IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 16, 2019, accepted December 5, 2019, date of publication December 10, 2019,
date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958883

A Task Oriented Computation Offloading Algorithm for Intelligent Vehicle Network With Mobile Edge Computing

JUN LIU[1], SHOUBIN WANG[2], JINTAO WANG[3], CHANG LIU[1], AND YAN YAN[1]
[1]College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
[2]Beijing Institute of Remote Sensing Information, Beijing 100085, China
[3]Civil Aviation Institute, Shenyang Aerospace University, Shenyang 110136, China

Corresponding author: Jintao Wang (wangjintao@sau.edu.cn)

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61671141.

ABSTRACT With the rise of intelligent and connected vehicles (ICVs), new vehicle applications continue to emerge, while the computing capability of vehicles remains limited. Mobile edge computing (MEC) is considered to be the most effective technique for mitigating vehicle computing pressure, with computation offloading being a key technology for MEC. To solve the problem of excessive task processing delay and energy consumption due to the vehicle-limited computing power in the vehicular network, we consider the tasks and the characteristics of MEC, and divide the tasks into indivisible tasks and divisible tasks according to the size of data (that is, whether it affects functionality after segmentation). Then, two computation offloading algorithms are proposed named binary offloading and partial offloading separately. The binary offloading unloads the task to the mobile edge computing server as a whole and selects only an optimal offloading site; thus, an improved upper confidence bound algorithm is adopted. The partial offloading divides the complex tasks with large data volumes through time slots processed by different MEC servers, and uses the Q-learning algorithm to find the most effective offloading strategy. The simulation results show that the total cost of delay and energy consumption of the binary offloading algorithm is lower when processing computationally intensive tasks. When addressing divisible and complex tasks, the partial offloading algorithm improves the real-time performance of the tasks significantly and conserves the energy of the vehicle terminal.

INDEX TERMS Mobile edge computing, intelligent and connected vehicle, upper confidence bound algorithm, Markov model, Q-learning.

I. INTRODUCTION

In the vehicle network, each vehicle is equipped with an on-board unit (OBU) [1], which enables certain calculation and storage functions. However, with an increasing number of vehicle services emerging, such as augmented reality, driverless, intelligent identification, etc., which require complex calculations, more complex computing power is required [2]. The limited computing and storage resources of a vehicle-mounted terminal can no longer meet the needs of these services, and the cost of improving the in-vehicle computing capability is enormous. Therefore, we introduce the cloud computing technology into the vehicle network, as the formation of vehicle cloud computing can meet these needs to some extent. However, with the development of mobile communication, the types and number of terminal devices continue to increase. If ordinary cloud computing technologies (centralized computing) are still adopted, the demand for cloud computing will increase dramatically, which will easily cause network congestion. The high latency of cloud computing also affects the user experience [3]. Therefore, this paper combines mobile edge computing (distributed computing) with the vehicle network. Because of the distributed computing, this method can overcome the shortcomings of the centralized processing of cloud computing and alleviate the computational pressure of vehicle terminals more effectively, as well as lower delay and energy consumption [4]. The mobile edge computing system consists of four basic

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

units: the edge cloud infrastructure, the routing subsystem, the capability open subsystem, and the platform management subsystem. It offloads computing services to the edge of the network, which not only expands the computing power of vehicle equipment but also allows the transmission distance from the vehicle terminal to the computation offloading station to become shorter. Key technologies of the Mobile edge computing system include computation offloading technology, wireless data caching technology, and local offloading technology. The calculation offloading technology [5] can effectively reduce the processing delay of task execution, reduce the energy consumption of vehicle terminals, and improve the quality of the user experience. It is an important means to realize real-time processing of terminal services in the MEC system. The computation offloading includes three steps: offloading strategy, cloud processing, and result return. The offloading strategy is the fundamental basis for data volume partitioning and migration, which is the core part of computing the uninstallation. Cloud processing is performed to determine the amount of data migrated to the cloud, relying on the computing power of the cloud. The result return means that the results of the task process will be delivered to the user terminal after cloud processing. In summary, the computation offloading task is an effective means to alleviate the computational pressure of the vehicle network, but in the calculation and offloading process, it is necessary not only to detect the optimal MEC platform but also to detect the transmission quality of the wireless channel at all times. Therefore, research on the computation offloading algorithm has become a hot yet difficult task in the current automotive network application environment.

According to the characteristics of the task, these tasks are divided according to the fine granularity. Compared with the previous scheme, the algorithm proposed in this paper can more effectively perform the calculation and offloading of the vehicle task. In this paper, the MEC server is deployed at the base station to reduce the burden on the vehicle, and the vehicle is provided nearby. The theoretical foundation for future intensive IoT technology, vehicle autopilot technology, and the popularization of VR technology is then laid. The overall process of the specific research program is shown in FIGURE 1.

For tasks that need to be offloaded, considering the impact of the amount of task data on the transmission delay and post-offload calculation time and load balancing as well as whether the task itself can be divided and the integrity of the function after partitioning, we divided all the tasks into two types. One includes tasks that need to be directly offloaded without division, that is, binary offloaded. The algorithm adopted is based on the improved upper confidence algorithm of a multi-arm gaming machine. The other is a data-intensive task. We divide it and then offload it separately, which is called partial offloading in this paper. The algorithm used is based on the Q-learning algorithm, which can make full use of the computing resources of each node. A lower delay results in better load balancing. The simulation results show that the performance of offloading after task classification is better than that of unclassified offloading.

II. RELATED WORK

Mobile edge computing and computation offloading technologies are mostly applied to mobile terminal tasks. In the past two years, with the rapid development of vehicle networks and vehicle applications, the task offloading model of mobile terminals has been gradually applied to the task offloading of vehicle terminals. To minimize the mobile energy consumption and the waiting time of calculation, a general offloading model for determining the offloading decision is developed, and the number of calculations of the task is represented by w (CPU cycle) [6], [7]. f_m indicates the CPU speed of the vehicle,d indicates the input data size, and f_s indicates the CPU speed of the cloud server. Efficient computational offload processing has a significant impact on the performance of user services. The literature [8] investigates the average transmission power minimization problem while stabilizing all transmission and computation queues under tasks QoS requirement constraints, in which the influence of time-varying channel on the resource allocation strategies is considered. The literature [9] explores the basic trade-off between energy consumption and service delay when providing mobile services in a vehicle network. When the available resources in a mobile vehicle are scarce, a novel model is proposed to describe the user's willingness to provide resources to other devices. Then, the Markov decision progress (MDP) framework is used to formulate the cost minimization problem, and the dynamic reinforcement learning scheduling algorithm and the deep dynamic scheduling algorithm are proposed to solve the offloading decision problem. In [10], a reliability-oriented stochastic optimization model based on dynamic programming is proposed to perform computational offloading under the delay constraint of application execution. Zhou et al. first deduce the theoretical lower bound for calculating the expected reliability of offloading and then consider the randomness of vehicle-to-infrastructure communication; they then propose an optimal data transmission scheduling mechanism to maximize the lower bound. The design of a computing offloading system framework is studied in [11] to achieve a seamless offloading task without affecting the normal use of vehicle services. The system framework dynamically makes decisions based on network information at different times, reducing the response time of in-vehicle services. The MEC is introduced into the vehicle network in [12] and the contract problem between the vehicle user and the service provider is solved regarding the calculation and offloading by contract theory. The literature [13] refers to vehicle edge computing (VEC), which constructs an intelligent offloading system for vehicular edge computing by leveraging deep reinforcement learning to execute computing-intensive applications on resource-constrained vehicles. A vehicle-based cloud relay (VCR) scheme is proposed for mobile computing offloading in literature [14]. The goal is to effectively utilize the com-

