IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 25, 2020, accepted September 14, 2020, date of publication September 18, 2020,
date of current version September 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024683

Computation Migration and Resource Allocation in Heterogeneous Vehicular Networks: A Deep Reinforcement Learning Approach

HUI WANG[1], HONGCHANG KE[2,3,4], GANG LIU[1], AND WEIJIA SUN[1]
[1]College of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China
[2]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[3]School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun 130012, China
[4]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

Corresponding author: Gang Liu (candyweiyi088@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61841602 and Grant 61806024, in part by the Jilin Province Education Department Scientific Research Planning Foundation of China under Grant JJKH20200618KJ, and in part by the Jilin Province Scientific and Technological Planning Project of China under Grant 2018C036-1.

ABSTRACT With the development of 5G technology, the requirements for data communication and computation in emerging 5G-enabled vehicular networks are becoming increasingly stringent. Computation-intensive or delay-sensitive tasks generated by vehicles need to be processed in real time. Mobile edge computing (MEC) is an appropriate solution. Wireless users or vehicles can offload computation tasks to the MEC server due to it has strong computation ability and is closer to the wireless users or vehicles. However, the communication and computation resources of the single MEC are not sufficient for executing the continuously generated computation-intensive or delay-sensitive tasks. We consider migrating computation tasks to other MEC servers to reduce the computation and communication pressure on current MEC server. In this article, we construct an MEC-based computation offloading framework for vehicular networks, which considers time-varying channel states and stochastically arriving computation tasks. To minimize the total cost of the proposed MEC framework, which consists of the delay cost, energy computation cost, and bandwidth cost, we propose a deep reinforcement learning-based computation migration and resource allocation (RLCMRA) scheme that requires no prior knowledge. The RLCMRA algorithm can obtain the optimal offloading and migration policy by adaptive learning to maximize the average cumulative reward (minimize the total cost). Extensive numerical results show that the proposed RLCMRA algorithm can adaptively learn the optimal policy and outperform four other baseline algorithms.

INDEX TERMS Vehicular networks, mobile edge computing, reinforcement learning, computation migration.

I. INTRODUCTION

With the development of wireless and vehicular networks, autonomous vehicles have gradually been introduced [1]–[3]. These networks are expected to become intelligent systems that can make predictions based on the results of self-learning and then make decisions using machine learning methods such as deep learning (DL) and reinforcement learning (RL) [4]–[6]. The entire communication system will be self-aware, self-learning, self-planning, self-growing, and self-evolving [7], because the safety and quality of service (QoS) requirements for computation and communication are more stringent, and low latency with high reliability is essential [8]. The computing power of on-board vehicular systems cannot meet the demands for computing and power consumption, and in an intensive data communication environment, it is difficult to ensure the mobility of the vehicle and the real-time reliability of the data [9]. Some studies have proposed offloading of computing data to the cloud side to reduce the pressure on the local processor [10], [11]. However, one of the more difficult problems in communicating with cloud servers is the backhaul delay of downlink transmission, which makes it impossible to meet the low-latency limit in vehicular networks [12], [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Zhang.

Mobile edge computing (MEC) has recently emerged as a means of computation and communication in wireless or vehicular networks because its servers are close to devices, and it provides supercomputing power; thus, it can meet the delay and energy consumption constraints [14], [15]. There is some research on computation offloading or resource allocation in wireless or vehicular networks. Wang *et al.* in [16] proposed an optimization method for computation offloading and physical resource block (PRB) allocation. An MEC server first estimates the computation overhead and makes offloading decisions; then, based on graph coloring, it allocates the PRBs. Guo *et al.* in [17] introduced a blockchain-based MEC model. On the basis of a proposed framework, a method of adaptive resource allocation and computation offloading was proposed to improve the throughput of the blockchain model and the QoS. However, these studies focus on a single MEC system, and assumed that the MEC servers have sufficient computing power. The number of mobile edge computing servers that are necessary in order to support the end-users needs. Apostolopoulos *et al.* in [18] formulated a two layer operation of autonomous MEC servers and QoS satisfaction of distributed mobile devices framework in distributed IoT network. At the first layer, autonomous MEC servers' activation is described as a minority game, and an active decision algorithm based on distributed learning is implemented to determine whether MEC servers are active or not. At the second layer, a non-cooperative game-based transmission power allocation for IoT devices algorithm is proposed to meet the QoS satisfaction. Ranadheera *et al.* in [19] studied the state of the art of minority game application in communication network, and formulated the computation offloading scheme in small cell networks. The proposed scheme can obtain the near-optimal offloading decision based on the reinforcement technique. Fu *et al.* in [20] proposed a resource allocation and transmission power control policy to maximize the total profit of a mobile operator based on an actor-critic algorithm with an eligibility traces algorithm. The continuous action space consists of the unit energy packet and allocated transmission power. Guo *et al.* in [21] presented a heuristic greedy offloading scheme (HGOS) to solve the energy consumption minimization with processing time constraints (EMTC) problem, and the proposed algorithm can obtain the near-optimal offload policy. Although the aforementioned works consider the multi-MEC scenarios, they does not consider the time-varying channel states.

In vehicular networks, autonomous vehicles can leave the region covered by the MEC server or cloud server. The performance of computation tasks generated by vehicles usually requires specific services or caches. These services or caches are usually stored on the cloud server and are then passed to the MEC server [22]. Thus, virtual machine migration or computation migration can be used to decrease the transmission delay and total cost. Zhang and Zheng in [23] implemented a policy for offloading and migration decisions based on Deep Q-learning (DQN). The proposed method can find the optimal policy according to the distance between the user and the MEC server to minimize the difference between the QoS and migration cost. Wang *et al.* in [24] proposed a method of migrating a service instance from one cloudlet to another according to the location of the user. The proposed method could minimize the total cost using dynamic programming (DP), and an online approximation algorithm was designed to improve the time complexity of the proposed algorithm. Wang *et al.* in [25] proposed a joint optimization method for the total cost and total delay based on microservice migration. Two algorithms were introduced: a DP-based offline algorithm and an RL-based online algorithm. The proposed algorithms consider the location of the vehicle and the relationship between the microservice and MEC server; they then provide a near-optimal migration solution. However, these works in [23]–[25] do not consider time-varying channel states or the cost of bandwidth utility; they also assume that the cost of migration is not large.

There are some existing DRL based schemes for MEC in vehicular networks. Qiao *et al.* in [26] proposed a deep deterministic policy gradient(DDPG)-based cooperative caching method for minimizing the content access cost while satisfying the constraint on content delivery latency. The proposed scheme is designed to a double time-scale Markov decision process (DTS-MDP) for content placement and updating in the large time slot, vehicle scheduling and bandwidth allocation in small time slot. For improving the content hit ratio, the priori knowledges about the content placement are set in advance. However, [26] considered more cache placement and cache delivery, and assumes that the cache placement is the priori, but did not take into account the resource management of vehicle and MBS. Liu *et al.* in [27] modelled a vehicle edge computing network architecture and proposed a DRL-based computation offloading method while considering the delay deadline and limited computation capabilities of vehicles to maximize the long-term system utility. The method can learning the optimal offloading policies and the resource allocation policy. Reference [27] considers the computation offloading and resource allocation on the MEC side and (Vehicle Edge Computing) VEC side, but when the number of (User Equipment) UEs increases, the curse of dimensionality will be trouble. Ning *et al.* in [28] formulated an intelligent offloading scheme based on DRL to minimize the offloading cost while satisfying the latency constraint of users. The proposed scheme consists of the Vehicle-to-Road(V2R) scheduling based on two-sided matching algorithm and Vehicle-to-Infrastructure(V2I) allocation based on distributed DRL method. However, although the proposed distributed DRL-based method can tackle the curse of dimensionality, resource allocation is a difficult problem for each agent, such as the bandwidth allocation of base station and the computing resources allocation of MEC server.

