocation in heterogeneous vehicular networks has been well studied to ensure its high efficiency [11]. In heterogeneous wireless access networks, the radio resource slicing [12] can be used to provide QoS isolation among different services for ICVs. In addition, Wu *et al.* [13] scheduled communication and computing jointly to guarantee the quality of computation offloading. Zhou *et al.* [14] optimized communication and caching jointly to reduce the data access delay. These methods optimize resource scheduling in one edge cloud, so they can hardly utilize multiple edge clouds to achieve load balance.

The resource scheduling across edge clouds creates new opportunities for improving the cost effectiveness of edge computing. In [15], Chen *et al.* investigated the computation offloading with the coordination of multiple edge clouds. The authors decomposed the original scheduling problem into two sub-problems, i.e., task placement and resource allocation. Furthermore, Chen *et al.* [16] focused on the workload balancing among edge clouds. Considering the spatiotemporal stochasticity of the workload, they utilized the Lyapunov optimization to jointly schedule radio access and computation offloading. These methods have effectively scheduled multi-access edge computing, and can be further enhanced by considering user mobility. Taking the radio handover and computation migration into consideration, Sun *et al.* [17] investigated the dynamic matching between tasks and edge clouds using the Lyapunov optimization and the multi-armed bandit. Some studies took advantage of service migration to guarantee QoS with the impact of user mobility. Taleb *et al.* [5] used value iteration for the Markov decision process (MDP) to perform cost-effective service migrations. Ouyang *et al.* [6] proposed a distributed approximation scheme to optimize service migration under long-term cost budget constraint. With the observation of user behavior and network status, Chen *et al.* [7] proposed a reinforcement learning based dynamic service migration method.

The low-dimensional resource scheduling is not enough to fully empower intelligent vehicles to effectively understand environment and plan actions, so it should be extended to a higher dimension.

B. High-Dimensional Resource Scheduling

In this subsection, we review recent studies that jointly scheduled communication, computing and caching. This high-dimensional resource scheduling is crucial to the quality of intelligent driving [18], but suffers immensely from the curse of dimensionality. To deal with this challenge, some studies leveraged the software-defined networks (SDN), network function virtualization (NFV) and deep learning to optimize the resources [19]–[21]. Based on the vehicle mobility model, Li *et al.* [22] modeled the multi-dimensional resource scheduling as a partially observable Markov decision process (POMDP), which is solved by value iteration. He *et al.* [23] used deep reinforcement learning (DRL) to optimize the virtualized networking, caching, and computing resources. They used a trial-and-error method to find the scheduling sequence to maximize the long-term performance. Tan *et al.* [24] proposed a multi-timescale deep learning framework, which chooses candidate resources based on the vehicle mobility, and determines resource allocation based on the real-time wireless channel quality. However, these centralized DRL methods have difficulty in dealing with the system complexity which grows exponentially with the number of vehicles.

To deal with the huge state-action space, some studies adopted MADRL to make resource allocation decisions distributedly. Chen *et al.* [25] let each mobile user independently learn an efficient computation offloading policy, which can coordinately minimize the global cost of power consumption and buffering delay. In heterogeneous networks, Amiri *et al.* [26] modeled the power control problem as a multi-agent MDP. They took advantage of multi-agent tabular Q-learning to distributedly manage the interference. Furthermore, Nasir *et al.* [27] proposed a distributed dynamic power allocation algorithm based on multi-agent deep Q-learning, which is scalable to large networks. Similarly, Meng *et al.* [28] exploited multi-agent actor-critic deep deterministic policy gradient to perform power allocation in wireless networks.

These high-dimensional resource scheduling methods have well improved the quality of edge computing. However, the quality will deteriorate largely when the demand for edge computing exceeds the capacity of nearby edge resources. In this paper, we not only perform resource scheduling to improve resource utility, but also utilize route planning to reshape the spatiotemporal distribution of resource demands, which optimizes the quality of vehicular edge computing in a new dimension.

C. Road Traffic Optimization

In transportation systems, the traffic lights are usually optimized to improve the efficiency of road traffic. Abdoos *et al.* [29] trained an independent control policy for each traffic light using reinforcement learning. Furthermore, considering the coordination of traffic lights, Chu *et al.* [30] proposed an MADRL based traffic signal timing method. As for non-signalized intersections, Qian *et al.* [31] proposed an alternately iterative descent method to implement safe and efficient driving for autonomous vehicles. The intersection scheduling can reactively optimize traffic flow, while the cooperative route planning among vehicles can proactively optimize traffic assignment. Lin *et al.* [32] adopted social clustering and game evolution to select vehicle routes. The authors have proved that their vehicle route selection game can converge to Nash equilibrium. In [33], vehicles are trained through Q-learning to cooperatively plan routes. Some studies [34], [35] have used the cooperation between vehicles and road-side infrastructure to achieve vehicle routing. Liu *et al.* [34] utilized SDN and edge computing to jointly solve the charging station selection and route planning problem for electric vehicles. As for the trade-off between system optimum and user optimum, Groot *et al.* [35] used a reverse Stackelberg game between the road authority and vehicles to relieve congestion.

In this paper, we use route planning to reshape the distribution of edge resource demands while keeping the road traffic efficient. Existing route planning methods can hardly deal with this cross-domain problem, so a novel joint optimization method should be proposed.

TABLE I
SUMMARY OF NOTATIONS

$\mathcal{E}$	The set of BSs integrating edge servers, $e \in \mathcal{E}$.
$\mathcal{V}$	The set of intelligent vehicles, $v \in \mathcal{V}$.
$\mathcal{O}$	The set of service entities hosted on edge servers.
o_v	The exclusive service entity of v, $o_v \in \mathcal{O}$.
x_v^e	Indicator denoting whether v is associated with e.
y_v^e	Indicator denoting whether o_v is hosted on e.
K	The number of orthogonal resource blocks.
W	The bandwidth of each resource block.
E	The number of vehicles associated with a BS.
P	The baseline transmit power of vehicles.
N	Noise power.
h	Channel power gain.
l_v	The distance from v to its associated BS.
B_e	The backhaul bandwidth of e.
U_e	The computing capacity of e.
R_v	The uplink wireless transmission rate of v.
b_v	The offloaded task size of v.
u_v	The number of CPU cycles required by v.
τ_v	The total service delay of v.
τ	The upper bound of service delay.
τ_v^{back}	The backhaul delay of v.
τ_v^{comm}	The wireless transmission delay of v.
τ_v^{comp}	The computing delay of v.
c_v	The individual cost of v.
c_v^{mgt}	The migration cost of v.
c_v^{tfc}	The traffic cost of v.
s_v	The system state observed by v.
r_v	The immediate reward of v.
a_v	The service migration and route planing actions of v.

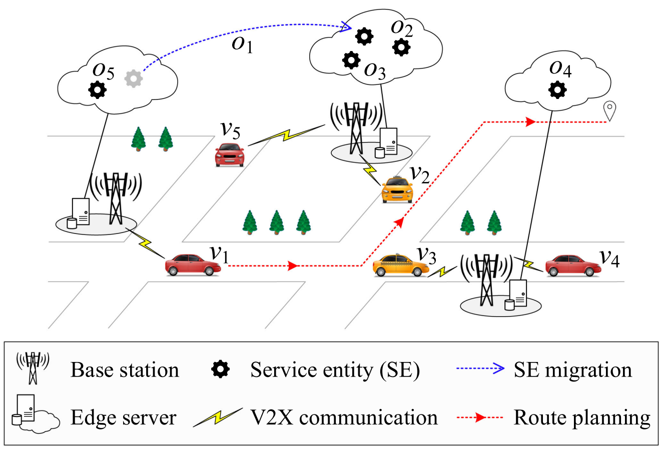

Fig. 2. The framework of edge-assisted intelligent driving.

