User Scheduling and Task Offloading in Multi-tier Computing 6G Vehicular Network

Haijun Zhang, Senior Member, IEEE, Lizhe Feng, Xiangnan Liu, Student Member, IEEE, Keping Long, Senior Member, IEEE, George K. Karagiannidis, Fellow, IEEE

Abstract—Many real-time application scenarios are developed in 6G communications. Driven by the low-latency data processing requirements, multi-tier computing has become an important technology to improve user experience and reduce network overhead. In this paper, we consider a multi-tier computation offloading network structure for 6G applications, in which the cloud computing center and the nearby vehicle edge server (VES) are able to partially calculate the tasks offloaded from the user equipment (UE), and the remaining task is processed locally in the UE. By jointly optimizing user scheduling, cloud offloading ratio, VES offloading ratio, and VES mobility, the objective function is to minimize the delay of the system transmission and computation under the constraints of discrete variables and energy consumption. To solve the problem, a primal-dual deep deterministic policy gradient (PD-DDPG) algorithm based on multi-tier computation offloading is proposed. Simultaneously, compared with baseline algorithms, PD-DDPG algorithm has an obvious advantage in both the speed of convergence and the system delay.

Index Terms—Multi-tier computing, computation offloading, deep primal-dual deterministic policy gradient, vehicle edge server.

I. INTRODUCTION

With the exponential growth of the artificial intelligence (AI) and 6G communication, there will be an explosion of computing-intensive mobile applications, such as intelligent driving, virtual reality, teledicine, smart factories [1]–[3], etc. As an effective solution to data explosion, cloud computing can provide powerful computing capabilities and centralized management guarantees for 6G communication network [4]–[6]. However, cloud computing has limitations. For instance, tasks uploading to the cloud computing center could create a long transmission delay, which is hard applied to the real-time computing tasks [7], [8], for example, the millisecond-level delay required for vehicle collision detection and warning is difficult to achieve in smart traffic scenarios [9], [10]. Moreover, such as video surveillance, uploading large-scale raw data to the cloud computing center requires a lot of bandwidth resources, which can bring huge pressure on the limited transmission bandwidth and creates huge computing costs [11], [12]. To solve the aforementioned problem, mobile edge computing (MEC) was presented to provide resources including computation, storage, network and communications at the edge of the mobile network [13]–[15], through the sinking of services and functions located in cloud to the edge of the network. In such a situation, the delay of the network responding to user requests is greatly reduced, and the possibility of network congestion in the transport network and the core network part is also reduced [16].

Although MEC has many advantages compared to cloud computing, there are some challenges of computing resources in MEC. The main problems to be solved are whether to offload, when to offload, and how many to offload. Many researchers have studied the offloading strategy under different performance requirements [17], [18]. Related works on MEC computing offload optimization about different offloading decisions have been investigated [19], [20], and partial offloading decision performs better than binary one. In [21] and [22], authors proposed solutions of the cloud with finite capacity and algorithms to reduce the complexity of calculating thresholds. In [23], an intelligent task offloading framework is proposed to handle the demanding challenge of delay-sensitive automotive tasks. In [24], an aerial-terrestrial communication system is designed to improve transmission strategy. To get lower delay transmission, this paper considered partial offloading decision on all local tier, edge tier, and cloud tier.

Since traditional edge servers are usually fixed on road side units (RSU) [25] or cellular base stations, the case of deploying MEC servers in mobile vehicles was considered to expand the range of computing services in this paper. To meet the low latency demand for task offloading, many
researchers have studied the mobility of edge servers and related resource allocation methods [26–29]. In [26], an MEC uninstalling scheme was proposed to reduce the cost of vehicular networks. In [27], considering the MEC-assisted mode, a joint strategy for vehicle and fixed servers was proposed to improve the vehicular network utility. According to the different situations, novel networks based on edge and fog computation were proposed in [28] and [29]. However, most of the existing literature consider the scenario of vehicles as user equipment (UE) asking for computing services, which may lead to service congestion resulting from a surge of service demands from UEs. In [30], multi-tier computing network can be a solution to the strict delay constraints problem, which can avoid service congestion through allocating computing tasks properly.