Different from the aforementioned works, we consider time-varying channel states and stochastic task arrival to

propose a DRL-based joint optimization scheme for computation migration and resource allocation (RLCMRA) to minimize the total cost of the MEC framework. The total cost consists of the delay cost, energy consumption cost, and bandwidth cost. The main contributions of our work are summarized as follows:

- We design an MEC framework for heterogeneous vehicular networks comprising a vehicle, multiple MEC servers, and a cloud server. The proposed MEC model considers time-varying channels, stochastic task arrival and the MEC servers randomly host the caching. To the best of our knowledge, the proposed model is the first work jointly considering time-varying channel state, the stochastic task arrival and hosting caching, and the bandwidth allocation.
- We investigate the computation offloading and migration problem and design the state space, action space, and reward function. The state space consists of the size of the arriving tasks, the channel state, the signal-to-interference-plus-noise ratio (SINR), and the state of hosting caching by MEC servers. The action space consists of the offloading decision, migration decision, and allocated bandwidth ratio. The reward is a negative total cost. For the reward function, we have considered the influence of many factors including the delay, energy consumption, and bandwidth costs.
- Furthermore, we use Double Deep Q-Learning (DDQN), which is a model-free DRL algorithm, and propose the RLCMRA algorithm to minimize the total cost of the designed MEC model. The RLCMRA algorithm can adaptively learn the optimal policy under stringent latency requirements in stochastic environments by exploiting two emerging technologies: fixed parameters in the target network and experience replay memory. It is worth noting that our proposed RLCMRA method considers more dimensions of state space including computation offloading or migration, bandwidth allocation and caching control compared with the exists works based on DRL [26]–[28].
- Extensive numerical simulations are implemented, and three baseline algorithms are designed to verify the effectiveness and superiority of the RLCMRA algorithm. The simulation results demonstrate that the RLCMRA algorithm outperforms three baseline algorithms.

The remainder of this article is organized as follows. In Section II, a design for an MEC model with a macro base station and some small base stations is presented, and the network, communication, resource allocation, and computation models are described. The problem is formulated in Section III, and the DRL-based optimization scheme is described in Section IV. Extensive numerical results are presented in Section V. Finally, the conclusions are listed in Section VI.

II. SYSTEM MODEL

A. NETWORK MODEL

We implement an MEC system for computation data migration and resource allocation in heterogeneous vehicular networks. The proposed system consists of a vehicle, multiple MEC servers, and a cloud server. As shown in Figure 1, an autonomous vehicle drives on city expressways or highways, and collects massive amounts of data, such as images or video, using sensors. Owing to constraints on its computational power and battery capacity, the vehicle may not meet deadlines for processing the data. The vehicle can offload the real-time computation data to the MEC server in the coverage region through a roadside unit (RSU) or small base station. MEC servers with high-performance processors are located near the vehicle; therefore, an MEC server can execute computation tasks offloaded from vehicles within the communication region covered by the server. Without loss of generality, we denote the set of execution caching services on the cloud side as $\mathcal{E} = \{1, 2, \cdots, E\}$. That is, the cloud server hosts all of the caching services. $\mathcal{M} = \{1, 2, \cdots, M\}$ denotes the set of MEC servers. Thus, there are M MEC servers and E execution caching services, as well as a cloud server with a macro base station. Each MEC server is connected to a small base station, which can communicate with a vehicle in real time, and only a single vehicle is within the coverage of the small base station. Although the MEC server is computationally powerful, it is affected by the channel state and the requirements of each computing task offloaded by the vehicle, such as delay limits or bandwidth restrictions. Therefore, we assume that there are E_m caching software for executing the computation tasks equipped with on the mec server m which can meet the computation requirement where $E_m \in \mathcal{E}$. For instance, the vehicle v generate the

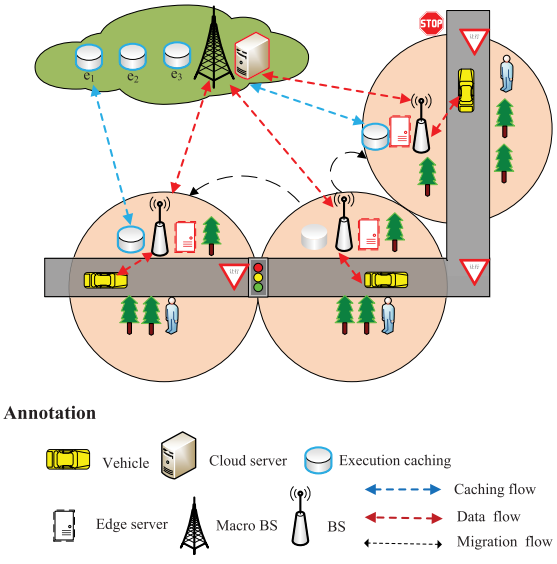


FIGURE 1. MEC framework.

computation-intensive tasks at a certain moment, it needs to be offloaded to the MEC server m for execution. Then, it is necessary to check whether the MEC server m hosts the execution cache e_m corresponding to the offloaded task. If there is, calculate the offloaded tasks directly on the current MEC server by e_m. If not, the offloaded tasks need to be migrated to other MEC server m' with the execution cache m for execution where $m' \in \mathcal{M}$ and $m' \neq m$. We assume that the autonomous vehicle moves from a source to a destination along a certain trajectory. However, m regions covered by m base stations can be seamlessly linked with each other. Table 1 defines the main notation used in this work.

TABLE 1. List of main notations.

Notation	Definition
$\mathcal{T}$	Index set of time slots
$\mathcal{E}$	Set of execution caching services
$\mathcal{M}$	Set of MEC servers
$h_{v,m}(t)$	Channel vector between vehicle v and the small base station in time slot t
$H_{v,m}(t)$	Channel gain between vehicle v and the small base station in time slot t
$P_{v,l}(t)$	Allocated power of vehicle v for local execution in time slot t
$P_{v,m}(t)$	Transmission power of vehicle v for offloading to the small base station in time slot t
$P_{v,m}^{mec}(t)$	Execution power of MEC server m for offloaded computation tasks in time slot t
$P_{v,m}^{migr}(t)$	Migration power for offloaded computation tasks in time slot t
$D_v^s(t)$	Size of arriving computation tasks generated by vehicle v in time slot t
$D_v^c(t)$	Number of CPU cycles required by vehicle v in time slot t
$D_v^{max}(t)$	Deadline for completing the tasks of vehicle v in time slot t
ζ_v	SINR of the small base station
$\tau_{v,m}$	Bandwidth allocated to vehicle v
λ_v	Arrival rate of computation tasks generated by vehicle v
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$Q(\mathcal{S},\mathcal{A})$	State-action function

B. COMMUNICATION AND RESOURCE ALLOCATION MODEL

For wireless communication, we assume that the vehicle can communicate with the small base station connected to the MEC server, and the MEC server can migrate computation tasks to another MEC server via the cloud server. Without loss of generality, the channel state between the vehicle and small base station is time-varying; that is, the channel condition changes in a certain time epoch (time slot). The base station connected to MEC server has N antennas and control the communicated vehicle with single antenna by using the linear detection algorithm zero-forcing (ZF) [29], [30].

For convenience, we split the continuous decision epoch (execution time) into time slots, which are discretized into specific time intervals. We denote the set of time slots as $\mathcal{T} = \{1, 2, \cdots, T\}$.

1) COMMUNICATION MODEL

Let $H_{v,m}(t)$ denote the channel vector between autonomous vehicle v and small base station m. We set the initial channel vector to $H_{v,m}(0) = h_{v,m}(d_0/d_{v,m})^{\sigma} I_m$, where d_0 is the distance constant, and $d_{v,m}$ is the actual distance between vehicle v and MEC server m. σ is an adjustment coefficient that depends on the communication environment [31].

In the proposed MEC model, we use the time correlation model to simulate the time-varying channel state following Gaussian Markov block fading autoregressive model [32], [33]. The channel state $H_{v,m}(t+1)$ in the next time slot $t+1$, can be written as

$$H_{v,m}(t+1) = \rho_v H_{v,m}(t) + \sqrt{1-\rho_v^2} e_v(t) \tag{1}$$

where ρ_v is the normalized correlation coefficient between time slot t and the next time slot, $t+1$. In addition, $e_v(t)$ is the error vector, and $e_v(t) \sim CN(0, I_m)$, which has a complex Gaussian distribution and is uncorrelated with $H_{v,m}(t)$.

We assume that the vehicle is allocated with an orthogonal spectrum resource block, and utilize the Rayleigh fading model and the free-space propagation path-loss model [22]. The SINR among the MEC servers in time slot t can be written as

$$\zeta_m(t) = \frac{P_{v,m}(t) \cdot H_{v,m}(t)}{\sum_{j \neq m, j \in \mathcal{M}} P_{v,j}(t) \cdot H_{v,j}(t) + \sigma_v(t)^2} \tag{2}$$

where $P_{v,m}(t)$ is the transmission power of vehicle v for offloading to MEC server m. $\sigma_v(t)$ is the variance of an additive white Gaussian noise at v [32], [33].