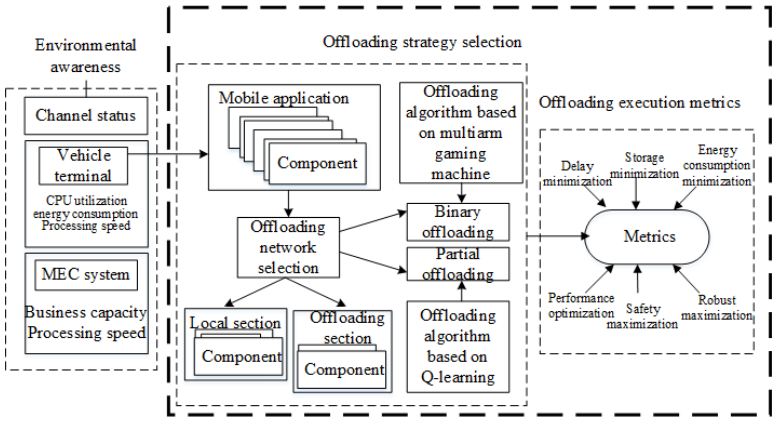


FIGURE 1. Overall process of the specific research program.

puting resources available in the surrounding smart vehicles of the mobile device in a highly dynamic network environment, where each vehicle-based cloud may be used only to help each mobile device perform a small portion of the application. The fog cloud computing offloading algorithm in the internet of vehicles (IoV) is proposed in literature [15]. With the goal of minimizing the power consumption of vehicles and computing facilities, the predictive combined transmission mode of the vehicle is designed, and the deep learning model of the computing facility is established to obtain the best offload allocation strategy. To investigates the computation offloading problem of the coexistence and synergy between fog computing and cloud computing, and achieve the allocation of computation resource and transmit power, jointly optimizing the offloading decisions was proposed in literature [16]. In order to satisfy heterogeneous requirements of communication, computation and storage in IoVs, literature [17] constructs an energy-efficient scheduling framework for MEC-enabled IoVs to minimize the energy consumption of RSUs under task latency constraints, and focus on the energy consumption control issues of MEC-enabled RSUs for the first time. The resource allocation for a multiuser Mobile-Edge Computation Offloading (MECO) system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) was proposed in literature [18]. Some researchers develop an optimal auction based on deep learning for the edge resource allocation. Specifically, they construct a multi-layer neural network architecture based on an analytical solution of the optimal auction [19]. There are also some scholars propose a distributed joint computation offloading and resource allocation optimization (JCORAO) scheme in heterogeneous networks with mobile edge computing [20], [21]. An optimization problem is formulated to provide the optimal computation offloading strategy policy, uplink sub-channel allocation, uplink transmission power allocation, and computation resource scheduling [22]. To enhance the system-wide performance, maximize the use of the social tie structure among mobile users to achieve mutually beneficial computation offloading decision making was proposed in literature [23]. To solve the non-convex optimization problem, an iterative algorithm for clock frequency control, transmission power allocation, offloading ratio and power splitting ratio was proposed in literature [24]. Some scholars study the matching problem between the MEC SPs and the UEs in a multi-MEC and multi-UE scenario. Within this scenario, MEC SPs are equipped with limited wireless and computational resources [25]. Features of Mobile edge computing are leveraged to propose a novel resource allocation approach over both communication and computation resources. The approach, implemented via successive convex approximation, is seen to yield considerable gains in mobile energy consumption as compared to conventional independent offloading across users [26]. Focusing on minimizing a cost function that depends on the latencies experienced by the users, an algorithm to minimize the latency experienced by the worst case user, under a target energy saving constraint per user was proposed in literature [27]. A low-complexity online algorithm was proposed in literature [28], namely, the Lyapunov optimization-based dynamic computation offloading algorithm, which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading. Considering a multi-mobile-users MEC system, a reformulation - linearization - technique - based Branch - and - Bound (RLTBB) method was proposed in literature [29], which can obtain the optimal result or a suboptimal result by setting the solving accuracy. The tradeoff between energy efficiency and spectral efficiency in multi-cell heterogeneous networks is investigated, objective is to maximize both energy efficiency and spectral efficiency of the network, while satisfying the minimum rate requirements of the users [30]. To transform the computation offloading problem into a convex problem and then decompose it in order to solve it in a distributed and efficient way, an alternating direction method of multipliers-based algorithm was proposed in literature [31]. To solve the optimization problem in a distributed and efficient way. An integrated fog and cloud computing (FCC) approach

was proposed in literature [32], and develop alternating direction method of multipliers based algorithms. A low-complexity suboptimal algorithm was proposed in literature [33], where the offloading decisions are obtained via semidefinite relaxation and randomization, and the resource allocation is obtained using fractional programming theory and Lagrangian dual decomposition. A multiservice resource optimization model for heterogeneous access to the internet of vehicles was proposed in literature [34]. The [35] proposed Q-learning-algorithm-based communication and computing joint resource allocation algorithm jointly dispatches various types of vehicle-linked fog resources, which enables efficient processing of intelligent transportation applications.

The above literatures optimize the delay or energy consumption of the offloading by different methods but does not consider the characteristics of the task itself and gives the corresponding solution.

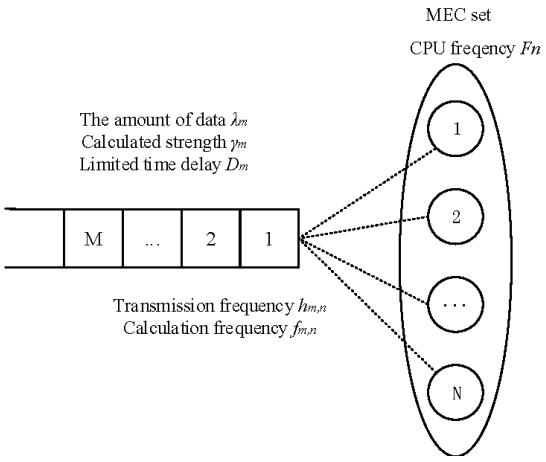


FIGURE 2. Coarse-grained task offloading model.

III. OFFLOADING STRATEGY

A. BINARY OFFLOADING

Through the computational offloading network established for the actual network scenario, the optimization strategy of the binary offloading model can be used to solve some algorithms for solving the multi-arm gaming machine problem, such as the greedy algorithm, softmax algorithm and upper confidence bound (UCB) algorithm [36]. This section describes the problem of a multi-armed gambling machine and the application of a design confidence algorithm in a binary offloading model.