III. System Model and Problem Formulation

In this section, we formulate the joint optimization problem of service migration and vehicle mobility. In Table I, some important notations are summarized for the ease of reference.

A. Network Model

As shown in Fig. 2, we consider an edge-assisted intelligent driving system that includes a set of BSs integrating edge servers $\mathcal{E}$, a set of intelligent vehicles $\mathcal{V}$, and a set of service entities (SEs) $\mathcal{O}$ hosted on edge servers. Let $e \in \mathcal{E}$ denote an edge server as well as its co-located BS. The edge servers are connected through backhaul links, which enables the load balancing among them. Each vehicle $v \in \mathcal{V}$ continuously connects to an always-on exclusive SE $o_v \in \mathcal{O}$ to achieve edge-assisted intelligent driving. The SE, defined as a bundle of individualized data of the vehicle/driver and the processing logic on the data, helps the vehicle to handle offloaded compute-intensive tasks such as camera data processing and dynamic route planning.

With the constraint on service delay, the serving SE should be placed on a local or a nearby edge server. Each vehicle sends service requests to the local edge server through associated BS, then the local edge server admits the requests if it hosts the corresponding SE, otherwise it forwards the requests through backhaul links to the edge server which hosts the corresponding SE. To maintain satisfactory service latency for vehicles, SEs should be dynamically migrated across edge servers to follow vehicle mobility. Besides, the resources in edge servers are usually managed by container-based virtualization and can thus be scheduled flexibly. Let $x_v^e(t) \in \{0,1\}$ denote whether vehicle $v \in \mathcal{V}$ is associated with BS and edge server $e \in \mathcal{E}$, and $y_v^e(t) \in \{0,1\}$ denote whether the SE of v is hosted on edge server e, in time slot t. In one time slot, each vehicle can only be associated with one BS, and its SE can only be hosted on one edge server, thus

$$\sum_{e\in\mathcal{E}} x_v^e(t) = 1, \forall v \in \mathcal{V},$$
$$\sum_{e\in\mathcal{E}} y_v^e(t) = 1, \forall v \in \mathcal{V}. \tag{1}$$

Backhaul Delay: The backhaul delay exists when the vehicle is served by a non-local edge server, and it is introduced by transmission, propagation, processing, and queuing. The transmission delay is denoted as $b_v(t)/B_e$, where $b_v(t)$ is the data size of vehicle v's offloaded task in time slot t, and B_e is the bandwidth of the outgoing link of the local edge server e. Here the transmission time of the computing result is ignored due to the small data size. In addition, the round-trip propagation, processing, and queuing delays are determined by the hop count between the associated edge server $e_1 \in \mathcal{E}$ and the serving edge server $e_2 \in \mathcal{E}$ of vehicle v. Therefore, the backhaul delay for computation offloading is given by

$$\tau_v^{back}(t) = \begin{cases} 0, & e_1 = e_2, \\ \frac{b_v(t)}{B_{e_1}} + 2\lambda d(e_1, e_2), & e_1 \neq e_2, \end{cases} \tag{2}$$

where $x_v^{e_1}(t) = 1$, $y_v^{e_2}(t) = 1$, $d(\cdot,\cdot)$ is the hop count between two edge servers, and λ is a positive coefficient.

Migration Cost: The migration of SE across edge servers incurs additional operating cost, e.g., the proactive replication of SE on the target edge server, and the release of resources on the source edge server. In this paper, it is considered that the migration cost is determined by the image size of each SE. Let $c_v^{mgt}(t)$ be the migration cost of moving the SE of vehicle v from edge server $e_1 \in \mathcal{E}$ to edge server $e_2 \in \mathcal{E}$ in time slot t,

$$c_v^{mgt}(t) = \begin{cases} 0, & e_1 = e_2, \\ \mu |o_v|, & e_1 \neq e_2, \end{cases} \tag{3}$$

where $y_v^{e_1}(t-1) = 1$, $y_v^{e_2}(t) = 1$, $|o_v|$ is the image size of v's SE, and μ is a positive coefficient.

B. Communication Model

The wireless communications significantly impact the quality of edge-assisted intelligent driving. A regular hexagonal deployment of BSs, with an orthogonal frequency-division multiple access (OFDMA) system, is focused in this paper. There are K orthogonal resource blocks without interference, and the bandwidth of each resource block is W. At each BS, the resource blocks are allocated averagely to the associated vehicles, and a resource block is allocated only to a single vehicle. The uplink transmission, which is used to upload the task data for edge computing, is the main bottleneck in radio access network. The uplink transmission rate achieved by vehicle v to its associated BS is given by

$$R_v = W \left\lfloor \frac{K}{E} \right\rfloor \log_2 (1 + \mathtt{SNR}_v), \tag{4}$$

where $\lfloor \cdot \rfloor$ is the floor function, E is the number of vehicles associated with the BS, and $\mathtt{SNR}_v$ is the received signal-to-noise ratio given by

$$\mathtt{SNR}_v = \frac{Phl_v^{\alpha(\epsilon-1)}}{N}, \tag{5}$$

P is the constant baseline transmit power of vehicles, h denotes the channel power gain, l_v is the distance from vehicle v to its associated BS, α is the path loss exponent, N is the noise power, and ϵ is the fractional channel inversion power control component.

Wireless Transmission Delay: For a computation offloading task, the average wireless transmission delay for v is

$$\tau_v^{comm}(t) = \frac{b_v(t)}{R_v}, \tag{6}$$

where the downlink transmission delay is omitted because the result data size is usually very small after processing on the edge server. Though a simplified communication model is used here, the model can be extended to multi-tier networks that consider inter-cell interference.

C. Computation Model

At each edge server, multiple SEs share the computing resource to help their serving vehicles process the offloaded tasks. The computing capacity of edge server e is denoted as U_e, measured by CPU cycles per second. It is assumed that the SEs hosted on edge server e share U_e evenly.

Computing Delay: Let $u_v(t)$ denote the number of CPU cycles required by the offloaded task of vehicle v in slot t. The computing delay on edge server is given by

$$\tau_v^{comp}(t) = \sum_{e\in\mathcal{E}} \frac{u_v(t) y_v^e(t)}{U_e / \sum_{v'\in\mathcal{V}} y_{v'}^e(t)}. \tag{7}$$

From the equation it can be seen that the computing delay on an edge server grows linearly with the number of SEs hosted on the edge server. As the number of SEs increases, the edge server may not provide satisfactory services for vehicles. The dynamic migration of SEs can balance the computing loads among edge servers, thus improving the QoS of intelligent driving.