2) RESOURCE ALLOCATION MODEL

In the proposed MEC model, the communication bandwidth can affect the QoS for an autonomous vehicle, where the total cost increases with increasing bandwidth. We assume that the MEC system is an orthogonal frequency-division multiple access system. During time slot t, only a few sub-channels can be used to transmit computation data. We denote the total bandwidth of the wireless channel for small base station m as W_m and the bandwidth ratio allocated to vehicle v in time slot t as $\tau_{v,m}(t)$. The uplink transmission rate of vehicle v can be written as

$$r_{v,m}(t) = \tau_{v,m}(t) W_m \log_2(1 + \zeta_m(t)) \tag{3}$$

where $0 \leq \tau_{v,m}(t) \leq 1$.

C. COMPUTATION MODEL

Currently, there are some works that consider the behavior characteristics of user-end side or MEC side about computation offloading. For user-end side an optimal data offloading of multi-users in multi-MEC is proposed in [34] taking into accounting risk-seeking or loss-aversion behavior of users. an optimal data offloading policy based on non-cooperative game among the users is formulated and Pure Nash Equilibrium (PNE) is obtained. For MEC side, the limitations in computational and radio resources of MEC are considered

in [35] and an energy efficient edge server activation scheme for computation offloading based on RL and minority games is formulated. The proposed scheme can obtain the tradeoff between the enough computation capacity and the efficient energy saving. In this article, due to the single vehicle drives on the road and the computation tasks generated by the vehicle are indivisible, we do not consider the behavior characteristics of user-end side and take into account the task migration of MEC side. In our future work, we will consider user-side behavior characteristics which is significant in multi-user scenario.

When vehicle v moves on the road, it can generate computation-intensive tasks in each time slot. These tasks can be executed locally or offloaded to the MEC server. In each time slot t, we define the computation task as $D_v(t) \doteq (D_v^s(t), D_v^c(t), D_v^{max}(t))$, where $D_v^s(t)$ is the input size of the computation task, $D_v^c(t)$ is the number of CPU cycles required to complete $D_v(t)$, and $D_v^{max}(t)$ is the maximum execution delay. We assume that the arrival of computation tasks $D_v(t)$ follows a Poisson distribution with λ_v, and λ_v is independently identically distributed. Thus, it can be derived that $\mathbb{E}\left[D_v^s(t)\right] = \lambda_v$. We denote the offloading decision as $o_{v,m}(t)$.

1) LOCAL EXECUTION

Even if the computation tasks generated by vehicle v are computation-intensive, when the input size of task $D_v^s(t)$ is not large, the computing power of the vehicle v is sufficient. In this case, the computing task can be considered to be executed locally. We set the power of vehicle v allocated for local execution in time slot t as $P_{v,l}(t)$. The number of CPU cycles allocated to vehicle v can be written as

$$f_v(t) = \sqrt{\frac{P_{v,l}(t)}{\kappa \cdot (1 - o_{v,m}(t)) \cdot D_v^c(t)}} \tag{4}$$

where κ is a switching coefficient that depends on the chip architecture of vehicle v.

The execution delay of a computation task can be calculated as

$$I_{v,l}(t) = \frac{(1 - o_{v,m}(t)) \cdot D_v^c(t)}{f_v(t)} \tag{5}$$

The total energy consumption for local execution is calculated by

$$E_{v,l}(t) = P_{v,l}(t) \cdot I_{v,l}(t) \tag{6}$$

2) COMPUTATION OFFLOADING

The computation tasks (captured videos or high-definition images) from the vehicle can be offloaded to the MEC server for execution. For facial recognition, for example, vehicle v can offload a video clip or image data of a human face in each time slot. As mentioned in Section II.B, the uplink transmission rate of offloading from vehicle v to MEC server m, $r_{v,m}(t)$, can be derived. Thus, the transmission time from vehicle v to MEC server m can be obtained as

$$I_{v,m}^o(t) = \frac{o_{v,m}(t) \cdot \beta_{1,m} \cdot D_v^s(t)}{r_{v,m}(t)} \tag{7}$$

where $\beta_{1,m}$ is the transmission control coefficient for computation offloading. The total energy consumption for offloading execution is given by

$$E_{v,m}^o(t) = P_{v,m}(t) \cdot I_{v,m}(t) \tag{8}$$

where $P_{v,m}(t)$ is the transmission power for offloading the computation task to MEC server m in time slot t.

D. MIGRATION MODEL

When vehicle v generates a computation-intensive task in time slot t, owing to its limited computing power, vehicle v needs to communicate with the small base station and then offload the computation task to the MEC server for execution. Without loss of generality, we assume that the computation tasks offloaded by the vehicle require a specific execution cache service e. Therefore, after MEC server m receives the offloaded tasks, it first determines whether the buffer server has hosted the current computation tasks and needs to execute the cache service. If not, MEC server m needs to host the cache service and then execute the task, or migrate the computation task to another MEC server, m' $\left(m' \neq m, m' \in M\right)$, for execution. We denote the migration decision as $\varsigma_{v,m}(t)$ and consider the cost of data migration, which consists of the migration delay and migration energy consumption. We assume that the migration delay is related to the size of the computation task $D_v^s(t)$, and is linearly related to $D_v^s(t)$. Then, the migration delay from MEC server m to MEC server m' can be obtained as

$$I_{v,m}^{migr}(t) = \varsigma_{v,m}(t) \cdot o_{v,m}(t) \cdot \left[\beta_{2,m} \cdot D_v^s(t) + \upsilon_v\right] \tag{9}$$

where $\beta_{2,m}$ is the control coefficient for computation migration. The total energy consumption for computation migration is given by

$$E_{v,m}^{migr}(t) = P_{v,m}^{migr}(t) \cdot I_{v,m}^{migr}(t) \tag{10}$$

where $P_{v,m}^{migr}(t)$ is the migration power.

When $\varsigma_{v,m}(t) = 0$, which indicates that the computing tasks are not migrated from the MEC server m, the offloaded computation task must be executed by the MEC server. The processing delay of MEC server m can be written as

$$I_{v,m}^{mec}(t) = (1 - \varsigma_{v,m}(t)) \cdot \frac{o_{v,m}(t) \cdot D_v^c(t)}{f_m(t)} \tag{11}$$

where $f_m(t)$ is the number of CPU cycles allocated to MEC server m, which can be derived using Eq. (4). It is worth noting that the processing power of MEC server is much stronger than that of vehicle v, and the switching coefficient of MEC server is set as κ_m.

The total energy consumed by the MEC server during processing is given by

$$I_{v,m}^{mec}(t) = P_{v,m}^{mec}(t) \cdot I_{v,m}^{mec}(t) \tag{12}$$

where $P_{v,m}^{mec}(t)$ is the processing power of MEC server m.

III. PROBLEM FORMULATION

In this section, we give an optimization scheme for computation offloading and task migration selection problems. Inspired by the base station selection optimization problem of literature [36] and literature [37], we describe in detail the solution to the proposed problem. In [36] proposed the game theoretic based optimal cell selection scheme to maximize their QoS-aware performance, that is, maximize the ratio of transmission rate to power. In [37] formulated the deep learning based optimal user-cell assignment only on the users' positions to the sum of uplink transmission rate. Similar to [36] and [37], our optimization scheme is the MEC server selection for computation offloading and task migration to maximize the total weight cost. However, different from literature [36] and [37], we also paid attention to the aspect of bandwidth allocation. Considering the cost of bandwidth rent, the cost function we designed consists of the total delay, total energy consumption, and total bandwidth cost during all the time slots T. Therefore, our proposal is a joint optimization problem which is more concerned with task migration and resource allocation.