Binary offloading is a computational offloading strategy where the offloading task is a computationally intensive small task that is often inseparable [37]. The purpose of this paper is to determine whether the task is processed only locally on the vehicle or migrated to the MEC system for processing and which MEC server is selected for offloading.

Assume that a vehicle terminal has N MEC server coverages, respectively labeled $N = \{1, 2, \ldots, N\}$, and that this MEC server is recorded as a collection A, each with different computing capabilities.

Each of the computational tasks m ($m \in M$) is described by three basic parameters: the input data length λ_m (bit) that needs to be offloaded, the calculation amount per bit of data γ_m, and the time limit D_m. The maximum CPU frequency at which the MEC server processes traffic is F_n ($n \in N$) and can be shared using processors while providing computing services for multiple tasks from multiple vehicles. Use computing power $f_{m,n}$ to describe the CPU frequency that the MEC server can assign to a task, which depends on several factors on the MEC server side, such as the maximum CPU frequency F_n, current total workload strength, etc. The partial offloading model is shown in FIGURE 2.

Suppose that a certain amount of data λ_m is used and that the task has a calculation amount per bit of data γ_m. The MEC server n is selected for the business calculation, and the CPU frequency allocated by the MEC server n to the task is fixed in the course of processing the task $f_{m,n}$ and, therefore, in the MEC server n. The computational delay in processing this task is shown in (1).

$$d_c(m, n) = \frac{\lambda_m \gamma_m}{f_{m,n}}. \tag{1}$$

Processing the service on the MEC server does not consume the energy of the vehicle terminal, so it is not necessary to calculate the energy consumed to process the service on the MEC server.

Input data are transmitted from the vehicle to the MEC server over the wireless uplink channel. This is expressed as the channel gain between the vehicle at the location l_m and the base station $n \in A$. Given the transmission power of the vehicle information, the maximum achievable uplink transmission rate is as shown in (2).

$$r(m, n) = W \log_2(1 + \frac{P_{tx} H_{m,n}}{\sigma^2 + I_{m,n}}). \tag{2}$$

Among them, w is the channel width, σ^2 is the channel noise, and $I_{m,n}$ is the inter-cell interference when the task offloads m to the server n. Therefore, the transmission delay and the transmission energy consumption are as shown in (3) and (4), respectively.

$$d_t(m, n) = \frac{\lambda_m}{r(m, n)}. \tag{3}$$

$$e_t(m, n) = \frac{\lambda_m P_{tx}}{r(m, n)}. \tag{4}$$

Assuming that the computing power of the vehicle terminal is strong and that the computing power of all MEC servers covering the vehicle terminal is weak, when a certain amount of data γ_m is calculated and the calculation amount of each bit of data λ_m is selected, the vehicle terminal performs local service calculation, and the data are transmitted. The delay and transmission energy consumption cost are zero, and the calculation delay and the calculation energy consumption when calculating the vehicle terminal processing service need

to be calculated as shown in (5) and (6).

$$d_c(m,n) = \frac{\lambda_m \gamma_m}{f_L}. \tag{5}$$

$$e_c(m,n) = \frac{\lambda_m \gamma_m P_L}{f_L}. \tag{6}$$

where f_L is the CPU frequency at the time of processing the service for the vehicle terminal and P_L is the power when the vehicle terminal processes the service.

In summary, the overall delay and total energy consumption of the m service processing are the objective functions, and each service delay and energy consumption are the respective constraints. The optimization function can be expressed as shown in (7)-(9).

$$\min \sum_{m=1}^{M} \upsilon_1[d_c(m,n)+d_t(m,n)]+\upsilon_2[e_c(m,n)+e_t(m,n)]. \tag{7}$$

$$s.t. \sum_{m=1}^{M} E(m,n) \leq \alpha B \tag{8}$$

$$d_c(m,n) + d_t(m,n) \leq D_m \tag{9}$$

where $\alpha \in (0,1]$ represents the proportion of the energy consumed by performing tasks in the vehicle terminal to the total energy and D_m represents the time m takes when the terminal performs the task. According to the analysis of the binary computing offloading model, we propose a binary offloading algorithm. The specific steps of the algorithm are as follows:

First, test the set of MEC servers covering the terminal vehicle in turn with the training samples, and record the first revenue of each MEC server. The training process flow chart is shown in FIGURE 3.

Second, according to the empirical conclusion of the upper confidence algorithm, $ucb(i) = \frac{1}{t}\sum_{j=1}^{t} h_i(j) + c\sqrt{\frac{\log t}{n}}$, calculate which MEC server the terminal should be offloaded to when the next task arrives, where $h_i(j)$ is the benefit of the number i server when offloading the calculation numbered j. t is the number of times the server with the number i is offloaded in total, n is the total number of iterations of training, and c is a constant.

Third, we regard the benefits of each training as a function of latency and energy consumption. Since the computing service is different, according to the requirements of the business processing for the delay and energy consumption, different weights are assigned υ_1 and υ_2, and the average income function is as shown in (10).

$$\overline{z_{m,i,t}} = \upsilon_1[\overline{d_c(m,i,t)} + d_t(m,i,t)] + \upsilon_2\overline{e(m,i,t)} \tag{10}$$

Fourth, through training, the accumulated regrets are constantly updated, and the accumulated regrets tend to be fixed. It can be considered that the performance of the current MEC server is optimal, and the cumulative regret calculation

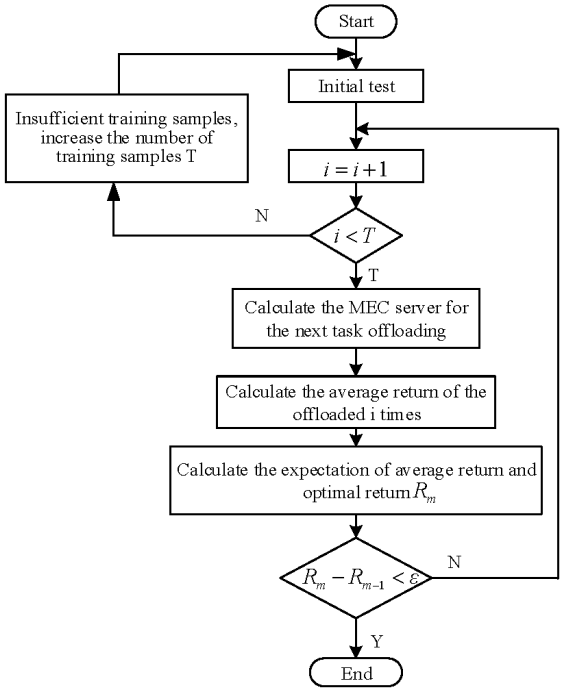


FIGURE 3. Training process flow chart.

formula is as shown in (11).

$$R_m = E[z(m,i) - z^*(m,n)] \tag{11}$$

where $z^*(m,n)$ is selected as the highest income of a certain MEC. R_m is the level of training, and the training ends when it reaches a certain value. The MEC server is obtained by selecting the MEC on the basis of the training iterations and making the accumulated regrets reach a balance.