D. Transportation Model

In this paper, the mobility of vehicles is modeled in a grid world [8]. Without loss of generality, the road network is partitioned into disjoint hexagonal grids in accordance with the deployment of BSs. Although the vehicles travel on the actual road network, the route planning is abstracted in a grid manner. Considering the multiple feasible link-level routes between two adjacent grids, we assume that the vehicles can autonomously plan link-level routes when next-hop grid is given. To this end, only the grid-level traffic conditions are leveraged for dynamic traffic assignment. Let $w^{g_1g_2}(t)$ denote the average travel time from g_1 to g_2 in time slot t, where g_1 and g_2 are adjacent grids. Specifically, $w^{g_1g_2}(t)$ is averaged over all vehicles that travel from g_1 directly into g_2 in time slot t, no matter what particular link-level route each vehicle has selected.

Traffic Cost: Let w_v be the total travel time of v between its origin-destination (O-D). And w_v is calculated as the summation of a sequence of $w^{g_1g_2}(t)$ along v's actual trajectory. Based on w_v, we define traffic cost to evaluate the overall performance of v's route planning actions,

$$c_v^{tfc}(t) = \begin{cases} w_v, & v \text{ finishes its trip in } t, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The traffic cost $c_v^{tfc}(t)$ can be obtained only when v finishes the trip, and it is the delayed and accumulative consequence of previous route planning actions.

E. Problem Formulation

To improve the composite quality of edge-assisted intelligent driving, the following problems are jointly optimized.

- Service migration: With the mobility of vehicles, it decides whether and where to migrate the SE. Considering the status of communication and computing resources, service migration is aimed at meeting the requirement of service delay and reducing the migration cost.
- Mobility optimization: For each vehicle, it generates an appropriate route to get to the destination. The cooperative route planning will not only increase transport efficiency but also optimize the spatiotemporal distribution of computation offloading tasks.

By jointly optimizing the service migration and vehicle mobility, *the objective of this paper is to minimize the system cost while satisfying the service delay requirements of vehicles*. The total service delay for vehicle v consists of wireless transmission delay, backhaul delay, and computing delay,

$$\tau_v(t) = \tau_v^{comm}(t) + \tau_v^{back}(t) + \tau_v^{comp}(t). \tag{9}$$

Let τ denote the upper bound of service delay. Therefore, $\tau_v(t) \leq \tau, \forall v \in \mathcal{V}$ should be achieved to ensure the quality of computation offloading. In addition, individual cost $c_v(t)$ is defined as the weighted sum of the migration cost and traffic cost incurred by vehicle v,

$$c_v(t) = (1-\omega)c_v^{mgt}(t) + \omega c_v^{tfc}(t), \tag{10}$$

where ω is the weight parameter. Therefore, the system cost for all vehicles in time slot t is

$$c(t) = \sum_{v\in\mathcal{V}} c_v(t). \tag{11}$$

It is needed to learn the optimal policy that minimizes the *long-term* system cost while satisfying the service delay requirements.

IV. Multi-Agent Deep Reinforcement Learning

A. MDP and Deep Q-Learning

In reinforcement learning, the environment is formulated as an MDP. A discounted MDP is represented as $(\mathcal{S}, \mathcal{A}, \mathtt{P}, \mathtt{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are environment states and system actions, respectively. For current state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, and next state $s' \in \mathcal{S}$, the probability of each state transition is defined as $\mathtt{P}(s'|s,a) \colon \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$; and the immediate reward is derived from $\mathtt{R}(s,a,s') \colon \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$. In addition, $\gamma \in [0,1]$ is the discount factor used to balance immediate and long-term reward. For the MDP, $\pi(s,a) \colon \mathcal{S}\times\mathcal{A} \to [0,1]$ is a policy which gives the probability of the agent taking action a when in state s. The best solution to the MDP is the optimal policy denoted as $\pi^*(s,a) \in \{0,1\}$.

Q-learning is a well-established model-free method, which enables an agent to learn the optimal policy through continual interactions (i.e., observations, actions and rewards) with the environment. To this end, $Q \colon \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is defined to evaluate the quality of a state-action combination. The Q-function for a specific π, is the expected accumulative rewards if the agent chooses action a in state s and utilizes π to generate follow-up actions,

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty}\gamma^i r(i+t)\big|s(t)=s, a(t)=a\right], \tag{12}$$

where r is the immediate reward received according to $\mathtt{R}$. Q-learning aims at finding the optimal policy that achieves the largest accumulative reward. To this end, an optimal Q-function can be given

$$Q^*(s,a) = \max_\pi Q^\pi(s,a). \tag{13}$$

Thus, the optimal policy π^* is any policy whose action in state s is $\arg\max_a Q^*(s,a)$. By leveraging the temporal-difference recursive equation, Q-function is updated as

$$\begin{aligned} Q_{k+1}(s,a) &= Q_k(s,a) \\ &\quad + \beta\left(r + \gamma\max_{a'} Q_k(s',a') - Q_k(s,a)\right), \end{aligned} \tag{14}$$

where $s' \in \mathcal{S}$ represents the state at the next time step, and β is the learning rate. $Q_k(s,a)$ can definitely converge to $Q^*(s,a)$ when a proper β is used.

Deep Q-learning uses a neural network to learn $Q^*(s,a)$. Thus it is more applicable to high-dimensional environment with large and complex state spaces. The neural network is called deep Q-network (DQN) and represented as $Q(s,a;\theta)$, where θ stands for network weights. The DQN can be trained by minimizing the following loss function

$$\mathcal{L}(\theta) = \mathbb{E}\left[(z - Q(s,a;\theta))^2\right], \tag{15}$$

where $z = r + \gamma \max_{a'} Q(s', a'; \bar{\theta})$ is the target Q-value, and the weights $\bar{\theta}$ get updated to θ slowly and periodically. However, the regular DQN may overestimate the action values, resulting in unstable training. To relieve this problem, double DQN [36] decouples the action selection from the calculation of target Q-value, and it calculates the target Q-value as $z = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \bar{\theta})$.

B. Multi-Agent Deep Reinforcement Learning

Though deep reinforcement learning can adapt to environment with complicated state spaces, a single super agent can hardly learn large-scale decentralized policies, whose joint action spaces grow exponentially with the quantity of actors. Decomposing a single monolithic agent into multiple simpler agents can reduce the dimensionality of the state and action spaces. Therefore, MADRL overcomes the scalability issue, which takes advantage of efficient communication and cooperation among agents to solve complex tasks. In MADRL, the problem of cooperation is a decentralized partially observable Markov decision process (Dec-POMDP). A Dec-POMDP is defined by the tuple $(\mathcal{V}, \mathcal{S}, \{\mathcal{A}_v\}, \{\Omega_v\}, \mathtt{P}, \mathtt{R}, \mathtt{O})$, where $\mathcal{V}$ and $\mathcal{S}$ represent agents and states, respectively. In addition, $\{\mathcal{A}_v\}$ denotes the possible actions taken by $v \in \mathcal{V}$, and $\{\Omega_v\}$ denotes the possible observations of v about $\mathcal{S}$. Finally, $\mathtt{P}$ is the joint state transition probability function, $\mathtt{R}$ is the joint reward function, and $\mathtt{O}$ is the joint observation function. In the reinforcement learning setting, we have no knowledge of $\mathtt{P}$, $\mathtt{R}$, and $\mathtt{O}$, which should be modeled by the interactions between agents and environment.