As mentioned in Section II, the total delay consists of the local execution delay, offloading delay, and migration delay, and can be written as

$$I_v(t) = I_{v,l}(t) + I^o_{v,m}(t) + I^{migr}_{v,m}(t) + I^{mec}_{v,m}(t) \tag{13}$$

The total energy consumption comprises the local execution energy consumption, offloaded energy consumption, and migration energy consumption, and can be written as

$$E_v(t) = E_{v,l}(t) + E^o_{v,m}(t) + E^{migr}_{v,m}(t) + E^{mec}_{v,m}(t) \tag{14}$$

However, the user has to pay for the rented spectrum and bandwidth. We assume that the unit price for the bandwidth usage is proportional to the bandwidth amount. Then, the total cost of bandwidth allocation can be calculated as $C_v(t) = \beta_{3,m}\tau_{v,m}(t)W_m$, where $\beta_{3,m}$ is the control coefficient for the bandwidth cost.

Finally, the total cost of the MEC model is defined as follows.

$$C(t) = \omega_1 \cdot I_v(t) + \omega_2 \cdot [E_v(t) + C_v(t)] + C^p_v(t) \tag{15}$$

where ω_1, and ω_2 are the control coefficient for adjusting the tradeoff between the cost of computational delay and the cost of energy consumption as well as the bandwidth meeting $\omega_1 + \omega_2 = 1$. $C^p_v(t)$ is the penalty for timeout during execution.

The cost minimization problem can be stated as follows:

$$\mathbf{P1} \quad \min_{(o_{v,m}(t),\varsigma_{v,m}(t),\tau_{v,m}(t))} \lim_{t\to\infty} \frac{1}{t}\sum_{t=1}^{T} C(t) \tag{16a}$$

$$\text{s.t. } 0 \le \tau_{v,m}(t) \le 1 \tag{16b}$$

$$o_{v,m}(t) \in [0, 1] \tag{16c}$$

$$\varsigma_{v,m}(t) \in [0, 1] \tag{16d}$$

$$I_{v,l}(t), I_{v,m}(t) + I^{migr}_{v,m}(t), I_{v,m}(t) + I^{mec}_{v,m}(t) \le D^{max}_v(t) \tag{16e}$$

where Eq. (16b) gives the constraint on the allocated bandwidth, which is less than 1. Eq. (16c) is the offloading decision for the computation tasks generated by vehicle v. Eq. (16d) indicates the migration decision of the computation tasks offloaded by vehicle v. Eq. (16e) shows that the local delay, offloaded migration delay, and offloaded execution delay meet the maximum delay constraint.

IV. DESIGN OF THE OPTIMAL SCHEME

The DRL-based joint optimization method for computation migration and resource allocation has been introduced in this section.

A. DRL

DRL combines the perception capacity of DL with the decision-making capacity of RL [38], [39]. The action can be directly controlled according to the input state from a certain environment [40]. DRL is an artificial intelligence method that is closer to human thought. Unlike DL, DRL can yield better results using fewer training samples [41]. The advantage of DRL is that it has more information, and is not limited by the given labels. However, DL gives the operating mechanism, and RL defines the goal of optimization [42].

DQN is a typical DRL algorithm that can solve the instability problem using function approximation in RL. It includes two advanced techniques: experience replay memory and target networks [43]. Experience replay memory enables the DRL agent to sample and train data from previously observed samples, which not only greatly reduces the amount of interaction required by the environment, but can also be used to sample a mini-batch of experience to reduce the differences in learning and updating.

As shown in Figure 2, the RL agent interacts continuously with the environment; it observes the current state s and then selects action a to obtain the greatest cumulative reward r. Here, we denote the state-value function, which represents the expected cumulative reward, as $Q(s, a)$, which can be written as

$$Q^{\pi}(s, a) = \mathbb{E}_{a\sim\pi}\left[r + \gamma Q\left(s', a'\right)\right] \tag{17}$$

where γ is the discount factor, and π is the policy. The optimal policy $\pi^\star$ is the policy that yields the maximum expected cumulative reward. It can be written as

$$\pi^* = \arg\max_{a\sim\pi} Q(s, a) \tag{18}$$

Without loss of generality, the Q-function in DQN is defined as $Q(s, a|\theta)$, where θ represents the weights of the main DQN network. Unlike table-based Q-learning algorithm, DQN uses a neural network to approximate the Q-function, and the update step of the Q-function updates the parameter θ according to the temporal difference error δ, which meets the criterion

$$\delta = r + \gamma \max Q'\left(s', a'|\theta'\right) - Q(s, a|\theta) \tag{19}$$

The neural network is updated using the stochastic gradient descent method. The updating of the Q-function is

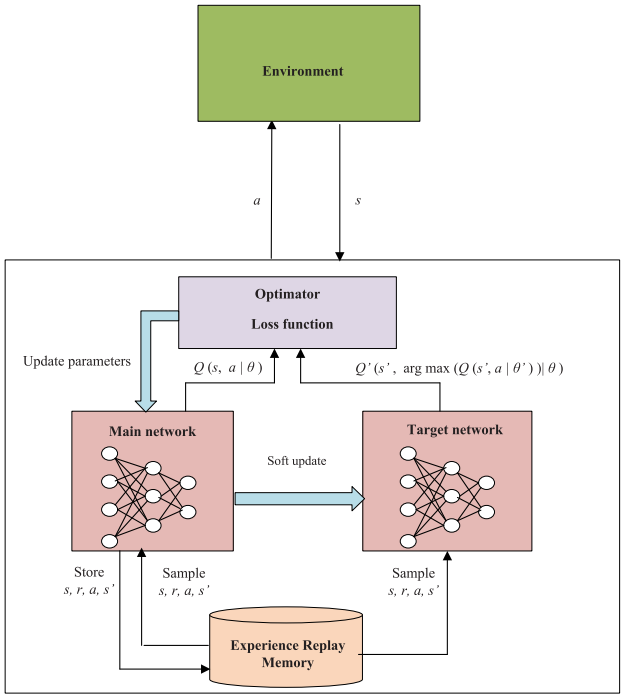


FIGURE 2. Structure of DDQN.

equivalent to the update process in traditional supervised learning, except that the DQN network is updated according to the maximum cumulative reward $\max Q(s, a)$, which can be updated as

$$L = \mathbb{E}_{(s,a,r,s')\sim M}[\left(r + \gamma \max Q\left(s', a'|\theta'\right) - Q\left(s, a|\theta\right)\right)^2] \tag{20}$$

DQN has an inherent problem with overestimation. For a selected policy in a certain state, the action that maximizes $Q(s', a'|\theta')$ is not selected in each sampling period, because the general policy is stochastic; therefore, the target value directly chooses the maximum $Q(s', a'|\theta')$ of the action will cause the target value to be greater than the true value. DDQN can effectively eliminate the overestimation problem of the DQN algorithm [44]. DDQN does not directly choose the maximum Q-value $Q(s', a'|\theta')$ of each action in the target network. Instead, it selects the action corresponding to the maximum Q-value $\max Q(s', a|\theta')$ in the main network, and then uses the selected action to calculate the target value in the target network. The target value can be updated as

$$Q'\left(s', a'|\theta'\right) \rightarrow Q'\left(s', \arg\max Q(s', a|\theta')|\theta\right) \tag{21}$$

B. OPTIMAL SCHEME BASED ON DRL

As discussed in Section III, our goal is to find the optimal computation offloading and task migration policy that minimizes the total cost, which is the weighted sum of the total delay, total energy consumption, and total bandwidth cost during all the time slots. We use DDQN to address the optimization problem in the proposed MEC model. First, we define the agent, state space, action space, and reward function.

1) THE AGENT

The agent is defined for each vehicle and is located on the cloud server. The agent has the global information of the MEC system. Thus, the agent interacts with the MEC environment to determine whether to offload or migrate computation tasks, and selects the allocated bandwidth ratio.

2) STATE SPACE

In each time slot, the vehicle generates computation-intensive tasks. As described in Section II, the arrival of computation tasks generated by the vehicle is stochastic. In addition, the channel state is time-varying. As the vehicle moves, it may be covered by different MEC servers; therefore, the communication bandwidth may change. In the proposed MEC system, the state consists of the size of the arriving tasks, the channel state, and the SINR in each time slot. The state $s \in \mathcal{S}$ can be defined as

$$\begin{aligned} s_t = \{&D_v(t), H_{v,1}(t), H_{v,2}(t), \ldots, H_{v,m}(t), \ldots, H_{v,M}(t), \\ &\zeta_1(t), \zeta_2(t), \ldots, \zeta_m(t), \ldots, \zeta_M(t)\} \end{aligned} \tag{22}$$

3) ACTION SPACE

The agent interacts with the MEC environment, observes the current state, and selects the action to obtain the reward. The agent can determine whether to offload or migrate computation tasks, and selects the allocated bandwidth ratio. Therefore, the action consists of the offloading decision, migration decision, and bandwidth ratio allocation. The action $a \in \mathcal{A}$ can be written as

$$a = \{o_{v,m}(t), \varsigma_{v,m}(t), \tau_{v,m}(t)\} \tag{23}$$

4) REWARD FUNCTION

In time slot t, the DRL agent can obtain a reward based on the current state and selected action. The object of the agent is to receive the maximum cumulative reward. We define the reward function as $R = -C(t)$. The problem of finding the optimal computation offloading and task migration policy that minimizes the total cost is transformed to the problem of maximizing the total cumulative reward.