In this algorithm, we added the exploration probability ε to explore the newly added MEC service, and the probability of $1-\varepsilon$ is used to optimize the utilization of the MEC with the best benefit so far. According to the introduction of the ε greedy algorithm, it is unreasonable for it to be a fixed value. Therefore, the value of ε is expressed by (12).

$$\varepsilon_n \overset{def}{=} \min\left\{1, \frac{cK}{d^2 n}\right\}, \quad c > 0 \&\& 0 < d < 1 \tag{12}$$

where K is the number of the MEC server in the MEC set, n is the number of tasks performed so far, and ε is adjusted according to the reward value of this MEC. Calculate the revenue of the current task to be offloaded to the MEC server, and update its average revenue and the accumulated regret value. If the cumulative regret value changes greatly, discard the uninstallation plan and restart the offloading decision for this task; if the decision is correct, proceed to the next task, which is to make a decision plan, until all the tasks are offloaded.

B. PARTIAL OFFLOADING

Since the vehicle terminal has mobility and the performance of the wireless channel of the terminal to the MEC server

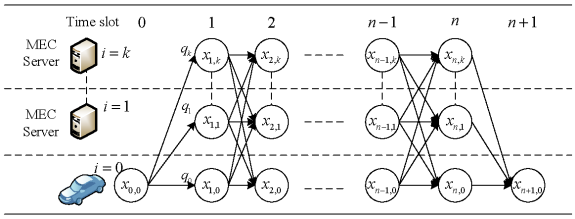


FIGURE 4. Markov state transformation model.

changes greatly, the binary offload can support only services with small calculation tasks and less computation time. For some high-intensity tasks, the transmission delay is high, which increases the requirements of the transmission quality of the wireless channel. Therefore, in response to these problems, this section divides these tasks into data by different processing time slots, abstracts the processing of the business into the state transition process of the Markov model, and uses the enhanced learning algorithm to calculate the Markov model problem for an optimal strategy to offload. The network model established according to the Markov process and the computational offloading requirement is shown in FIGURE 4. A mobile application is divided into time slots, which are considered to be composed of linear topology components. Each time slot task can be migrated to an offload site. Suppose that these n time slots offload tasks to k offloading sites. Regard the vehicle terminal as the number 0 site, and define the k *th* execution point state as q_k. The total execution status is denoted by Q, and $Q = \{q_0, q_1, q_2, \ldots, q_{k-1}, q_k\}$, where q_k is the number i offloading site and q_0 is the vehicle device itself. We consider that each offloading site has a different computing power and network bandwidth. Therefore, the computing service size of different time slots of each node varies, and the performance of the offloading strategy is measured by the total time cost and energy cost. Define $T_\upsilon = \{t_{\upsilon_0}, t_{\upsilon_1}, t_{\upsilon_2}, \ldots, t_{\upsilon_{k-1}}, t_{\upsilon_k}\}$ to represent the time cost of the component when the first time slot is executed on each offload site.

If f_i is expressed as the CPU clock speed (cycles/sec) of the offload site, and ω_υ is expressed as the total number of CPU cycles required for the instruction of component υ, then the calculation time of executing component υ on site q_i is as shown in (13).

$$t_{\upsilon_i} = \frac{\omega_\upsilon}{f_i} \tag{13}$$

Define $T_{u\upsilon} = \{t_{u_0\upsilon_0}, t_{u_0\upsilon_1}, \ldots, t_{u_0\upsilon_k}, t_{u_1\upsilon_0}, \ldots, t_{u_k\upsilon_k}\}$ to indicate the communication time from which the data are passed, where the transmission time of the components u to q_j depends on the site q_i. The data are transferred from u to v, where $t_{u_i\upsilon_j}$ represents the time it takes to transfer data from component u of site q_i to component υ on site q_i. The communication time depends on the number of data bytes transferred and the network bandwidth used between sites. Suppose $d_{u,\upsilon}$ represents the data size passed to component υ by component u and that $r_{i,j}$ represents the channel bandwidth between sites q_i and q_j. Then, the communication time is as shown in (14).

$$t_{u_i\upsilon_j} = \frac{d_{u,\upsilon}}{r_{i,j}} \forall(u, v) \in n, \quad \forall i, j \in [0, k] \tag{14}$$

Similarly, the definition of p_c is the calculated power of the vehicle terminal, and p_s and p_r are the data power of the vehicle terminal and the power of the received data, respectively. Therefore, the calculated energy and the transmitted energy of the vehicle terminal are as shown in (15) and (16), respectively.

$$e_{\upsilon_i} = t_{\upsilon_i} \times p_c, \quad (i = 0) \tag{15}$$

$$e_{u_i\upsilon_i} = \begin{cases} p_s \times t_{u_i\upsilon_i}, & \forall(u, \upsilon) \in ni = 0,\ j \in [1, k] \\ p_r \times t_{u_i\upsilon_i}, & \forall(u, \upsilon) \in ni = 0,\ j \in [1, k] \end{cases} \tag{16}$$

The goal of task offloading is to determine the offloading decision to assign n components to $k + 1$ sites so that the energy consumption on the vehicle equipment is within an acceptable range and the time cost is the smallest when the service is processed. This problem can be simulated as a delay optimization problem, which has a constraint on energy costs. The formulation minimization problem is shown in (17).

$$\begin{aligned} &\min \{\sum_{\upsilon\in V}\sum_{i=0}^{k} t_{\upsilon_i} + \sum_{(u,\upsilon)\in n}\sum_{i=0}^{k}\sum_{j=0}^{k} t_{u_i\upsilon_j}\} \\ &st. \sum_{\upsilon\in V}\sum_{i=0}^{k} e_{\upsilon_i} + \sum_{(u,\upsilon)\in n}\sum_{i=0}^{k}\sum_{j=0}^{k} e_{u_i\upsilon_j} \le \Delta energy \end{aligned} \tag{17}$$

Aiming at the optimality function of the Markov model, the time difference method based on enhanced learning is studied. The off-policy or on-policy algorithm is used to solve the utility function by means of value iteration and strategy iteration.