The emphasis of this paper is to design a multi-tier computing vehicular network, in which cloud servers and vehicle edge servers (VESs) can provide computing services for local UEs. A multi-tier computation offloading optimization problem is proposed to minimize the aggregate latency. The time-varying channel state and uncertain network communication information is considered. In [31], an AI architecture with integrated network resources and pervasive AI capabilities for supporting customized services is proposed. As an effective solution, machine learning has been applied in solving optimization problem [32]. In [33], the Q-learning algorithm is used to reduce the delay of computation task in a vehicle edge computing network. Though Q-learning algorithm, the Q value can be estimates accurately, and the increasing state space can cause dimensional explosion [34]. With deep neural network (DNN), deep reinforcement learning (DRL) can perform better than RL. [35–38]. Fig. 1 shows the development of the DRL algorithm including the above algorithms. Two RL algorithms: Deep Q-Network (DQN) [39] and primal-dual deep deterministic policy gradient (PD-DDPG) [40] based on multi-tier computation offloading was proposed to determine the policies of the optimization problem.

In this paper, a multi-tier computing architecture consisting of local tier, vehicle tier, and cloud tier is considered. When computing tasks can not be processed by the lower tiers, the upper tier cloud computing center can calculate rest of them. Network congestion and data accumulation can be effectively avoided. Owing to the fully utilized computing strength of multi-tier servers and the mobility of the edge server, the system delay (the total computing and transmission delay of all devices) may be substantially cut down. The following are the contributions of this paper.

- A cloud-edge-device collaborative computing network composed of cloud servers, VESs, and UEs is designed, in which both VES and cloud servers can provide computing services for UE to offload computing task.
- With random task traffic and time-varying channel states, user scheduling, task offloading, and VES mobility are considered. Then the non-convex optimization problem is described as a Constrained Markov decision process (CMDP). The objective function is to minimize the aggregate latency of tasks offloading while determining user device selection and task offloading ratios.
- Since computation offloading requires support for continuous action spaces and the CMDP model has high-complexity system states, a multi-tier computation offloading method based on PD-DDPG is proposed, which is the composition of actor network generating unique action and critic network solving the Q-value function approximately. The multi-tier computing offloading PD-DDPG algorithm is adopted to optimize strategy of user scheduling, task offloading, and VES mobility in the cloud-edge-device collaborative computing network.
- The performance of PD-DDPG algorithm is evaluated by simulations. The simulation outcomes show that the algorithm can obtain better performance compared with the existing algorithms under different random tasks, computing power, and number of users.

The rest of this paper is arranged as follows. Section II introduces the cloud-edge-device multi-tier computing structure. The third part introduces the objective function for the optimization issues. Section IV introduces the basic concepts of different algorithms in brief and proposes a multi-tier computing offloading algorithm based on PD-DDPG. Section V shows the simulation outcomes of PD-DDPG algorithm. Finally, Section VI summarizes the work of this paper.

II. SYSTEM MODEL AND FORMULATION OF OPTIMIZATION PROBLEM

A cloud-edge-device collaborative computing network model is designed, considering the computing offload and
resource allocation in the uplink cloud-edge collaborative access network. Fig. 2 is the cloud-edge computing network (CECN) architecture, which includes a series of remote radio heads (RRHs) with \( j \in \{1, 2, \ldots, J\} \) connecting directly to the cloud servers, a series of VESs with \( m \in \{1, 2, \ldots, M\} \) and all UEs that need to process task offloading are denoted as \( k \in \{1, 2, \ldots, K\} \). UEs are intelligent devices (smartphones, laptops, etc.) which contain processing and caching capabilities. In traditional MEC networks, edge servers connected to fixed base stations may be not capable of satisfying the communication and computing needs of UEs. Numerous vehicles in cities are frequently in contact with UEs and provide them with offloading services. Therefore, the edge computing server is extended to VES and any UEs can be a candidate for VES. In cloud mode, the UE can offload partial computing tasks to the cloud service center from RRHs. In VES mode, another part of offloading tasks can be processed by VESs, which includes intelligent vehicles that provide computing services in roads within the communication range. The DRL controller can make precise execution decisions through the UE’s location information, VES’s location information, congestion information, randomly generated computing tasks information, and available energy consumption information.