For the optimal computation offloading and task migration policy that minimizes the total cost, we use DDQN, which is a value-based DRL algorithm, and we use the proposed DRL-based RLCMRA algorithm to maximize the average cumulative reward. The RLCMRA algorithm is presented in Algorithm 1.

The RLCMRA algorithm exploits two emerging technologies: fixed parameters and experience replay memory.

The agent observes state s in the state space $\mathcal{S}$ in each time slot and then chooses an action a in the action space $\mathcal{A}$. The immediate reward r is obtained. Furthermore, the current state s, action a, reward r, and next state s' are combined in the tuple $\langle s, a, r, s'\rangle$, which is pushed into the experience

replay memory. In the learning phase, the mini-batch samples are popped from the experience replay memory and fed into the main network and target network of the RLCMRA algorithm. Using the two neural networks, $Q(s, a|\theta)$ and $Q'(s', \arg\max Q\left(s', a \mid \theta'\right)|\theta)$ can be obtained. The stochastic gradient descent method is used to update the parameters of two networks.

Algorithm 1 Computation Migration and Resource Allocation Policy Based on RLCMRA Scheme

Initialize main RLCMRA networks and target networks with random parameters: θ, θ';
Initialize the experience replay memory to capacity M.
for episode $= [1, 2, \cdots, E_{max}]$ **do**
  Reset the MEC simulation environment including the initial state s_0 and initial reward r_0;
  Initialize the agent so that it can interact with the environment;
  for $[t = 1, 2, \cdots, T]$ **do**
    The agent observes state s, which includes the arrival of computation tasks, channel state, and SINR by interacting with the MEC environment;
    According to the policy π, select a from the action space $\mathcal{A}$ on the basis of $\epsilon - greedy$;
    if the random number $rnd > \epsilon$:
      Select action a using Eq. (21);
    else:
      Randomly select action a;
    Observe the reward r. Then the next state s' can be obtained by interaction with the simulation environment;
    If M is not full, push $<s, a, r, s'>$ into M as the training samples of the main network of RLCMRA;
    Randomly select the mini-batch m samples from M as training samples to train the main network of RLCMRA;
    In line with Eq. (20), obtain the loss L and compute the gradient $\nabla_\theta L$ of RLCMRA's main network;
    Update the parameters θ of RLCMRA's the main network using gradient descent;
    If the number of iteration steps is greater than the frequency of parameter updating, copy all the parameters θ of RLCMRA's main network to the parameters θ' of RLCMRA's target network;
  end for
end for

C. COMPLEXITY ANALYSIS OF ALGORITHM

The proposed RLCMRA algorithm is based on the neural network. Here, we consider using a fully connected neural network to simulate the optimization function. As far as the complexity of the algorithm is concerned, it is related to the dimensions of the input layer, the number of hidden layers, the number of neurons in the hidden layer, the dimensions of the output layer and the number of training samples. The complexity of RLCMRA is $O(n * k_1 + \sum_{k=2}^{K}(k * n))$, where $n * k_1$ is the number of all samples, k_1 is the dimension of the first layer for the model, and $\sum_{k=2}^{K}(k * n)$ is the total number of overall parameters. K is the total number of neural network's layers.

We can see that the dimensions of the state space, the number of hidden layers, the number of neurons, dimensions of the action space and the size of experience replay memory affect the complexity of RLCMRA. About the state space and the action space, we have already described in detail in section IV.B. The dimensions of the state space is $1 + 2 * M$, the dimensions of the action space is $1 + M * S$, S is the discretized level for the bandwidth. In addition, when our algorithm is training, because we use the mini-batch and gradient descent method for optimization, the complexity of the algorithm will be dramatically reduced. On the other hand, the factor that most affects the complexity of the algorithm is the number of training samples. Thanks to the fact that the proposed RLCMRA execute based on two emerging technologies: fixed parameters and experience replay memory. Fixed parameters ensure that the algorithm is off-line, which can meet the computational requirements of higher complexity algorithms. Experience replay memory ensures the storage of the massive traing samples, and the RLCMRA can randomly select samples in the experience replay memory during training, eliminating the correlation of the training samples. The update speed of the experience memory is significantly lower than the change position of the vehicle, and the channel state of base station which ensures the training efficiency of the proposed algorithm and thus meets the complexity requirements of the RLCMRA.

V. NUMERICAL RESULTS

To verify and evaluate the proposed RLCMRA algorithm, we establish a simulation environment based on the proposed MEC system.

A. SIMULATION SETUP

A PC having an Intel Core i9-9900K CPU with a maximum frequency of 5.0 GHz and a GeForce RTX2080Ti graphics card with 11 GB of video memory was used. We used the Pycharm integrated development environment with TensorFlow 1.10.

There were ten MEC servers with high-performance processors, and each MEC server was connected to a small base station. The MEC servers randomly host the execution caching service. The time slot is set as $t = 10$ ms. To model the communication environment, we set $d_0 = 1$ and $\rho_v = 0.95$; the radius of each MEC server was 100 m. The maximum bandwidth $W_m = 1$ MHz, $\sigma_v(t) = 10^{-9}$. In the computation model, $\kappa = 10^{-27}$, $\kappa_m = 0.45 * 10^{-28}$, and $D_v^c(t)$ is related to $D_v^s(t)$ according to Eq. (4). The number of CPU cycles required for local execution is $1.46 * 10^7$ when $D_v^s(t) = 5$. $D_v^{max}(t) = 1$(one time slot), $P_{v,l} = 2$ W, $P_{v,m} = 2$ W, $P_{v,m}^{migr} = 2$ W, and $P_{v,m}^{mec} = 2$ W. The initial value of $\beta_{1,m}$,

$\beta_{2,m}$, and $\beta_{3,m}$ are respectively set as 0.2, 0.03, 1. For the ratio of bandwidth allocation $\tau_{v,m}$, it is discretized into 10 levels, that is to say, $\tau_{v,m} \in \{0.1, 0.2, \cdots, 1.0\}$.

The main and target neural networks used in the RLCMRA algorithm have identical structures consisting of the fully connected layers. Regarding the number of layers and neurons for the the main and target networks, we will discuss later in the numerical results on the complexity analysis of RLCMRA algorithm. The maximum number of episodes is set to 1000, and the maximum number of steps in one episode is set to 200. The timeout penalty is $C_v^p = 2.5$. The maximum capacity of the experience replay is set to 200,000. The discount factor is set to $\gamma = 0.8$. The initial learning rate is set to $\alpha = 0.001$.

B. PERFORMANCE COMPARISON

The goal of this study is to identify the optimal computation offloading and task migration policy that minimizes the total cost, which is a weighted sum of the total delay, total energy consumption, and total cost of bandwidth during all the time slots. Three benchmark algorithms have been introduced for comparison with the proposed algorithm.

1) HEURISTIC GREEDY OFFLOADING SCHEME (HGOS)

According to [21], the computation tasks generated by the vehicle in each time slot are executed locally, offloaded to the MEC server, or migrated to another MEC server by the heuristic greedy algorithm, which can obtain the near-optimal offloading policy.

2) ALL OFFLOADING BUT NOT MIGRATION(AOBNM)

All the computation tasks generated by the vehicle in each time slot are offloaded to the MEC server, but are not migrated to another MEC server.

3) ALL OFFLOADING THEN MIGRATION(AOM)

All the computation tasks generated by the vehicle in each time slot are offloaded to the MEC server and migrated to another MEC server that hosts the execution caching server, regardless of the size of the computing task and the computing power of the MEC server.