According to the analysis of the partial calculation offloading model, a partial offloading algorithm based on Q-learning is proposed. The specific details of the algorithm are as follows:

In the Q-learning algorithm, the agent makes the state transition decision through the instantaneous action value and the cumulative reward value. For the partially unloaded Markov model, this paper sets the minimum processing unit of MEC to 50 *kb*. When the vehicle intelligent terminal task is uninstalled, the channel state and computing power of the surrounding MEC are monitored, and (m_i, ω_{total}) is the processing state of the task, where m_i represents the number of the MEC of the current processing task and ω_{total} represents the task that has been processed until the current state. Take (m_i, ω_i) as the next action to be performed by the task, where ω_i is the amount of data to be unloaded in the next action. We set the initial reward function to 0, and the instantaneous reward function for each step of the vehicle intelligent terminal offloading task is calculated with an equation using the total delay and the reciprocal of the total energy consumption linear expression. d_{t_i} is the amount of data unloaded by the vehicle intelligent terminal at time t,

f_i is the service processing frequency of the unloaded MEC station, c_i is the channel bandwidth of the slave terminal and the MEC station to be uninstalled, p_s represents the transmission power of the vehicle intelligent terminal, and e_{t_i} represents the service processing. The energy consumed at the time is not consumed by the terminal when processing at the MEC end; thus, only the energy consumption at the terminal processing is calculated. Hence, only when $t = 0$ is added is e_{t_i} added to the reward function, and the reward function is as shown in (18).

$$r_t = 1/[k_1(\frac{d_{t_i}}{f_{t_i}} + \frac{d_{t_i}}{c_i}) + k_2(\frac{d_{t_i}}{c_i}p_s + e_{t_i})] \tag{18}$$

Set at time t, the vehicle intelligent terminal selects action a_t, the environment moves from state s_t to s_{t+1}, and the given award is r_t. Then, $r_t = r(s_t, a_t, s_{t+1})$. The probability distribution with s_{t+1}) and r_t is determined by s_t and a_t, and Q_t is the action estimation value of the state action pair at the beginning of time t. In the process of Q-learning, the optimal function can be approximated by optimizing the computational $Q(s, a)$ function; the basic update rules are shown in (19).

$$\begin{aligned} Q(s,a) &= r + \gamma \max_{a\in A} Q(s', a') \\ &= (1-\lambda_t)Q(s,a) + \lambda_t\{r + \gamma \max_{a\in A} Q(s', a')\} \end{aligned} \tag{19}$$

where λ_t is the learning rate, which represents the learning speed. The larger the value is, the faster the convergence, but if the value is too large, it may cause immature convergence and affect the optimal result. Through the above analysis, the partial offloading algorithm based on Q-learning is as shown in Table 1.

Algorithm 1 Coarse-Grained Task Offloading Algorithm Based on Q-Learning

1: Initialize the state action value, and read the task amount;
2: Repeat (for each task amount);
3: Carry out data dicing based on each MEC performance to detect the MEC performance and quantity;
4: Repeat (for each step in a training cycle);
5: Calculate the state of each action by selecting each state;
6: Select a_t from s_t according to the greedy algorithm strategy;
7: Execute a_t, and obtain the returns r_t and s_t;
8: Iteratively update the action value function;
9: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a\in A} Q(s_{t+1}, a) - Q(s_t, a_t)]$;
10: $s_t \leftarrow s_{t+1}$;
11: Until the task processing result is passed back to the terminal;
12: Until all task data processing is completed.

As can be seen from the table, each learning process of the vehicle intelligent terminal is the entire task of completing the state from the initial state to the target state. In the iterative process of each learning of the vehicle intelligent terminal, the reward value of the current state and the estimated value of a certain action are affected only by the previous state. That is, the training of the algorithm transmits data only once, so it is also called a one-step iterative algorithm. That is, the reward function of the enhanced learning has the drawback of hysteresis. This means that if the agent is unsuccessful in the development of the step n, it can only be the state of the n step state s, and the action a can calculate the current instant return $r(s_n, a_n) = -1$. The immediate returns for all states in which the previous $n-1$ step is cleared. Therefore, for any previous state S and action a, the immediate return function $r(s, a)$ does not indicate the quality of the strategy.

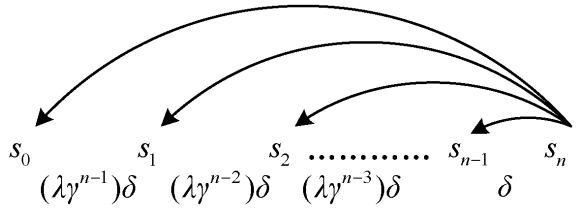


FIGURE 5. Qualification trace impact diagram.

In the Q-learning algorithm studied in this paper, the election degree in the instantaneous difference method is introduced, which is called the qualification trace, such that e_t represents the number t qualification trace, and the qualification trace can be used to indicate each action and state before the number t step. To determine the weight of this, such that the lag of the data can be solved, the mathematical expression of the qualification trace is as shown in (20).

$$e(s,a) = \begin{cases} \lambda\gamma e_{t-1}(s,a) other \\ 1 s = s_t,\ a = a_t \\ 0 t = 0 \end{cases} \tag{20}$$

The expression of the qualification trace means that when a certain state value is repeatedly utilized and a new action is selected, the qualification of the new action is set to 1. As the number of iterations increases, the weight of this state decays exponentially, as shown in (21); its specific effect is shown in FIGURE 5.

$$\delta_{t+k} = r_{t+k} + \gamma \max_{a\in A}(Q(s_{t+k+1}, a) - Q(s_{t+k}, a_{t+k})) \tag{21}$$

The introduction of the qualification trace makes the Q-learning algorithm memory-recognizable. Therefore, the accumulated state of the stored Q-state through the iteration of the corresponding Q-value is as shown in (22).

$$\begin{aligned} Q(s_t, a_t) &= Q(s_t, a_t) + \sum_{k=1}^{n-1} e_{t+k}(s_t, a_t)\delta_{t+k} \\ &= Q(s_t, a_t) + \sum_{k=1}^{n-1} (\lambda\gamma)^k \delta_{t+k} \end{aligned} \tag{22}$$

After the above analysis, this paper proposes an improvement in the partial offloading algorithm after introducing the

qualification traces for Q-learning. The specific algorithm flow is shown in Algorithm 2.

Algorithm 2 Partial Offloading Algorithm Based on Q-Learning

1: Initialize the state action value, and read the task amount;
2: Repeat (for each task amount);
3: Carry out data dicing based on each MEC performance to detect the MEC performance and quantity;
4: Repeat (for each step in a training cycle);
5: Calculate the state of an action selected for each state;
6: Select a_t from s_t according to the greedy algorithm strategy;
7: Execute a_t, and obtain the returns r_t, s_t and $\delta = r + \gamma \max_{a \in A}(Q(s', a) - Q(s, a))$;
8: Until the task processing result is passed back to the terminal;
9: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)]$;
10: $s_t \leftarrow s_{t+1}$;
11: Until the task processing result is passed back to the terminal;
12: Until all task data processing is completed.

Read the data amount from the text, and perform data dicing according to the task execution performance of the MEC monitored by the terminal. When the state value function of the current state and the predicted action value function of the next action are calculated, the action value is performed using the value of the state and the action. The update of the function, after multiple iterations, forms an optimal offload strategy.

IV. SIMULATION AND ANALYSIS

A. BINARY OFFLOADING SIMULATION AND ANALYSIS

The binary offloading task represents some inseparable tasks or tasks that can be processed only at the terminal. For the processing of inseparable tasks, an upper confidence algorithm that addresses the problem of multi-arm gaming machines is proposed. Then, this algorithm is slightly improved to make it more suitable for network scenarios. The text is designed for the parameters of the simulation scenario, considering that there are five detectable base stations around the terminal, with each base station having one MEC server. The calculation frequency and the maximum calculation amount of each MEC server are set, and each task can be involve one of the MEC servers or the terminal itself to process the task at a certain time. The size of the uploaded task volume is read from the terminal and saved in a text document for queuing because the simulation is for binary operation offloading; thus, the task volume is relatively small. This paper is specified below 50 *kb*, and the specific simulation network scene parameters are shown in Table 1.