MADRL is complex because all agents potentially interact with each other and concurrently learn their respective policies. Independent Q-learning (IQL) [37] is a straightforward and commonly used method for multi-agent learning. With IQL, each agent regards the behavior of others as a component of environment dynamics, and then trains its network parameters independently. Since the learned policies of some agents can be efficiently transferred to other homogeneous agents, the IQL is completely scalable. In this paper, we take advantage of IQL to train cooperative agents for the joint optimization of service migration and vehicle mobility.

V. MADRL-Based Joint Service Migration and Mobility Optimization

In this paper, MADRL is exploited to cope with the curse of dimensionality in the joint service migration and mobility optimization problem. As a natural decomposition of the problem, each SE is regarded as an agent to maximize the accumulative utility of communication, computing, and routing. The joint optimization problem is formulated as a Dec-POMDP, whose system state, agent action, and reward model are defined in this section. Then a two-branch convolutional deep Q-network is designed and trained for agents to get the optimal policy.

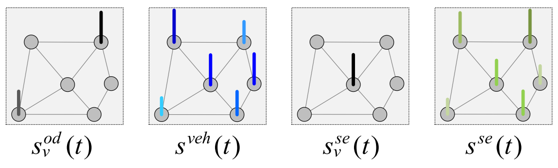

Fig. 3. Representation of system state.

A. State Space, Action Space and Reward Function

1) State Space: The joint state observed by vehicle v in slot t is defined as

$$s_v(t) = \left\{s_v^{od}(t), s^{veh}(t), s_v^{se}(t), s^{se}(t)\right\}, \tag{16}$$

where $s_v^{od}(t)$ denotes the current grid and destination grid of vehicle v, $s^{veh}(t)$ represents the number of vehicles in each grid, $s_v^{se}(t)$ is the edge server hosting v's SE, and $s^{se}(t)$ represents the number of SEs hosted by each edge server. It should be noted that $s^{veh}(t)$ indicates not only the traffic condition in each grid but also the load of each BS. Considering the spatial distribution of grids, we use multi-channel graph signals to represent the system state. The graph signals can form a unified representation of the system states for any network coverage (e.g., the hexagon for macro cell, and the Voronoi tessellation for small cell). In directed graph $G = (\mathcal{G}, \mathcal{T})$, each node in $\mathcal{G}$ represents a grid, and an edge exists in $\mathcal{T}$ if the corresponding grids are adjacent to each other. Specifically, $s_v^{od}(t) \in \mathcal{G}^2$, $s_v^{se}(t) \in \mathcal{G}$, $s^{veh}(t) \in \mathbb{N}^{|\mathcal{G}|}$, and $s^{se}(t) \in \mathbb{N}^{|\mathcal{G}|}$ are represented as graph signals. An example of the system state is shown in Fig. 3.

2) Action Space: In the system, every agent has to decide where to transfer its SE, and what the next grid for the vehicle to travel to. Therefore, the composite action of vehicle v in slot t is represented as

$$a_v(t) = \left\{a_v^{mgt}(t), a_v^{rt}(t)\right\}, \tag{17}$$

where $a_v^{mgt}(t)$ performs the SE transfer from $e \in \mathcal{E}$ to $e' \in \mathcal{E}$, i.e., $y_v^e(t-1) = 1$, $y_v^{e'}(t) = 1$; and $a_v^{rt}(t)$ gives the next grid $g \in \mathcal{G}$ which is adjacent to current grid. Besides, let $\mathcal{A}_v^{mgt}(t)$ and $\mathcal{A}_v^{rt}(t)$ be the set of all possible migration actions and routing actions, respectively. Thus, action $a_v(t)$ is valid if $a_v^{mgt}(t) \in \mathcal{A}_v^{mgt}(t)$ and $a_v^{rt}(t) \in \mathcal{A}_v^{rt}(t)$.

3) Reward Function: The reward function for vehicle v should minimize the individual cost $c_v(t)$ while meeting the required service delay τ. The immediate reward is expressed as

$$r_v(t) = -c_v(t) - d_v(t) + \tilde{r}_v(t), \tag{18}$$

where $d_v(t)$ denotes the penalty carried once the current service delay of v exceeds the upper bound,

$$d_v(t) = \begin{cases} C^-, & \tau_v(t) > \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

$C^- > 0$ is a constant penalty; and $\tilde{r}_v(t)$ represents the reward for v reaching its destination,

$$\tilde{r}_v(t) = \begin{cases} C^+, & v \text{ finishes its trip in } t, \\ 0, & \text{otherwise}, \end{cases} \tag{20}$$

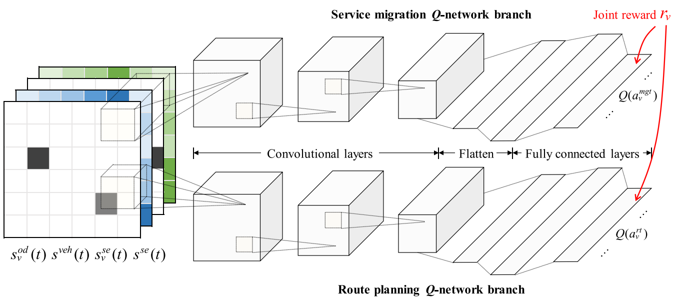
Service migration Q-network branch
Joint reward r_v
$Q(a_v^{mgt})$
Convolutional layers
Flatten
Fully connected layers
$Q(a_v^{rt})$
$s_v^{od}(t)$ $s^{veh}(t)$ $s_v^{se}(t)$ $s^{se}(t)$
Route planning Q-network branch

Fig. 4. The architecture of two-branch deep Q-Network.

$C^+ > 0$ is a large reward for v reaching its destination. The action of agent determines what reward will be gained, i.e., the agent gets an immediate reward $r_v(t)$ with respective observed state $s_v(t)$ and action $a_v(t)$ in time slot t. Each agent aims at learning the optimal policy which maximizes the accumulative reward, given by

$$\mathcal{R}_v(t) = \max \mathbb{E}\left[\sum_{i=0}^{T-1} \gamma^i r_v(i+t)\right], \tag{21}$$

where T denotes the look-ahead slots. By leveraging IQL, the system objective can be approached by achieving $\mathcal{R}_v(t)$ independently for every agent.

B. The Architecture of DQN

As DQN is an off-policy DRL algorithm, it is much more sample efficient than other on-policy DRL algorithms, particularly in this multi-agent environment. This is because each agent can be trained by reusing both its past experience and other agents' experience. Besides, DQN can be easily extended to plan multi-task actions, which is suitable for optimizing service migration and route planning simultaneously. Therefore, DQN is used as the agent network which maps the observed system state to the state-action values.

As a regular hexagonal grid model is considered, we leverage a convolutional neural network (CNN) to capture the spatial features from G. The observed state components $s_v^{od}(t)$, $s^{veh}(t)$, $s_v^{se}(t)$ and $s^{se}(t)$ are organized into 4-channel input features of DQN. As shown in Fig. 4, the observed state $s_v(t)$ is fed into convolutional layers followed by fully connected layers to calculate the state-action value for each composite action. A multi-task learning scheme is used to deal with the joint optimization problem, i.e., two network branches are designed to perform route planning and service migration, respectively. The network branch for route planning is fed with current location, destination location, traffic conditions, and the loads of base stations. These state components are selected because route planning aims to optimize traffic efficiency as well as the distribution of resource demands. Meanwhile, the network branch for service migration is fed with current location, destination location, traffic conditions, the loads of base stations, and the loads of edge servers, because the service migration should be optimized by considering the mobility of vehicles and the resource status of edge clouds. The joint reward in (18) is used to coordinate these two action branches, which will be elaborated in the following subsection.