A. Communication Model

The whole communication cycle \( T \) are divided into \( I \) time slots, and UEs move at a low speed. Assuming that Poisson distribution computing tasks are generated. There are three offloading options for these tasks: local, cloud, and VES. Considering an actual scenario, the limitation of computing power makes UEs unable to afford complete task calculations locally. The UEs must offload partial tasks to the VES and cloud server. During each time slot, VESs stop at a fixed location and choose UE \( k \) to communicate.

When the UE selects the cloud mode, the user sends partial computing tasks to the cloud server through the RRH. Besides, transmission rate can permit quality of service. Therefore, the transmission rate from UE \( k \) to RRH \( j \) is given by:

\[
R_{j,k} = B \log_2(1 + \text{SINR}_{j,k}) \tag{1}
\]

where \( B \) is the communication bandwidth, and signal-to-interference-ratio (SINR) of UE \( k \) to RRH \( j \) can be formulated by:

\[
\text{SINR}_{j,k} = \frac{P_k g_{j,k}(i)}{\sum_{n \neq j} P_k g_{j,n}(i) + \sigma_j^2} \tag{2}
\]

where \( g_{j,k}(i) \) represents the channel gain of UE \( k \) to RRH communication radio link at time slot \( i \). \( \sigma_j^2 \) denotes the additive white Gaussian noise whose distribution is \((0, \sigma_j^2)\) received by RRH \( j \). \( P_k \) is the transmit power of UE \( k \).

Similarly, when the DRL controller decides to offload the computing task to VES \( m \), UE \( k \) offloads partial task to VES \( m \) through the wireless link. Combined with the reality of the scene and considering the situation of obstacles, the transmission rate is expressed as:

\[
R_{m,k} = B \log_2(1 + \text{SINR}_{m,k}) \tag{3}
\]

where the SINR of UE \( k \) to VES \( m \) denotes as:

\[
\text{SINR}_{m,k} = \frac{P_k g_{m,k}(i)}{\sum_{n \neq k} P_k g_{m,n}(i) + b_k(i) P_{\text{NLOS}} + \sigma_m^2} \tag{4}
\]

where \( \sigma_m^2 \) denotes the additive white Gaussian noise whose distribution is \((0, \sigma_m^2)\) received by the VES. \( P_{\text{NLOS}} \) denotes the radio transmission loss between UE \( k \) with VES, \( b_k(i)\) denotes the situation that there is obstacle or not between VES \( m \) and UE \( k \) at time slot \( i \) \cite{20}. \( g_{m,k}(i) \) denotes the channel gain of UE \( k \) to VES \( m \) communication wireless link at time slot \( i \), specifically,

\[
g_{m,k}(i) = \alpha_0 d_k^{-2}(i) = \frac{\alpha_0}{\|p(i + 1) - q_k(i)\|^2} \tag{5}
\]

where \( \alpha_0 \) is expressed as the channel gain when the reference distance is 1 m. \( d_k(i) \) is expressed as the Euclidean distance between UE \( k \) and the VES \( m \) at time slot \( i \).

B. Computing Model

Delay resulted in the CECN model include computational delay and transmission delay. The reason why computational
delays arise is that tasks are executed in different computing tiers, which include local tier, edge tier, and cloud tier. Compared to local processing, uploading tasks from UEs to edge servers or cloud servers could generate transmission delay. There is also a transmission latency generated by downloading calculated data. A new cloud-edge-device collaborative computing strategy is brought forward to minimize the latency by computing tasks under limited computing resources.