In addition to the above network scene parameters, this paper also sets the calculation frequency and the maximum calculation amount of each MEC server and tests the performance of the vehicle terminal. The parameters are shown in Table 2.

TABLE 1. Network parameters.

parameter	value
Number of base stations	5
Number of MECs per base station	1
Transmit power	$50 - 100mw$
Upload task size	$< 50kb$
Terminal transmission rate	$1000bps$
Terminal receiving rate	$1000bps$

TABLE 2. Network parameters.

Type	Calculation frequency	Maximum calculation
M1	$2MHz$	$20000kb$
M2	$3MHz$	$20000kb$
M3	$5MHz$	$20000kb$
M4	$1MHz$	$20000kb$
M5	$2.5MHz$	$20000kb$
Vehicle terminal	$0.05MHz$	$--$

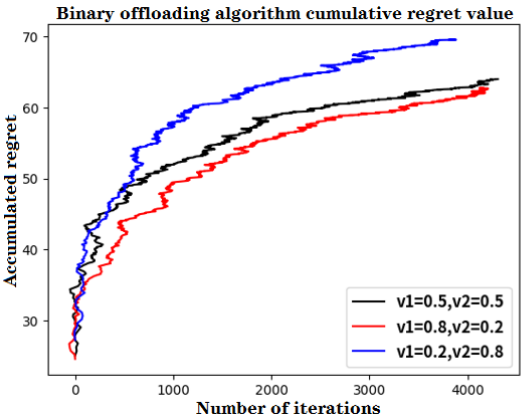


FIGURE 6. Cumulative regret value.

The cumulative regret value of the algorithm is simulated by taking the calculation task volume of 30 *kb* as an example. The relationship between the cumulative regret and the number of iterations is shown in FIGURE 6. It can be seen from the beginning that the terminal recognizes the performance of the external MEC system. When the amount of knowledge is relatively small, the regret value increases rapidly. As the number of iterations increases, the vehicle terminal continuously learns and explores the surrounding environment, and the search strategy tends to optimize the strategy. Accordingly, the unfortunate increase is reduced. Among them, most of the binary offloading tasks are services with high real-time requirements. This experiment verifies the cumulative regret value of the binary computing offloading caused by different values of the benefit function of the delay function and the energy of the income function, where υ_1 represents the coefficient of total delay for business processing and υ_2 represents the coefficient of energy consumption for business processing.

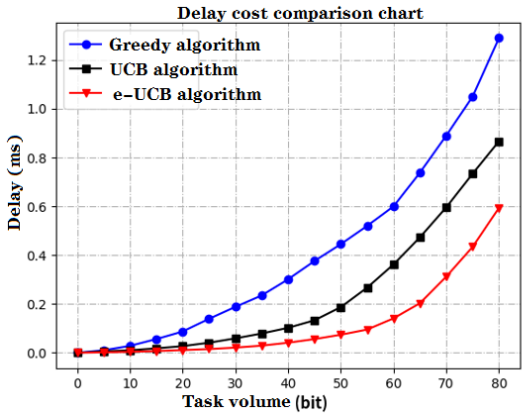


FIGURE 7. Comparison of delay costs for different tasks.

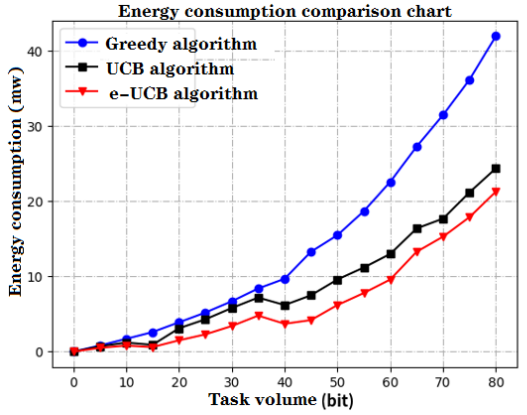


FIGURE 8. Comparison of energy costs for different tasks.

In the comparative simulation experiment of the binary offloading algorithm, the comparison of the delay between the 0 *bit* and 80 *bit* tasks and the energy consumption is taken. This paper compares the greedy offloading algorithm proposed by [38] for the binary offloading algorithm with the classical computing offloading algorithm based on the upper confidence bound algorithm and the $\varepsilon - UCB$ algorithm proposed in this paper. FIGURES 7 and 8 show that the upper confidence algorithm is adopted. When the MEC server is trained first to obtain a certain prior knowledge by the upper confidence algorithm, the delay and energy consumption of the task processing are low, and the $\varepsilon - UCB$ algorithm proposed in this paper has lower cost in terms of delay and energy consumption; thus, this offloading algorithm works better. In addition, binary offloading is mainly proposed for delay-sensitive intensive small tasks, while small tasks are based on 50*kb*. As shown in the figure, the effect of the offloading algorithm works best at 50 *kb*. The effect is weakened.

Since the energy consumption of the task of performing the calculation and offloading on the MEC server mainly comes from the transmission consumption of the request data, when the task selection is performed at the terminal, the energy consumption of the terminal is derived from the energy consumption of the task execution; thus, the energy consumption is slightly higher; see the two spikes in the figure for proof.

In summary, the performance of the binary offloading algorithm is mainly analyzed from the accumulated regret value, the delay of business processing and the cost of energy consumption. These analyses are mainly based on the selection of the reward function. The simulation analysis mainly compares the accumulated regret value and the cost of energy consumption and delay when performing binary offloading. Although the other factors are not considered, the upper confidence bound algorithm can, through training to establish the computing power function of each MEC server, reduce the delay of the task execution and the cost of energy consumption. To better adapt to the mobility problem of a device, the proposed $\varepsilon - UCB$ algorithm is adopted in the upper confidence algorithm. The exploration probability is increased to explore the computing power of the unknown MEC server, and the influence of the movement of the vehicle equipment on the task offloading algorithm is solved.

B. PARTIAL OFFLOADING SIMULATION AND ANALYSIS

Part of the task represents a large amount of data and can be divided. For this kind of task, a partial offloading algorithm based on Q-learning is proposed. By analyzing the instantaneous difference method, the qualification trace is used to improve the hysteresis of Q-learning, and the algorithm is simulated and analyzed. Since this section is aimed at partial analysis, the task set in this paper is more than 50 *kb*. Therefore, the specific simulation network scene parameters, except the task size, are as described in the above section of this chapter. The table shows and sets the calculation frequency and the maximum calculation amount of each MEC server, and the performance of the vehicle terminal is tested. The parameters are shown in Tables 1 and 2.