As a result of homogeneity, each agent $v \in \mathcal{V}$ has the same copy of DQN in time slot t. If two or more vehicles are currently with the same O-D pair and their SEs are hosted on the same edge server, the observed state will be identical, so their following actions will be exactly the same. To deal with this phenomenon, a stochastic policy for the agent is used: the actions taken by the agent are drawn from a distribution derived by applying a `softmax` over the Q-values. In this way, the vehicles with the same state can probably take different actions, thus avoiding the bursty resource requests from vehicles.

C. The Training of DQN

We leverage the paradigm of centralized training and distributed execution to ease implementation. Within the multi-agent environment, all agents are assumed to take actions simultaneously, so each agent has no knowledge of other agents' current actions. As the agents are considered homogeneous, all their accumulative experiences are used to train a single DQN in a centralized way. This can effectively increase the amount of training data generated per step of the environment.

The complicated coupling of service migration and mobility optimization poses a big challenge on the training of multi-task DQN. As separate optimizations cannot achieve the system optimum, we exploit a joint reward instead of separate rewards to train these two branches. Specifically, the reinforcement learning environment for the service migration branch is defined as $s = \{s_v^{od}(t), s^{veh}(t), s_v^{se}(t), s^{se}(t)\}$, $a = a_v^{mgt}(t)$, $r = r_v(t)$; while the reinforcement learning environment for the route planning branch is defined as $s = \{s_v^{od}(t), s^{veh}(t)\}$, $a = a_v^{rt}(t)$, $r = r_v(t)$. Governed by joint reward $r_v(t)$, the service migration can be optimized by considering route planning, and vice versa. However, the dual exploration of these two branches is confronted with large action space, which decelerates the convergence to optimal policy. To overcome this limitation, we just train one network branch during a period of time, and keep the other branch fixed during this period. In this way, the migration policy of SEs is iteratively updated under a stable route planning policy, and then the routing policy of vehicles is iteratively updated under a stable service migration policy. The above procedure should be repeated until the training converges. The training algorithm is shown in Algorithm 1.

In the algorithm, we use superscript $\mathtt{F}$ and $\mathtt{L}$ to differentiate the fixed branch and the learning branch. For example, $Q(\cdot)^{\mathtt{L}}$ is the learning network branch, then $a_v^{\mathtt{L}}(t)$ is the action derived from $Q(\cdot)^{\mathtt{L}}$, and $\theta^{\mathtt{L}}$ is the network weights of $Q(\cdot)^{\mathtt{L}}$. In addition, double DQN mechanism is adopted: 1) a main network to learn the Q-values from agents; and 2) a separate target network which is updated periodically to compute target Q-values. Besides, experience replay is exploited to stabilize training by decorrelating the training examples in each batch to update the main network.

D. The Distributed Execution of DQN

After completing the training process, the model parameters of DQN are transferred to each agent on the edge server. When

Algorithm 1: Multi-agent deep Q-learning for joint service migration and mobility optimization.

```
Input:
  $\varepsilon$: exploration probability.
  $Y_1$: frequency of updating the target network.
  $Y_2$: frequency of exchanging fixed and learning branch.
1  Initialize main DQN $Q(s,a;\theta)$ with random weights $\theta$.
2  Initialize target DQN $Q(s,a;\bar{\theta})$ with weights $\bar{\theta} \leftarrow \theta$.
3  Initialize the learning branch $\mathtt{L} \leftarrow rt$, and the fixed branch
     $\mathtt{F} \leftarrow mgt$.
4  for each episode do
5      for each time slot $t = 1, 2, \ldots, T$ do
6          for each agent $v \in \mathcal{V}$ do
7              Select $a_v^{\mathtt{F}}(t) \leftarrow \arg\max_a Q\left(s_v(t), a; \theta\right)^{\mathtt{F}}$.
8              Execute action $a_v^{\mathtt{F}}(t)$ independently at $v$.
9              UpdateMainDQN$(\mathtt{L}, v, \theta, \bar{\theta})$.
10         Every $Y_1$ steps, update target DQN $\bar{\theta} \leftarrow \theta$.
11         Every $Y_2$ steps, exchange $\mathtt{F} \leftrightarrow \mathtt{L}$.

Procedure: UpdateMainDQN$(\mathtt{L}, v, \theta, \bar{\theta})$
1  if $rand() \leq \varepsilon$ then
2      Select random action $a_v^{\mathtt{L}}(t) \in \mathcal{A}_v^{\mathtt{L}}(t)$.
3  else
4      Select $a_v^{\mathtt{L}}(t) \leftarrow \arg\max_a Q\left(s_v(t), a; \theta\right)^{\mathtt{L}}$.
5  Execute action $a_v^{\mathtt{L}}(t)$ independently at vehicle $v$, obtain
     joint reward $r_v(t)$, and observe new state $s_v(t+1)$.
6  Store the tuple $\langle s_v(t), a_v^{\mathtt{L}}(t), r_v(t), s_v(t+1)\rangle$ into the
     experience replay buffer $\mathcal{B}^{\mathtt{L}}$.
7  Randomly sample a batch of tuples $\{\langle s_i, a_i, r_i, s_{i+1}\rangle\}_{i=1}^{I}$
     from the experience replay buffer $\mathcal{B}^{\mathtt{L}}$.
8  for each tuple $i = 1, 2, \ldots, I$ do
9      if episode terminates at step $i+1$ then
10         $z_i = r_i$.
11     else
12         $z_i = r_i + \gamma Q\left(s_{i+1}, \arg\max_{a'} Q\left(s_{i+1}, a'; \theta\right)^{\mathtt{L}}; \bar{\theta}\right)^{\mathtt{L}}$.
13 Perform a gradient descent step on $\mathcal{L}(\theta^{\mathtt{L}})$ with respect to
     the learning branch weights $\theta^{\mathtt{L}}$, where loss function is
     $\mathcal{L}(\theta^{\mathtt{L}}) = \frac{1}{I}\sum_{i=1}^{I}\left[z_i - Q(s_i, a_i; \theta)^{\mathtt{L}}\right]^2$.
```

each vehicle travels into a new grid, its agent uses the DQN to perform service migration and plan vehicle routing based on the observation of system state. The state observation of each agent is formed from the total effect of the overall population and its individual status. In this way, the interactions between single agent and the population are used to approximate the complex interactions among agents. Specifically, the individual status $s_v^{od}(t)$ and $s_v^{se}(t)$ can be directly observed by the agent, while the total effect $s^{veh}(t)$ and $s^{se}(t)$ can be effectively aggregated from all agents through the backhaul network of edge servers. Finally, the distributed agents' closed-loop observations and actions can jointly optimize service migration and route planning for intelligent vehicles.

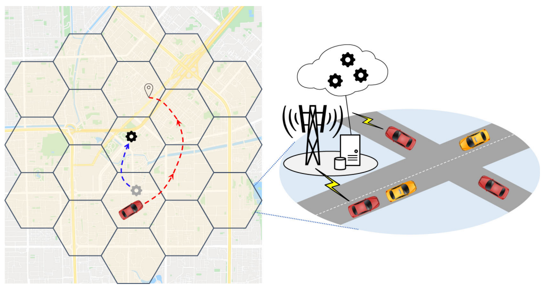

Fig. 5. The simulation scenario.