4) ALL LOCAL EXECUTION(ALE)

All the computation tasks generated by the vehicle in each time slot are executed locally, regardless of the computing power of the vehicle.

C. SIMULATION RESULT

First, we performed extensive simulations of the proposed RLCMRA algorithm. We present the average cumulative rewards during all episodes. Here, each random arrival rate of the vehicle is trained 10 runs and each different channel state is trained 10 runs, that is, 10 different random seeds are set for each arrival rate of the vehicle which follows Poisson distribution. Similarly, 10 different random seeds are set for each error vector of the channel state which follows Complex Gaussian distribution. The learning results of RLCMRA are the mean of 10 best runs from all 100 runs respectively. To this end, the results obtained by training and learning are more effective and precise.

Figure 3 shows the simulation result for different learning rates α. Here, $\omega_1 = \omega_2 = 0.5$, and $\lambda_v = 5$, i.e., $\mathbb{E}[D_v^s] = 5$. For $\alpha = 0.01$, the RLCMRA algorithm cannot be guaranteed to converge, but oscillates in a certain region. For $\alpha = 0.001$ and $\alpha = 0.0001$, the RLCMRA algorithm converges. Because the arrival of computation tasks is stochastic, the curves are somewhat volatile. The performance of the RLCMRA algorithm for $\alpha = 0.001$ is better than that for $\alpha = 0.0001$. Therefore, in the simulation environment used here, we set $\alpha = 0.001$.

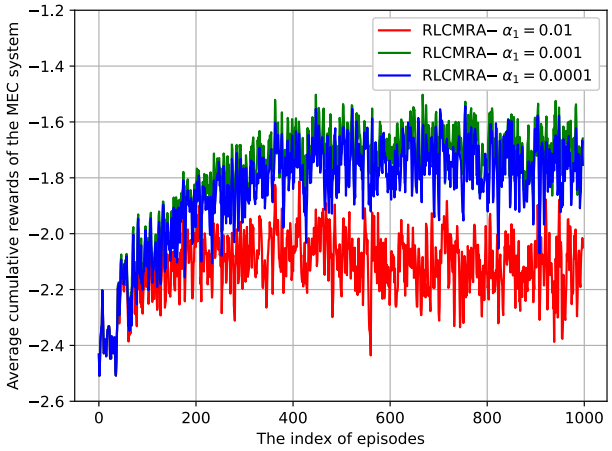


FIGURE 3. Simulation results of MEC model for various values of the learning rate α.

Figure 4 displays the results for different values of ω_1. The arrival of computation tasks is set to $\lambda_v = 5$. Regardless of the value of ω_1, the RLCMRA algorithm converges well. According to Eq. (15), since the total delay cost $I_v(t)$ is less than the energy computation and bandwidth allocation costs, the average cumulative reward increases with increasing ω_1.

In addition to the execution delay and energy consumption costs, our optimization scheme considers the cost of the rented spectrum and bandwidth as well. As shown in Figure 5, the average cumulative rewards for $\beta_{3,m}$ values of 0.5, 1.0, and 1.5 are different. According to Section IV, the unit price for the bandwidth usage is proportional to the bandwidth amount. Therefore, the average cumulative reward increases with increasing $\beta_{3,m}$. However, the convergence of the RLCMRA algorithm is not affected by the change coefficient $\beta_{3,m}$.

Figure 6 illustrates the numerical results on the theoretical complexity analysis of the proposed RLCMRA algorithm. The structure of the main network and target network for RLCMRA is fully connected network. The layer of the main network and target network is set as 2, 3, 4 respectively, and the number of neurons are 300, 200, 200, 200 respectively. We set $\omega_1 = \omega_2 = 0.5$, and $\lambda_v = 5$. As shown in Figure 6(a),

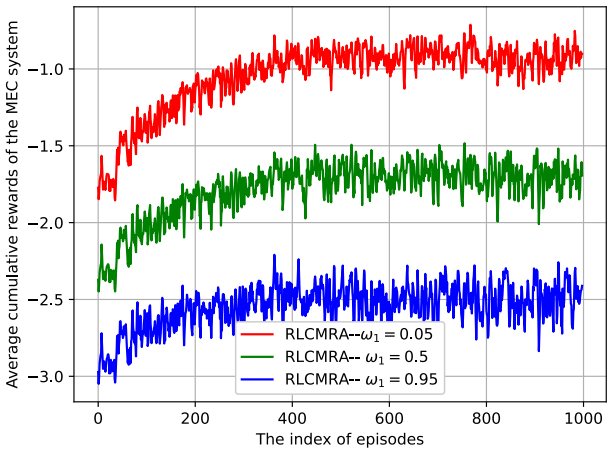


FIGURE 4. Simulation results of MEC model for various values of ω_1.

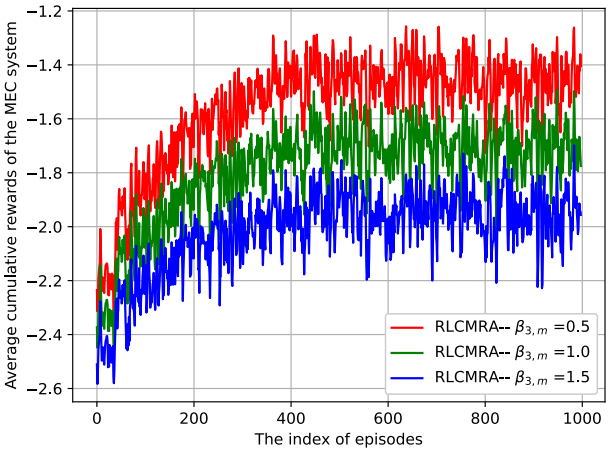
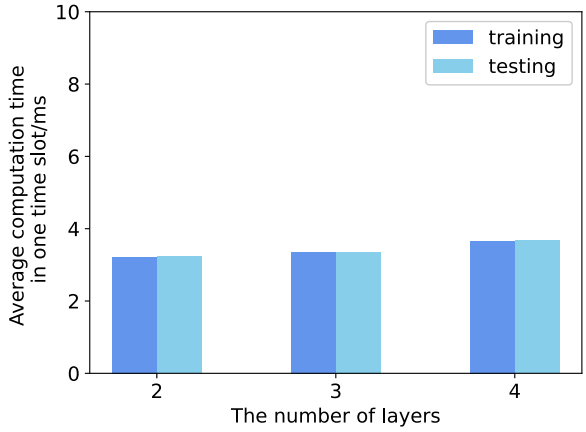


FIGURE 5. Simulation results of MEC model for various values of the coefficient $\beta_{3,m}$.

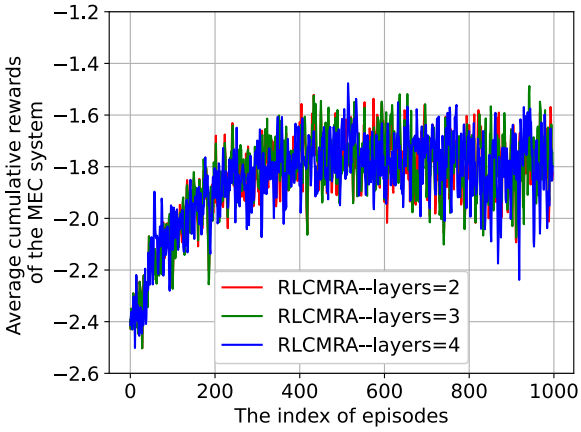
the average computation time of the RLCMRA algorithm executing during the training and testing increases as the number of layers increases, but the average computation time do not significantly change. This is because the complexity of RLCMRA is related to the number of layers for the neural network, but it is not an exponential change, but an additive change. But we can see from Figure 6(b), The convergence of RLCMRA is not greatly affected by the number of layers for neural network. No matter how many layers of the neural network of the algorithm are, the algorithm can guarantee convergence. So, In simulation environment of our proposed MEC framework, layer of the main network and target network for RLCMRA is set as 2, and the number of neurons are 300, 200 respectively for each layer.

To verify the advantages of the proposed RLCMRA algorithm over the other methods, Figures 7, 8, 9, 10, and 11 compare the results of the RLCMRA algorithm and the four benchmark algorithms.