We compare the convergence speed and strategy of the Q-learning algorithm and the improved Q-learning algorithm. The results are shown in FIGURE 9. The state point Q-value of the training in the Q-learning is continuously iterated and trained to reach the optimal state. The already converged state can affect the correct choice of the latter. After the qualification trace is added, the memory matrix is generated, the relationship between states is established, the efficiency of the traditional Q-learning is improved, and the update speed of the Q-value is accelerated. Therefore, after fewer iterations, the optimal strategy can be achieved, and the convergence speeds up.

The idea of calculating the offloading is to process the vehicle terminal task upload until the end of the task, to complete the processing of a task, and to perform a state transition in each time slot. In this paper, the task quantity is changed from 100 *bits* to 100 *steps*, and eight tests are performed. The two algorithms are tested for optimal policy state transition.

In this experiment, data of size 100800 *bits* are selected, and the classical offloading algorithm based on the Lyapunov function, improved Q-learning algorithm and traditional

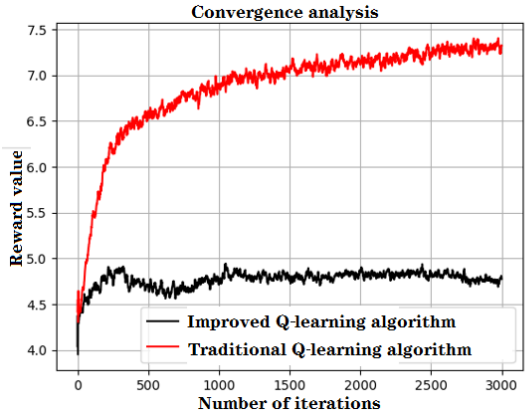


FIGURE 9. Convergence comparison.

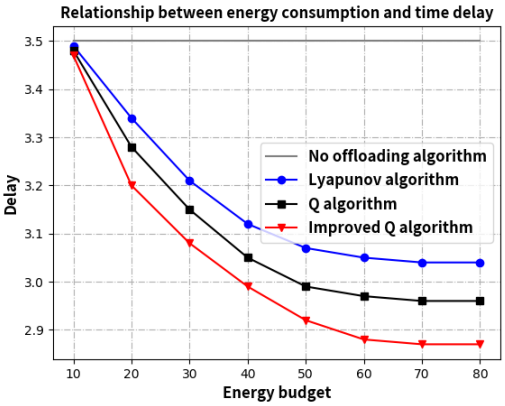


FIGURE 12. Relationship between energy consumption and time delay.

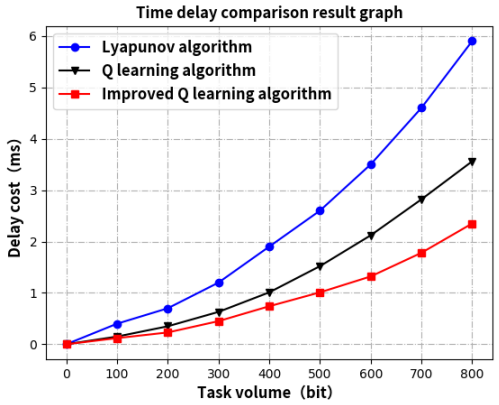


FIGURE 10. Comparison of delay costs for different tasks.

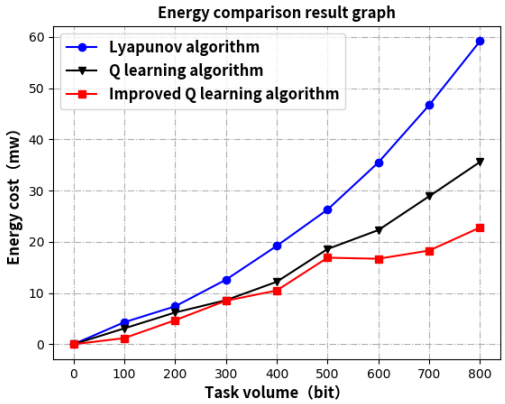


FIGURE 11. Comparison of energy costs for different tasks.

Q-learning offloading algorithm proposed in [39] are compared and analyzed. As shown in FIGURES 10 and 11, the time delay and energy cost of the offloading by the MEC are lower than those of the Lyapunov-function-based offloading algorithm, and the delay and energy consumption of the improved Q-learning offloading algorithm are less than those of the conventional Q-learning algorithm.

Because the energy of the vehicle terminal is limited, the simulation experiment is also carried out on the relationship between the energy consumption and the delay consumed by the task execution. The algorithm satisfies the adjustment of the optimization strategy when the energy level is fixed. Now the task volume is fixed to 500 *bits*. Under the premise of limiting the energy budget, the experimental comparison of the algorithm is shown in FIGURE 12. When there is no offloading algorithm, the energy consumption is a fixed value. When considering the energy consumption, the Q-learning offloading algorithm and the improved Q-learning offloading algorithm experience a gradual decrease as the energy consumption increases until the energy budget reaches the energy consumed only by the terminal, and the link between latency and energy consumption is subsequently weakened. It is true that under the premise of a constant energy budget for the same task, the improved partial offloading algorithm is better than the traditional Q-learning offloading algorithm

After analyzing the index delay and energy consumption of the offloading calculation, the proportion of the task quantity offloading can also directly reflect the offloading strength of the calculation offloading. In this paper, the offloading ratio is analyzed by the histogram, and the different tasks are counted according to the offloading result. Under the two algorithms, the size of each task is downloaded. The result is shown in FIGURE 13. Overall, the improved partial offloading algorithm is more powerful, and the traditional Q-learning offloading is performed for 100 *bit* and 700 *bit* tasks. The algorithm chooses not to perform the offloading, only performing the service in the terminal, and the improved partial offloading algorithm performs a small part of the offloading calculation. Combined with the comparison of the graph delay results, the delay of the improved partial offloading algorithm is lower than that of the traditional Q-learning offloading algorithm, while at 300 *bits*, the traditional Q-learning offloading algorithm has a slightly higher offloading ratio than that of the improved partial offloading algorithm. However, when compared with the graph delay,

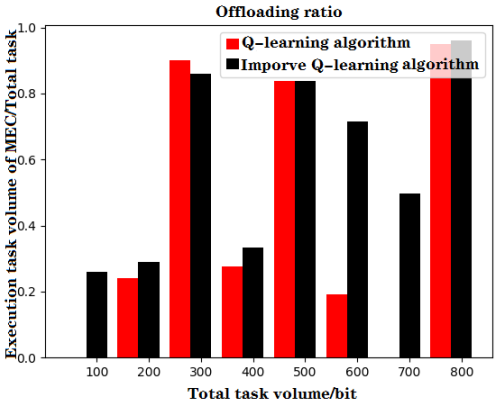


FIGURE 13. Offloading ratio.

the partial offloading algorithm proposed in this paper has a better performance.

In the simulation experiment of the partial offloading algorithm, the convergence speeds of the two algorithms are analyzed separately, and in different offloading algorithms, the strategy selection of the offloading MEC system according to the values of different Q-learning functions varies; thus, the final optimal offloading strategy is very different. In this respect, for the offloading strategy presented for the two algorithms, the delays in taking the respective offloading strategies, the delay analysis under different energy constraints, and the offloading ratio of the two algorithms, a comparative analysis was conducted.