1) Local computing mode

In the CECN, the partial offloading strategy is used for the task computation execution of all UEs in each time slot. Among them, \( r_{c,k}(i), r_{f,k}(i) \in [0,1] \) represents the offloading ratio of cloud tier and VES tier, where \( r_{c,k}(i) + r_{f,k}(i) \leq 1 \). The local calculation delay of UE \( k \) at time slot \( i \) is formulated by:

\[
D_{\text{local},k}(i) = \frac{(1 - r_{c,k}(i) - r_{f,k}(i))M_k(i)s}{f_{1,k}},
\]

where \( M_k(i) \) is the task size to compute of UE \( k \), \( s \) represents the cycles of the CPU required to process each unit byte, \( f_{1,k} \) represents the computing power of UE \( k \).

2) VES computing mode

Due to the limited computing resources of UEs, when tasks are offloaded to VESs through wireless link, the delay includes transmission delay and computing delay. In CECN, the downloading delay from VESs can be neglected. The transmission delay of UE \( k \) to VES \( m \) can be expressed as:

\[
D_{\text{transform},k}(i) = \frac{r_{c,k}(i)M_k(i)}{R_{k,m}}.
\]

The computational processing delay on VES \( m \) can be expressed as:

\[
D_{\text{calculate},k}(i) = \frac{r_{c,k}(i)M_k(i)s}{f_{c,k}},
\]

where \( f_{c,k} \) is the calculate ability of VES.

3) Cloud computing mode

In certain cases, only joining VES to expand computing power still cannot meet the computing requirements of UEs, so that DRL controller needs to choose the cloud computing mode. Similar to the VES computing model, delay is generated by processing computing tasks and transmitting data. Additionally, the delay in the cloud mode is generated in two ways, which is from the UE \( k \) to the RRH \( j \) or the RRH \( j \) to the cloud servers. The transmission delay from UE \( k \) to cloud computing through RRH \( j \) can be expressed as:

\[
D_{\text{upload},k}(i) = \frac{r_{c,k}(i)M_k(i)}{R_{j,k}} + \frac{r_{c,k}(i)M_k}{R_{j,\text{cloud}}},
\]

where \( R_{j,c} \) represents the transmission rate from RRH \( j \) to cloud servers. The processing latency of UE \( k \) on cloud tier is formulated by:

\[
D_{\text{calculate},k}(i) = \frac{r_{c,k}(i)M_k(i)}{f_{c,k}},
\]

where \( f_{c,k} \) denotes the maximum processing capability that the cloud server can allocate to the UE \( k \) to complete computing tasks.

The amount of data uploaded is greater than the amount of data processed. Due that cloud servers are far away from UEs and are connected to core networks, the download delay can not be ignored, which can be expressed as follows:

\[
D_{\text{download},k}(i) = \frac{M_k^{\text{processed}}(i)}{R_{c,j}} + \frac{M_k^{\text{processed}}(i)}{R_{j,k}},
\]

where \( R_{c,j} \) denotes the downlink rate from the cloud servers to the RRU, and the uplink and downlink data rates are different. \( D_{f,k} \) and \( D_{c,k} \) denote the total delay generated from the VES computing mode and the cloud computing mode, respectively, which can be calculated by:

\[
D_{f,k} = \frac{r_{c,k}(i)M_k(i)}{R_{k,m}} + \frac{r_{c,k}(i)M_k}{R_{0,\text{cloud}}},
\]

\[
D_{c,k} = \frac{r_{c,k}(i)M_k(i) + M_k^{\text{processed}}(i)}{R_{0,k}} + \frac{r_{c,k}(i)M_k}{R_{c,0}}.
\]