VI. Simulation Results and Discussions

A. Simulation Settings

1) Macroscopic Traffic Flow: As shown in Fig. 5, we consider a road network covered by $|\mathcal{G}| = 19$ hexagonal grids, which are determined by the coverage areas of macro cells. The traffic simulation is performed based on the Macroscopic Bureau of Public Roads (MBPR) model [38]. The MBPR model is used to estimate the average travel time given the dynamic traffic flow over a region-wide road network, which is calculated as

$$\bar{w} = \bar{w}_* + \bar{w}_* \eta V^2, \tag{22}$$

where $\bar{w}$ is the average travel time per kilometer, $\bar{w}_*$ is the free-flow travel time per kilometer, V is the hourly total traffic flow from the region to a neighboring region via different links, and η denotes the sensitivity of average travel time towards traffic congestion. In a specific road network, the real-world GPS data from vehicles can be used to perform model calibration for the MBPR function. In this paper, without loss of generality, model parameters $\bar{w}_* = 0.018$ h/km and $\bar{w}_*\eta = 9.45 \times 10^{-8}$ h^3/km/veh^2 are used throughout the following simulations. The O-D demand on the road network is generated randomly. Constrained by limited computational power, both intelligent vehicles with edge computing and traditional vehicles without edge computing are considered in the following simulations.

2) Computation Offloading: The macro cells shown in Fig. 5 are deployed to assist intelligent driving, and their communication and computing capacity is considered to be homogeneous. When a vehicle performs computation offloading in time slot t, the size of data transmitted to its SE varies from $b_v(t) = 200$ MB to $b_v(t) = 600$ MB in different scenarios, while the required CPU cycles follow a uniform distribution $u_v(t) \sim \mathcal{U}(2,4) \times 10^8$. The other parameters used throughout simulations are shown in Table II.

3) Multi-Agent Deep Q-Network Hyperparameters: The hyperparameters used for training the deep Q-network can be found in Table III.

B. Compared Methods

The proposed MADRL-based joint optimization method (MAJO) is compared with the composite of the following algorithms.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Outgoing link bandwidth, $B_e, \forall e \in \mathcal{E}$	100 Mbps
Computing capacity, $U_e, \forall e \in \mathcal{E}$	100 GHz
Resource block bandwidth, W	180 kHz
Total number of resource blocks, K	50
Maximum transmit power of vehicles, P	200 mW
Noise spectral density	-174 dBm/Hz
Path loss exponent, α	3.75
Power control factor, ϵ	0.25
Coverage radius of BS	0.5 km
Image size of SE, $\|o_v\|, \forall v \in \mathcal{V}$	500 MB
Upper bound of service delay, τ	0.35 s
Coefficient of backhaul delay, λ	0.02 s/hop
Coefficient of migration cost, μ	0.002 /MB
Discount factor, γ	0.95
Penalty for unacceptable service delay, C^-	0.2
Reward for reaching destination, C^+	1.5

TABLE III
DEEP Q-NETWORK HYPERPARAMETERS

Parameter	Value
Replay size	100
Minibatch size	16
Exploration probability, ε	0.05
Update frequency of target network, Y_1	100
Exchange frequency of two branches, Y_2	200

1) Shortest-Distance Routing (SR): Dijkstra algorithm is used to find the route with the shortest travel distance for each vehicle.

2) DQN-Based Routing (DR): A separate DQN is used to plan routes based on dynamic traffic conditions. The system state is $\{s_v^{od}(t), s^{veh}(t)\}$, the action is $a_v^{rt}(t)$, and the reward function is $r_v^{routing}(t) = -c_v^{tfc}(t) + \tilde{r}_v(t)$.

3) Lazy Migration (LM): The SE will not be migrated until its service delay exceeds the upper bound τ. Once the migration is triggered, the SE image is migrated to the local edge server of the vehicle.

4) DQN-Based Migration (DM): A separate DQN is used to migrate SEs based on the loads of base stations and edge servers. The system state is $\{s_v^{od}(t), s^{veh}(t), s_v^{se}(t), s^{se}(t)\}$, the action is $a_v^{mgt}(t)$, and the reward function is defined as $r_v^{migration}(t) = -c_v^{mgt}(t) - d_v(t)$.

In the following subsection, the performance of MAJO is compared with SRLM, SRDM, DRLM, and DRDM. It is worth noting that MAJO and DRDM have the same neural network architecture, but have different ways of using the system reward to train the neural network.

C. Simulation Results

The objective of this paper is to minimize the weighted sum of migration cost and traffic cost while satisfying the service delay requirement. However, an appropriate weight parameter ω defined in (10) should be chosen to make a trade-off between the optimization of service migration and that of route planning. To analyze how ω affects the system performance, the pure total reward $r_v^{routing} + r_v^{migration}$, the average service delay, and the average travel time with respect to ω are shown in Fig. 6, where $b_v(t) = 200$ MB and $|\mathcal{V}| = 30$. When $\omega = 0.7$, the best pure total reward and service delay are achieved, and the road traffic keeps efficient enough. Under this circumstance, the MAJO can well reshape the distribution of edge resource demands without decreasing the road traffic efficiency. Therefore, ω is set to 0.7 in the following simulations.

The total reward is a comprehensive reflection of the migration cost, traffic cost, and service delay. Therefore, the total reward is considered a key metric for the composite quality of edge-assisted intelligent driving. The total reward with respect to iteration is shown in Fig. 7. With different task sizes and different traffic volumes, MAJO can converge quickly and always provide the highest total rewards compared to other separate optimization methods. It is worth noting that MAJO performs much better than DRDM, though the network architecture of these two methods nearly keeps the same. Two DQNs of DRDM are trained with separate rewards, so its optimization of route planning does not consider the loads of edge clouds, and the optimization of service migration does not consider the trajectories of intelligent vehicles. It is demonstrated that the joint optimization of service migration and vehicle mobility can largely improve the quality of edge-assisted intelligent driving.

After comparing the performance on total reward, which is a composite metric, we show the performance on service delay, service success rate and travel cost, respectively. The service delay determines the response time of an intelligent vehicle for its varying environment. In Fig. 8(a), the average service delays under different task sizes are shown, where $|\mathcal{V}| = 30$. It can be seen that the average service delay obtained by MAJO is below 0.25 second, and the superiority of MAJO becomes more obvious on large task size. The reason is that MAJO not only optimizes service migration based on the loads of edge clouds, but also optimizes the routes of vehicles to balance the distribution of edge computing demands. In the meanwhile, other methods may plan efficient routes for vehicles, but the capacity of edge clouds along the routes is not adequate to support intelligent driving. Then, the service success rates under different values of delay upper bound are evaluated in Fig. 8(b), where $b_v(t) = 300$ MB, $|\mathcal{V}| = 40$, and the values of delay upper bound follow a uniform distribution at the corresponding interval. A task is considered to be successful if its service delay is below the upper bound. The service success rate is defined as the ratio of the number of successful tasks to the total number of offloaded tasks. From the figure, it can be seen that the MAJO achieves the highest service success rate for different values of service upper bound. Finally, the average travel time is used to represent the performance of route planning. Fig. 8(c) shows the average travel time for different traffic volumes, where $b_v(t) = 200$ MB. As SRLM and SRDM utilize the same routing algorithm, their routing performance keeps the same and is merged with label SR in the figure. For the same reason, the routing performance of DRLM and DRDM is merged with label DR. The simulation results show that MAJO can coordinate the routing of vehicles to avoid congestion.