V. CONCLUSION

This paper is devoted to designing a more efficient calculation and offloading algorithm for vehicle terminals, making the processing of services more efficient and improving the quality of service for users. Through the analysis of the principle of computing offloading, the offloading task is further divided according to the size of the traffic volume, and the algorithm is divided into two types: binary offloading and partial offloading. For the binary offloading algorithm, the total cost of delay and energy consumption of the service processing by the upper confidence algorithm is lower than that of the traditional binary offloading algorithm, and the addition of the exploration probability further compensates for the defect of the traditional binary offloading for the mobile service processing. For the partial offloading algorithm, we simulated the proposed Q-learning partial offloading algorithm and the traditional Q-learning algorithm. It can be seen from the comparison results that in the tasks process under the same task amount, the processing delay and the energy consumption of the vehicle terminal of the Q-learning algorithm that joins the qualification trace are lower than those of the traditional Q-learning algorithm; meanwhile, the convergence speed is fast. It is thus proved that the existence of the MEC system is necessary in response to the explosive growth in traffic, and effective computation offload can significantly improve the processing speed of a task and the user experience. At the same time, it also provides a powerful proof for the role of the MEC system in producing low-latency and high-efficiency services for users.

REFERENCES

[1] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, "TrajCompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[2] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, and Y. Li, "When deep reinforcement learning meets 5G vehicular networks: A distributed offloading framework for traffic big data," *IEEE Trans. Ind. Informat.*, to be published.

[3] Y. Peng, A. Tan, J. Wu, and Y. Bi, "Hierarchical edge computing: A novel multi-source multi-dimensional data anomaly detection scheme for industrial Internet of Things," *IEEE Access*, vol. 7, pp. 111257–111270, Jul. 2019.

[4] X. Wang, Z. Ning, X. Hu, L. Wang, L. Guo, and B. Hu, "Future communications and energy management in Internet of vehicles: Toward intelligent energy-harvesting," *IEEE Wireless Commun.*, to be published.

[5] J. Lee, Y. Yi, S. Chong, and Y. Jin, "Economics of WiFi offloading: Trading delay for cellular capacity," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1540–1554, Mar. 2014.

[6] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, 2013.

[7] C. Chen, S. Jiao, S. Zhang, W. Liu, L. Feng, and Y. Wang, "TripImputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 10, pp. 3292–3304, Oct. 2018.

[8] S. Li, G. Zhu, and S. Lin, "Joint radio and computation resource allocation with predictable channel in vehicular edge computing," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Maui, HI, USA, 2018, pp. 3736–3741.

[9] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, Feb. 2019.

[10] J. Zhou, D. Tian, Y. Wang, Z. Sheng, X. Duan, and V. Leung, "Reliability-oriented optimization of computation offloading for cooperative vehicle-infrastructure systems," *IEEE Signal Process. Lett.*, vol. 26, no. 1, pp. 104–108, Jan. 2019.

[11] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biol. Inspired Comput.*, Piscataway, NJ, USA: IEEE Press, Dec. 2009, pp. 210–214.

[12] Y. Umenai, F. Uwano, Y. Tajima, M. Nakata, H. Sato, and K. Takadama, "A modified cuckoo search algorithm for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Piscataway, NJ, USA: IEEE Press, Jul. 2016, pp. 1757–1764.

[13] Z. Ning, P. Dong, X. Wang, J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 1, no. 1, pp. 1–25, Jan. 2019.

[14] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11181–11191, Nov. 2018.

[15] X. Wang, X. Wei, and L. Wang, "A deep learning based energy-efficient computational offloading method in Internet of vehicles," *China Commun.*, vol. 16, no. 3, pp. 81–91, Mar. 2019.

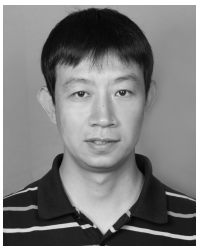
[16] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Jun. 2019.

[17] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled Internet of vehicles: Toward energy-efficient scheduling," *IEEE Netw.*, vol. 33, no. 5, pp. 198–205, Sep./Oct. 2019.

[18] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[19] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[20] Z. Ning, Y. Feng, M. Collotta, X. Kong, X. Wang, and L. Guo, "Deep learning in edge of vehicles: Exploring trirelationship for data transmission," *IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5737–5746, Oct. 2019.
[21] Y. Peng, Y. Yu, L. Guo, D. Jiang, and Q. Gai, "An efficient joint channel assignment and QoS routing protocol for IEEE 802.11 multi-radio multi-channel wireless mesh networks," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 843–857, 2013.
[22] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.
[23] L. Tang, X. Chen, and S. He, "When social network meets mobile cloud: A social group utility approach for optimizing computation offloading in cloudlet," *IEEE Access*, vol. 4, pp. 5868–5879, Sep. 2016.
[24] 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.
[25] H. Zhang, F. Guo, H. Ji, and C. Zhu, "Combinational auction-based service provider selection in mobile edge computing networks," *IEEE Access*, vol. 5, pp. 13455–13464, Jul. 2017.
[26] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.
[27] M. Molina, O. Mulioz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. IEEE 25th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Washington, DC, USA, Sep. 2014, pp. 1093–1098.
[28] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[29] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.
[30] C. C. Coskun and E. Ayanoglu, "Energy- and spectral-efficient resource allocation algorithm for heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 590–603, Jan. 2018.
[31] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
[32] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018.
[33] 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.
[34] Y. Peng, X. Wang, L. Guo, Y. Wang, and Q. Deng, "An efficient network coding-based fault-tolerant mechanism in WBAN for smart healthcare monitoring systems," *Appl. Sci.*, vol. 7, no. 8, pp. 817–835, 2017.
[35] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.
[36] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. Sha, "Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1478–1496, Jun. 2017.
[37] Y. Peng, Q. Song, Y. Yu, and F. Wang, "Fault-tolerant routing mechanism based on network coding in wireless mesh networks," *J. Netw. Comput. Appl.*, vol. 37, pp. 259–272, Jan. 2014.
[38] M. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE 16th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Stockholm, Sweden, Jun./Jul. 2015, pp. 186–190.
[39] 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.



JUN LIU was born in 1969. He received the Ph.D. degree in communications and information systems from the Northeastern University. He is currently an Associate Professor with Northeastern University. His research interests include space information networks, self-organizing networks, and information security.

SHOUBIN WANG is currently a Researcher with the Beijing Institute of Remote Sensing Information. His research interests include wireless networks, computer communication networks, and the IoT.

JINTAO WANG received the Ph.D. degree from the University of Chinese Academy of Sciences, in 2018, and the M.Sc. degree from Northeastern University, China, in 2012. He currently works with Shenyang Aerospace University. His research interests include space information networks, industrial control networks, and wireless mesh networks.

CHANG LIU is currently pursuing the master's degree in electronics and communication engineering with Northeastern University, China. Her current research interests include computation offloading and space-earth integration networks.

YAN YAN received the degree in electronics and communication engineering from Northeastern University, China. Her research direction includes 5G mobile communication and edge computing.