C. Mobility Model

Considering the real situation, it is assumed that UEs move stochastically at low speed within range of CECN in each time slot, and only one VES is considered to communicate with UE and perform edge computing offload each time. Vehicle driving is completely safe during the entire communication process. The VESs are intelligent driving vehicles which are endowed with computing power, and they contain intelligent devices such as sensors to avoid accident and enhance the communication capability of vehicles. The VESs are only allowed to move within a secure area, whose boundary is shown in section V. At the time slot \( i \), the VES stops at a fixed position after driving on a safe road for a period of time. Then DRL controller selects one of the UEs to establish communication with VES. After offloading partial computing tasks to VES and cloud, the UE processes the remaining tasks locally. \( \mathbf{p}(i) = [x(i), y(i)]^T \) is expressed as the matrix vector of VES. VES travels from \( \mathbf{p}(i) \) to \( \mathbf{p}(i+1) \), and the moved position can be expressed as:
where \( v(i) \in [0, v_{max}] \) is expressed as the driving speed of VES, and \( \eta(i) \in \{0, \pm \frac{\pi}{2}\} \) is expressed as driving direction of VES at time slot \( i \).

### D. Energy Consumption Model

The energy consumption in CECN mainly includes VESs driving energy consumption and VESs operation energy consumption at time slot \( i \). Only VES tier is considered for cloud servers is far away from UEs and VESs. The energy consumption of the VES driving at time slot \( i \) can be formulated by:

\[
E_{\text{vehicle}}(i) = \frac{1}{2} \omega_v t_v \|v(i)\|^2, \tag{15}
\]

where \( \omega_v \) is the load condition of the VES, and \( t_v \) is the driving time of the VES. Correspondingly the energy consumption of VESs operation at time slot \( i \) is mainly used for transmitting and computing. When the calculation is performed on VESs, its computational power is given by:

\[
p_{f,k}(i) = \varphi f_{f,k}^3, \tag{16}
\]

where \( \varphi \) is the calculation factor of VES’power. Therefore, the computational energy consumption on the VES at time slot \( i \) is formulated by:

\[
E_{\text{calculate}}^{f,k}(i) = p_{f,k}(i) D_{\text{calculate}}^{f,k}(i). \tag{17}
\]

Based on models above, the task offloading optimization problem is summarized in the cloud-edge-device collaborative computing network system scenario. To minimize aggregate latency, the objective function of the problem is jointly optimizing user scheduling, VES mobility, task offloading, and energy efficiency controlling under the constraints of discrete parameters. Delay of all UEs at time slot \( i \) is reduced. The problem can ultimately be expressed as follows:

\[
\min \{k(i),r_{f,k}(i),r_{c,k}(i),v(i)\} \sum_{i=1}^{I} \sum_{k=1}^{K} \{D_{\text{local},k}(i), D_{f,k}(i) + D_{c,k}(i)\}
\]

s.t. C1 : \( \sum_{i=1}^{I} k(i) = 1, \forall i \in \{1, 2, ..., I\}, k \in \{1, 2, ..., K\} \),

\( C2 : \sum_{i=1}^{I} \sum_{k=1}^{K} k(i) M_k(i) = M, \)

\( C3 : r_{c,k}(i) + r_{f,k}(i) \leq 1, r_{c,k}(i), r_{f,k}(i) \in [0, 1], \)

\( C4 : b_k(i) \in \{0, 1\}, \forall i, k, \)

\( C5 : \sum_{i=1}^{I} (E_{f,k}^{\text{vehicle}}(i) + E_{f,k}^{\text{calculate}}(i)) \leq E, \forall k, \)

\( C6 : p(i) \in \{(x(i), y(i)) | x(i) \in [0, L], y(i) \in [0, W]\}, \forall i. \tag{18} \)

The constraint C1 means that only one UE can be selected for task calculation and offloading in each time slot. Constraint C2 guarantees that all task computations are completed within the entire communication time period. Constraint C3 limits the bounded range of edge offload ratio and cloud offload ratio. Constraint C4 represents the congestion situation of the radio channel between UE \( k \) and VES \( m \) within a time slot. Constraint C5 considers energy-saving energy consumption to ensure that the energy consumption of VESs traveling and computing within the time slot is under the maximum allowed energy consumption. Constraint C6 means that the VESs can only move within a given area.