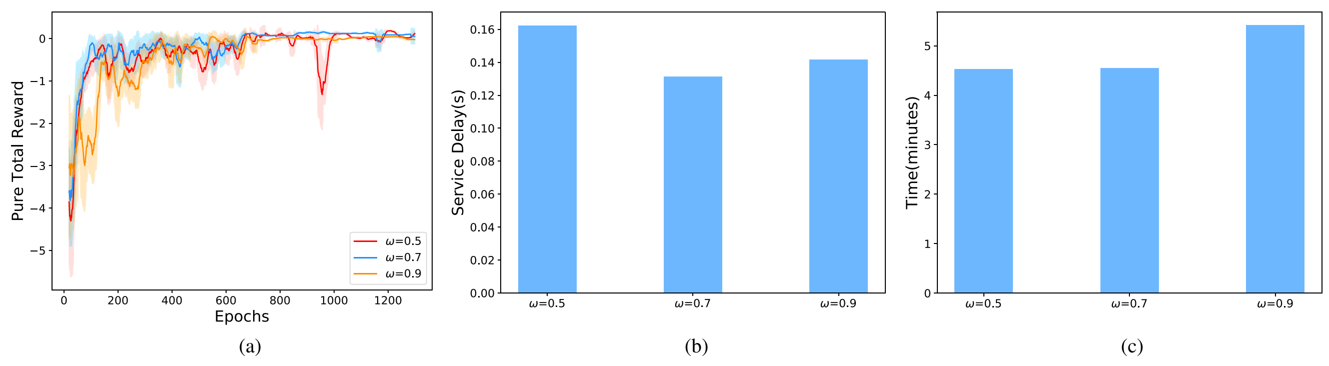

Fig. 6. Performance on the choice of weight ω for MAJO. For pure total reward, the solid lines are smoothed by window size of 20 epochs, while shaded areas show the standard deviations. (a) Pure total reward (b) Average service delay (c) Average travel time.

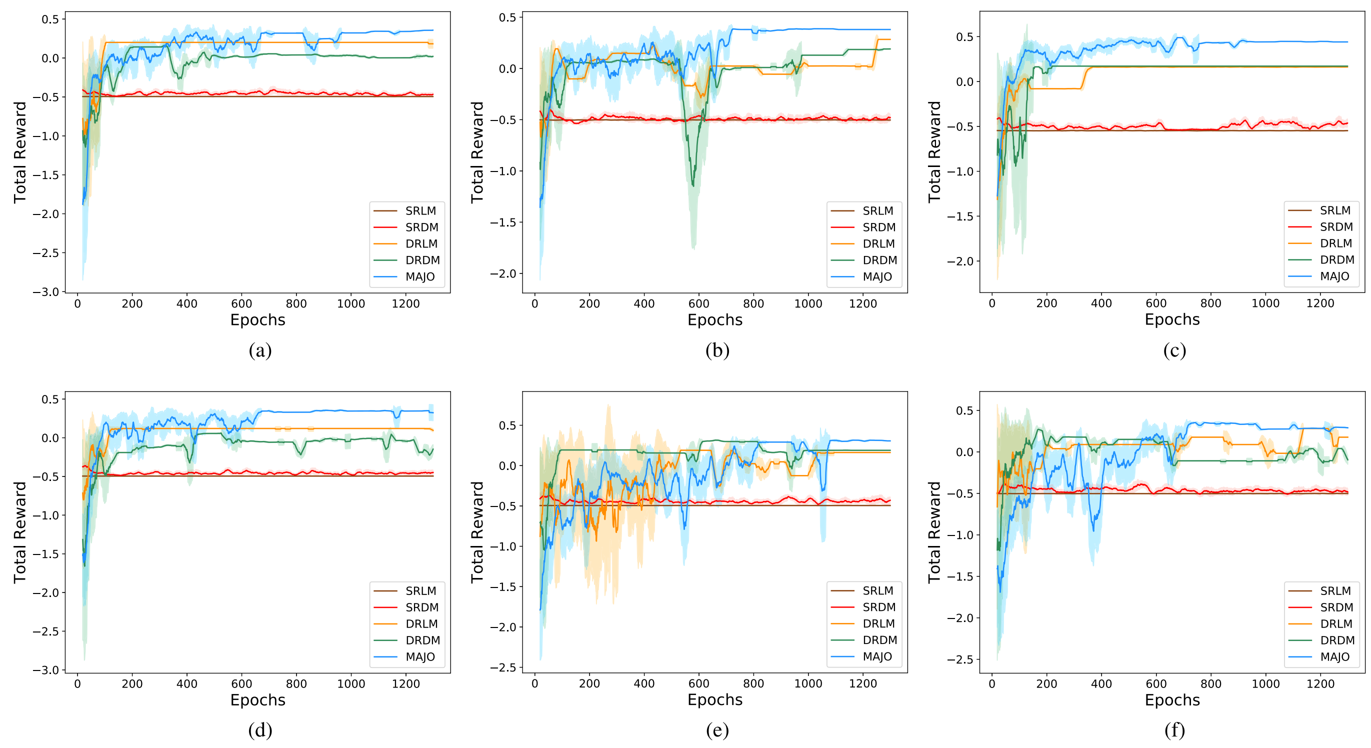

Fig. 7. Performance on total reward. The solid lines are smoothed by window size of 20 epochs, while shaded areas show the standard deviations. (a) $b_v(t) = 300$ MB, $|\mathcal{V}| = 30$. (b) $b_v(t) = 400$ MB, $|\mathcal{V}| = 30$ (c) $b_v(t) = 500$ MB, $|\mathcal{V}| = 30$ (d) $b_v(t) = 200$ MB, $|\mathcal{V}| = 30$ (e) $b_v(t) = 200$ MB, $|\mathcal{V}| = 40$ (f) $b_v(t) = 200$ MB, $|\mathcal{V}| = 50$.

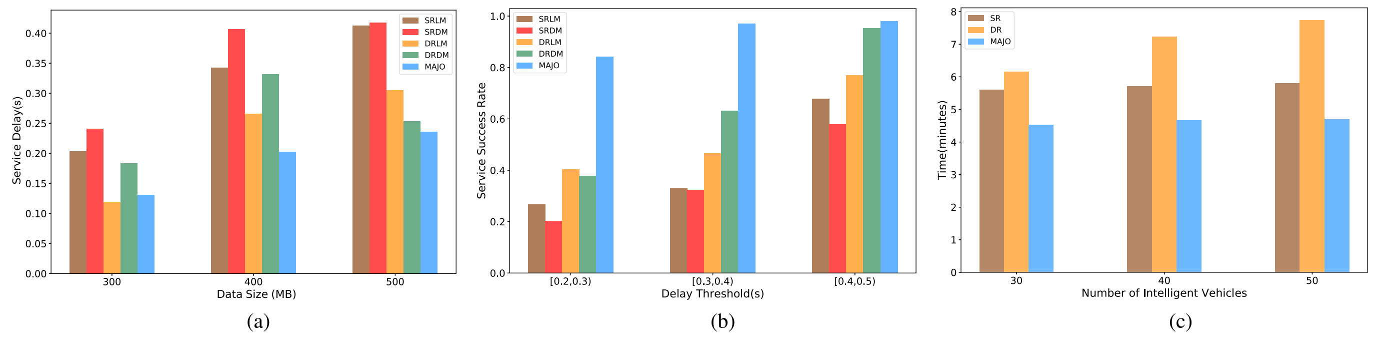

Fig. 8. Performance on service delay, service success rate and travel time. (a) Average service delay (b) Service success rate (c) Average travel time.