Figure 7 illustrates the relationship between the size of the arriving computation tasks and the average cumulative reward of the proposed MEC system. Here, $\omega_1 = \omega_2 = 0.5$. As shown in Figure 7(a), as the size of the arriving computation tasks increases, the average cumulative reward decreases. Owing to the limited computing power and battery level, the ALE algorithm usually cannot complete the task processing within the deadline, and thus exhibits the worst performance. The performance of HGOS algorithm is greater than that of AOM, AOBNM and ALE due to it pays more attention to exploring different offloading modes according to the size of the arrival tasks. When $\lambda_v < 5$, the arriving computation tasks are not great, and the processing capacity of the MEC server is sufficient; thus, the cost of computation migration is greater than the cost of processing on the MEC server. Consequently, the AOBNM algorithm outperforms the AOM algorithm. When $\lambda_v > 5$, Due to

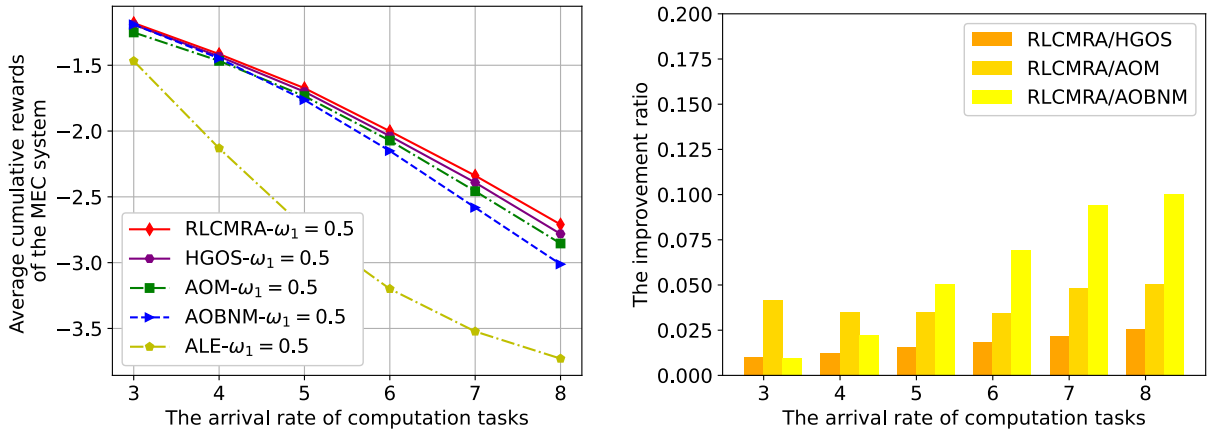


(a) Camparision result of different layers for execution time



(b) The average cumulative rewards

FIGURE 6. The complexity analysis of RLCMRA.



(a) Comparison result of different algorithms

(b) The improvement ratio

FIGURE 7. Simulation results for MEC model for arriving tasks of various sizes.

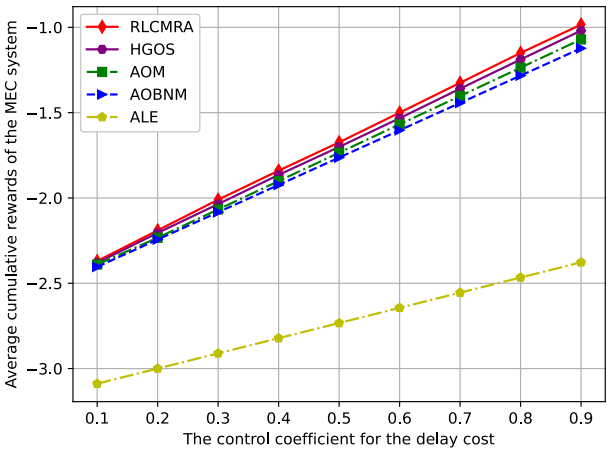


FIGURE 8. Comparison on the different algorithms for the tradeoff between delay and energy consumption, bandwidth cost.

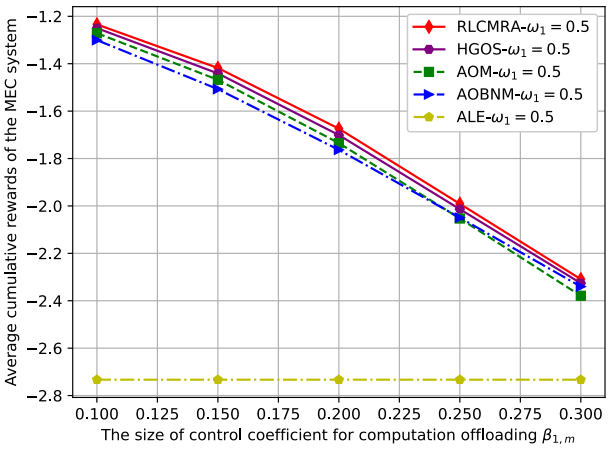


FIGURE 9. Simulation results for MEC model for various transmission control coefficients for computation offloading.

the great number of tasks arriving, the computing power of MEC server has been challenged, and the execution efficiency of migrating tasks to other MEC servers with execution caching is greater than computation on the current MEC server without execution caching. However, regardless of the value of λ_v, the RLCMRA algorithm yields the greatest average cumulative reward. As shown in Figure 7(b), as far as the improvement ratio for RLCMRA's performance is concerned, the improvement ratio of RLCMRA's is over 5% compared to the AOBNM when the arrival rate of computation tasks is greater than 5. Although the HGOS algorithm can obtain the better offloading policy, compared to the proposed RLCMRA, whose performance is less than that of RLCMRA about 1.5% in each time slot. For the computation-intensive and delay-sensitive computation tasks in vehicular networks, the improvement of performance for RLCMRA is still obvious.

Figure 8 shown the Comparison on the different five algorithms for the tradeoff between delay and energy consumption, bandwidth cost. According to Eq. (15), ω_1, and ω_2 are the control coefficient for adjusting the tradeoff between the cost of computational delay and the cost of energy consumption as well as the bandwidth, meeting $\omega_1 + \omega_2 = 1$. As shown in 8, when the delay cost coefficient $\omega_1 < 0.5$, the five algorithm is more focused on considering the impact of energy consumption and bandwidth costs. Therefore, RLCMRA, HGOS, AOM, and AOBNM are greatly affected and resulting in their average cumulative rewards becoming smaller. However, we can see from Figure 8 that no matter what value the delay cost coefficient ω_1 is, in terms of the average cumulative rewards, RLCMRA outperforms other four algorithms.

Figure 9 shows the relationship between the transmission control coefficient for computation offloading and the average cumulative rewards of the proposed MEC system.

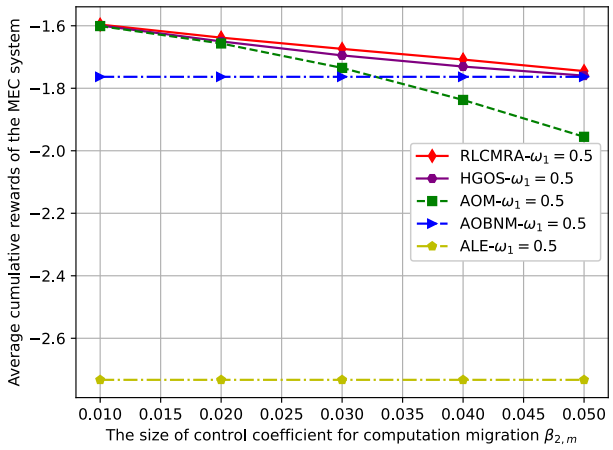


FIGURE 10. Simulation results for MEC model for various control coefficients for computation migration.

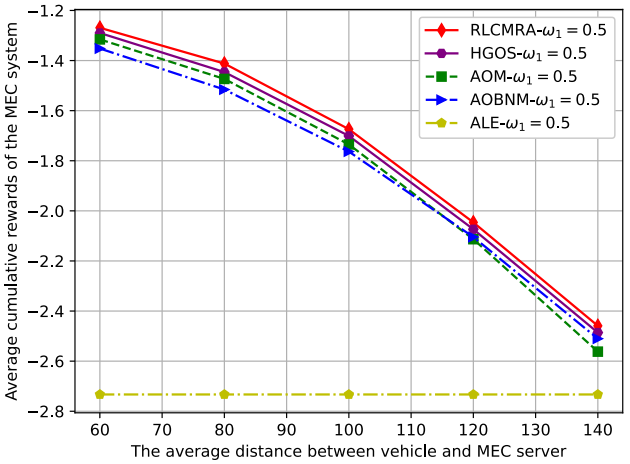


FIGURE 11. Simulation results for MEC model for various distances between the vehicle and MEC server.