## III. MDP and Solutions For Multi-tier Computing Offloading

In CECN system, more attention is paid to the container of task offloading. There is a set of computing tasks requiring to be offloaded within a period of time. In each time slot, the agent determines computation offloading, the corresponding offloading ratio and the VES real-time location, considering both user scheduling and task offloading optimization problems. The optimization problem is a non-convex problem and is intractable. Therefore, this task offloading problem is built as an MDP model, and a DRL algorithm is employed to efficiently find a solution. To begin with, the state space, action space, and reward function of the problem are defined. Then the latency minimization problem is described to solve with a DQN-based DRL approach. Furthermore, using PDDPG to train efficient and available policies is discussed.

### A. Environment Settings

This section models the problem of user scheduling and tasks offloading based on the DRL algorithm.
1) State space

In CECN, the VESs are equipped with smart sensors to obtain real-time information about the environment. In each time slot, the system states are composed of VES location information, UE location information, obstacle information, random computing task information, and energy consumption size information. The system state of the time period is expressed as:

$$S_i = \{p(i), q_k(i), b_k(i), M_k(i), M_{ef}(i), E(i)\}. \quad (19)$$

The meaning of each element is as follows:
- $p(i)$: The position information of VES.
- $q_k(i)$: The position information of UE $k$, where $k = 1, 2, ..., K$.
- $b_k(i)$: The congestion situation between the VES to offload and the chosen UE $k$ at time slot $i$.
- $M_k(i)$: The size of the computing task randomly generated by the UE at time slot $i$.
- $M_{ef}(i)$: The size of the remaining computing tasks which the system needs to process during the entire communication time.
- $E(i)$: The remaining energy which is available for the VES at time slot $i$. When $i = 1$, $E(i) = E_0$.

2) Action space

When the system takes a large number of actions, there will be such a large amount of computation that it can produce a large amount of delay and a lot of energy consumption. Therefore, to reduce computational complexity, only one UE at each decision moment is considered. The set of actions at time slot $i$ can be expressed as $A_i$, which includes the number of UE $k$ for offloading service, the offloading ratio of the UE to the VES, the offloading ratio of the UE to the cloud server and the driving distance of the VES. The actions in each action space are described as:

$$A_i = \{k(i), r_{f,k}(i), r_{c,k}(i), v(i)\}. \quad (20)$$

3) Reward function

An accurate reward function can assist in obtaining appropriate strategies, the objective function is usually used as the reward function in an efficient learning process. The target is to minimize the system latency defined in formula (18) through maximizing reward function, so set the reward function as:

$$R_i = R(S_i, A_i) = -D_{system}(i), \quad (21)$$

where the system total delay at time slot $i$ is given by:

$$D_{system}(i) = \sum_{k=1}^{K} \max\{D_{local,k}(i), D_{f,k}(i) + D_{c,k}(i)\}. \quad (22)$$

B. Solution Based on Q-Learning

The proposed multi-tier computing offloading process can be approximated as a MDP. The MDP model can be defined as $\{S_i, A_i, P(S_{i+1}|S_i, A_i), R_i\}$, where $P(S_{i+1}|S_i, A_i)$ denotes the state transition probability. $A_i$ denotes the action executed by DRL controller at time slot $i$, and $S_{i+1}$ represents the system state at the next stage.

Actually, state transition probabilities are hard to obtain. Q-learning is a classic RL algorithm, in which agent learns Q-values of states and actions, then selecting the action which corresponds to the highest Q-value. The Q-function returns the maximum expected reward according to the policy $\pi$ as follows:

$$Q^\pi(S, A) = \mathbb{E}[\sum_{t=0}^{T} \frac{\xi R_t}{(1 - \gamma)^t} | S_0 = S, A_0 = A, \pi], \quad (23)$$

where $\xi$ is the discount factor, and $R_t$ is the reward at time slot $i$. $S_0$ is the initial state. $Q^*$ insures the maximum value of accumulated rewards, which can be formulated by:

$$Q^*(S, A) = \max_\pi Q^\pi(S, A). \quad (24)$$

According to Bellman formula [27], $Q^*$ can be expressed as:

$$Q^*(S_i, A_i) = E_{i+1}[R_{i+1} + \xi \max_{A_{i+1}} Q^*(S_{i+1}, A_{i+1})]. \quad (25)$$