TABLE IV
THE GENERALIZATION OF MAJO *W.R.T.* TOTAL REWARD

# Intell. Veh.	Data Size		
	400 MB	500 MB	600 MB
40	0.4065	0.4065	0.4065
50	0.3954	0.3954	0.3854
60	0.3183	0.3103	0.2913

As the system evolves, the well-trained agents are confronted with gradually changing scenarios (e.g., the number of intelligent vehicles, and the type of computation tasks). Therefore, it needs to evaluate the MAJO's generalization capability: whether the well-trained agents can work effectively in unseen scenarios. We use the experience gathered from different scenarios to train the agent network, then apply the learned policy to both existing and unseen scenarios. Specifically, the agent network is trained with a mixed setting $|\mathcal{V}| = 30, 40, 50$ and $b_v(t) = 300, 400, 500$ MB, and the performance of learned policy on total reward is shown in Table IV. We can find that the learned policy works well not only on existing scenarios but also on unseen scenarios (i.e., the last row and the last column in the table). It reflects that MAJO has the ability to scale and generalize. However, if the scenario changes dramatically, the newly gathered experience should be used to train the agent network incrementally.

VII. CONCLUSION

In this paper, we have developed a joint service migration and mobility optimization approach for vehicular edge computing. To cope with the curse of dimensionality on the joint optimization problem, we have proposed a multi-agent deep reinforcement learning algorithm, which decomposes the state observations and actions of a monolithic centralized agent into multiple simpler agents. Besides, we have constructed a two-branch deep Q-network, and exploited an alternate stationary policy for each branch during training time to decrease training convergence time. Extensive simulations have shown that the proposed method can effectively reduce the system cost and the service delay of vehicles. In the future, we will extend the joint service migration and mobility optimization method to a multi-tier wireless communication and multi-access edge computing scenario.

REFERENCES

[1] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network toward 6G: Machine-learning approaches," *Proc. IEEE*, vol. 108, no. 2, pp. 292–307, Feb. 2020.

[2] G. Luo *et al.*, "Software defined cooperative data sharing in edge computing assisted 5G-VANET," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2019.2953163.

[3] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, Jan./Feb. 2018.

[4] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.

[5] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Mobile Comput.*, vol. 7, no. 2, pp. 369–382, Apr.–Jun. 2019.

[6] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[7] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service-migration mechanism in edge cognitive computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 30:1–30:15, 2019.

[8] J. Li, D. Fu, Q. Yuan, H. Zhang, K. Chen, S. Yang, and F. Yang, "A traffic prediction enabled double rewarded value iteration network for route planning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4170–4181, May 2019.

[9] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[10] C. Wang, C. Chou, P. Lin, and M. Guizani, "Performance evaluation of IEEE 802.15.4 nonbeacon-enabled mode for Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3150–3159, Dec. 2015.

[11] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei, "Soft-defined heterogeneous vehicular network: Architecture and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 72–80, Jul./Aug. 2016.

[12] Q. Ye, J. Li, K. Qu, W. Zhuang, X. Shen, and X. Li, "End-to-end quality of service in 5G networks: Examining the effectiveness of a network slicing framework," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 65–74, Jun. 2018.

[13] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 244–12 258, Dec. 2018.

[14] H. Zhou, N. Cheng, J. Wang, J. Chen, Q. Yu, and X. Shen, "Toward dynamic link utilization for efficient vehicular edge content distribution," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8301–8313, Sep. 2019.

[15] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[16] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[17] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[18] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[19] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, Jan.–Mar. 2020.

[20] X. Fu, R. Yu, J. Wang, Q. Qi, and J. Liao, "Performance optimization for blockchain-enabled distributed network function virtualization management and orchestration," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6670–6679, Jun. 2020.

[21] X. Shen *et al.*, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, Jan. 2020.

[22] M. Li, P. Si, and Y. Zhang, "Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9073–9086, Oct. 2018.

[23] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[24] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[25] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," 2018, *arXiv: 1812.07394*.

[26] R. Amiri, M. A. Almasi, J. G. Andrews, and H. Mehrpouyan, "Reinforcement learning for self organization and power control of two-tier heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 3933–3947, Aug. 2019.

[27] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.

[28] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," 2019, *arXiv:1901.07159*.

[29] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Traffic light control in non-stationary environments based on multi agent Q-learning," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Washington, DC, USA, 2011, pp. 1580–1585.

[30] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[31] B. Qian, H. Zhou, F. Lyu, J. Li, T. Ma, and F. Hou, "Toward collision-free and efficient coordination for automated vehicles at unsignalized intersection," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 408–10 420, Dec. 2019.

[32] 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.

[33] Z. Cao, H. Guo, J. Zhang, F. Oliehoek, and U. Fastenrath, "Maximizing the probability of arriving on time: A practical Q-learning method," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 4481–4487.

[34] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 1, pp. 217–228, Jan. 2020.

[35] 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.

[36] H. V. Hasselt, "Double Q-learning," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 2613–2621.

[37] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. Int. Conf. Mach. Learn.*, Amherst, MA, USA, 1993, pp. 330–337.

[38] W. Wong and S. Wong, "Biased standard error estimations in transport model calibration due to heteroscedasticity arising from the variability of linear data projection," *Transp. Res. B, Methodol.*, vol. 88, pp. 72–92, 2016.

Quan Yuan (Member, IEEE) received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications (BUPT), China, in 2018. He is currently a Postdoctoral Fellow at the State Key Laboratory of Networking and Switching Technology, BUPT. His current research interest includes mobile computing, crowdsensing, and vehicular networks.

Jinglin Li (Member, IEEE) received the Ph.D. degree in computer science and technology from BUPT. He is currently a Professor at the State Key Laboratory of Networking and Switching Technology, BUPT. His research interests are mainly in the areas of network intelligence, mobile Internet, Internet of Things, and Internet of Vehicles.

Haibo Zhou (Senior Member, IEEE) received the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2014. From 2014 to 2017, he was a Postdoctoral Fellow with the Broadband Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He is currently an Associate Professor with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. His research interests include resource management and protocol design in cognitive radio networks and vehicular networks.

Tao Lin is currently an M.S. Candidate at the State Key Laboratory of Networking and Switching Technology, BUPT. His current research interest includes deep reinforcement learning, Internet of Vehicles.

Guiyang Luo received the B.S. degree in electrical and communications from Beijing Jiaotong University (BJTU), Beijing, China, in 2015. He is currently a Ph.D. Candidate at the State Key Laboratory of Networking and Switching Technology, BUPT. His current research interest includes software-defined networks, medium access control and intelligent transportation system.

Xuemin Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, social networks, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen received the R.A. Fessenden Award in 2019 from IEEE, Canada, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL and the Vice President on Publications of the IEEE Communications Society.