According to Eq. (7), when the transmission control coefficient $\beta_{1,m}$ increases, the transmission delay of computation offloading must increase as well. The average cumulative rewards of the RLCMRA algorithm are greater than that of the AOM, AOBNM, and ALE algorithms. The ALE algorithm is not affected by changes in the transmission control coefficient $\beta_{1,m}$; therefore, its curve is very stable.

Figure 10 shows the relationship between the control coefficient for computation migration and the average cumulative rewards of the proposed MEC system. According to Eq. (9), the migration delay $I_{v,m}^{migr}$ has a linear relationship with the data migration coefficient $\beta_{2,m}$. For $\beta_{2,m} > 0.3$, because the migration delay of some offloaded computation tasks cannot meet the delay requirements, the penalty for timeout during execution affects the effect of AOM. The cost of computation migration increases rapidly which result in the average cumulative rewards of AOM decrease dramatically. Therefore, it is more cost-effective to process the computation data on the appropriate MEC server. The ALE and AOBNM algorithms are not affected by $\beta_{2,m}$; thus, their average cumulative rewards do not change. Therefore, the RLCMRA algorithm yields greater average cumulative rewards than the other four algorithms. It is worth noting that when $\beta_{2,m} > 0.5$ the cost of computation migration is too great, most offloaded tasks cannot complete the task migration under the delay constraint. So, in terms of the average cumulative rewards, the AOBNM is the optimal policy and the performance of RLCMRA is approximately equal to that of AOBNM.

As shown in Figure 11, the distance between the vehicle and MEC server affects the average cumulative rewards of the RLCMRA, AOBNM, and AOM algorithms. At larger distances between the vehicle and MEC server, the average cumulative reward is lower, and thus the total cost of the entire MEC system is higher. The RLCMRA algorithm outperforms the AOBNM, AOM, and ALE algorithms, demonstrating its greater adaptability.

VI. CONCLUSION

In this article, we modeled an MEC framework for heterogeneous vehicular networks. To minimize the total cost of the proposed MEC framework, which consists of the delay cost, energy computation cost, and bandwidth cost, we proposed the RLCMRA algorithm. For a time-varying channel state and stochastically arriving computation tasks, the RLCMRA algorithm could make offloading and migration decisions, and choose the bandwidth ratio that maximized the average cumulative reward. The RLCMRA algorithm outperformed three benchmark algorithms under various parameter configurations.

REFERENCES

[1] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1262–1276, Jun. 2019.

[2] C.-M. Huang, M.-S. Chiang, D.-T. Dao, W.-L. Su, S. Xu, and H. Zhou, "V2 V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17741–17755, Mar. 2018.

[3] F. Qiao, J. Wu, J. Li, A. K. Bashir, S. Mumtaz, and U. Tariq, "Trustworthy edge storage orchestration in intelligent transportation systems using reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 29, 2020, doi: 10.1109/TITS.2020.3003211.

[4] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[5] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, early access, Jul. 24, 2019, doi: 10.1109/TMC.2019.2928811.

[6] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[7] Y. Cui, Y. Liang, and R. Wang, "Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks," *IEEE Access*, vol. 7, pp. 21246–21253, Dec. 2019.

[8] H. Ye, G. Y. Li, and B.-H.-F. Juang, "Deep reinforcement learning based resource allocation for V2 V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[9] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.
[10] M. Goudarzi, M. Zamani, and A. T. Haghighat, "A fast hybrid multi-site computation offloading for mobile cloud computing," *J. Netw. Comput. Appl.*, vol. 80, pp. 219–231, Feb. 2017.
[11] L. Hadded, F. Ben Charrada, and S. Tata, "Efficient resource allocation for autonomic service-based applications in the cloud," in *Proc. IEEE Int. Conf. Autonomic Comput. (ICAC)*, Trento, Italy, Sep. 2018, pp. 193–198.
[12] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed Fog/Cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.
[13] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.
[14] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Jun. 2017.
[15] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.
[16] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.
[17] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.
[18] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Game-theoretic learning-based QoS satisfaction in autonomous mobile edge computing," *Global Inf. Infr. Netw. Syst.*, vol. 66, no. 6, pp. 1–5, Oct. 2018.
[19] S. Ranadheera, S. Maghsudi, and E. Hossain, "Minority games with applications to distributed decision making and control in wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 184–192, Oct. 2017.
[20] F. Fu, Z. Zhang, F. R. Yu, and Q. Yan, "An actor-critic reinforcement learning-based resource management in mobile edge computing systems," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 8, pp. 1875–1889, Aug. 2020.
[21] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.
[22] C. Xu, Y. Wang, Z. Zhou, B. Gu, V. Frascolla, and S. Mumtaz, "A low-latency and massive-connectivity vehicular fog computing framework for 5G," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.
[23] C. Zhang and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 96, pp. 111–118, Jul. 2019.
[24] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.
[25] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. S. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, early access, Dec. 5, 2019, doi: 10.1109/TMC.2019.2957804.
[26] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.
[27] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.
[28] Z. Ning, Y. Li, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, and B. Hu, "When deep reinforcement learning meets 5G-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1352–1361, Feb. 2020.
[29] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[30] C. Singhal and S. De, *Resource Allocation in Next Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017.
[31] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 179349–179363, Dec. 2019.
[32] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Energy and spectral efficiency of very large multiuser MIMO systems," *IEEE Trans. Commun.*, vol. 61, no. 4, pp. 1436–1449, Apr. 2013.
[33] H. A. Suraweera, T. A. Tsiftsis, G. K. Karagiannidis, and A. Nallanathan, "Effect of feedback delay on Amplify-and-Forward relay networks with beamforming," *IEEE Trans. Veh. Technol.*, vol. 60, no. 3, pp. 1265–1271, Mar. 2011.
[34] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.
[35] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," Nov. 2017, *arXiv:1711.09012*. [Online]. Available: http://arxiv.org/abs/1711.09012
[36] E. E. Tsiropoulou, G. K. Katsinis, A. Filios, and S. Papavassiliou, "On the problem of optimal cell selection and uplink power control in open access multi-service two-tier femtocell networks," in *Proc. Int., Conf. Adhoc. Netw. Wire*, Benidorm, Spain, Jun. 2014, pp. 114–127.
[37] A. Zappone, L. Sanguinetti, and M. Debbah, "User association and load balancing for massive MIMO through deep learning," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Oct. 2018, pp. 1262–1266.
[38] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge MA, USA: MIT Press, Mar. 1998.
[39] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, pp. 1057–1063, Feb. 2020.
[40] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. ICML*, New York City, NY, USA, Jun. 2016, pp. 2829–2838.
[41] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, New York City, NY, USA, Jun. 2016, pp. 1928–1937.
[42] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Syst. (NIPS)*, San Jose, CA, USA, Jun. 2017, pp. 6379–6390.
[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[44] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, Phoenix, AZ, USA, Feb. 2016, pp. 2094–2100.

HUI WANG received the B.S., M.S., and Ph.D. degrees from the College of Computer Science and Technology, Jilin University, in 2004, 2007, and 2010, respectively. She is currently working with the Changchun University of Technology. She has published more than 20 articles on international publications. Her research interests include wireless networks and machine learning.

HONGCHANG KE received the B.S. and M.S. degrees from the College of Computer Science and Technology, Jilin University, in 2004 and 2007, respectively. He is currently pursuing the Ph.D. degree in computer application technology with Jilin University. He is currently an Associate Professor with the Changchun Institute of Technology. He has published more than 20 articles on international publications. His research interests include wireless networks and vehicular networks, especially for offloading and caching optimization.

GANG LIU received the B.S. and M.S. degrees from the College of Computer Science and Technology, Jilin University, in 1998 and 2006, respectively. He is currently a Professor with the Changchun University of Technology. He has published more than 30 articles on international publications. His research interests include wireless communication and network security.

WEIJIA SUN received the B.S. degree from the College of Computer Science and Technology, Liaoning Normal University, in 1990, and the M.S. degree from the College of Computer Science and Technology, Jilin University, in 2004. He is currently a Professor with the Changchun University of Technology. His research interests include wireless communication and network security.