In general, Q-function is implemented recursively by exploiting $(S_i, A_i, R_i, S_{i+1})$, which includes the current state, action, instant reward, and transition state information for the next time slot $i + 1$. Therefore, the Q function is updated as follows:

$$Q_{i+1}(S, A) = \alpha R_i + \xi \max_{A_{i+1}} Q(S_{i+1}, A_{i+1}) + (1 - \alpha)Q(S_i, A_i), \quad (26)$$

where $\alpha$ is the learning rate. Through choosing an appropriate learning rate, the iterative algorithm can ensure $Q_{i+1}(S, A)$ converges to $Q^*(S, A)$.

C. Solution Based on DQN

The Q-learning algorithm is a different-policy temporal difference learning method that was proposed earlier. DQN uses the neural network to approximate the value function in Q-Learning and make improvements for practical problems.
Because DQN is one of the baseline algorithms, the specific algorithm process will not be expanded, the process of which is shown in Algorithm 1. A neural network needs to be defined for classification, a loss function need to be defined similar to Q-learning. By minimizing the loss function, the deep Q function is trained to the target value. The loss function can be expressed as:

\[
L(\theta) = \mathbb{E}[(R_i + \gamma \max Q(S_{i+1}, A_{i+1})|\theta) - Q(S_i, A_i|\theta)]^2.
\]

(27)

Algorithm 1 DQN algorithm for Multi-tier Computation Network

1. **Input:** Discount factor $\gamma$; Learning rate $\alpha$; Parameter update interval $C$;
2. Initialize: Experience replay memory $\mathcal{D}$, and the capacity is $N$;
3. Initialize randomly: Parameter of Q network $\phi$;
4. Initialize randomly: Parameter of Q target network $\phi'$;
5. repeat
6. Initialize state $S_i$ in (19);
7. repeat
8. Select action $A_i = \pi^\ast$ in (20) at state $S_i$;
9. Execute action $A_i$, observe environment, get online reward $R_i$ and new state $S_i$;
10. Put $(S_i, A_i, R_i, S_{i+1})$ into $\mathcal{D}$;
11. Sample $(S_j, A_j, R_j, S_{j+1})$ from $\mathcal{D}$;
\[
y = \begin{cases} R_j, & S_j \text{ is the end state,} \\ R_j + \gamma \max_{a_i} Q_{\phi'}(S_j, A_j), & \text{otherwise.} \end{cases}
\]
\[
(28)
\]
12. Train Q network with the loss function $L(\theta)$ from (27);
13. Update state $S_i \leftarrow S_{i+1}$;
14. Update $\phi' \leftarrow \phi$ every $C$ steps;
15. until $S_i$ is the end state
16. until $Q_{\phi}(S, A)$ convergence

**Output:** The Q network $Q_{\phi}(S, A)$.

In section V, the baseline algorithms above are implemented. To solve the multi-tier computing offloading problem, the problem is converted to a CMDP according to the real situation and the PD-DDPG algorithm is proposed. Besides obtaining the proposed algorithm, the performance of different DRL algorithms is compared in the part of Simulation.

IV. SOLUTION BASED ON PRIMAL-DUAL DDPG ALGORITHM

It is essential to ensure the security of agents in some practical scenarios. Different from the standard DDPG with MDP, which only needs to maximize the reward function, the action taken by the agent needs to avoid the dangerous situation at different states. Therefore, long-term discounted reward is considered to achieve balance between maximizing the reward and reducing the risk cost.

A. Constrained Markov Decision Process

To satisfy the long-term discounted reward under policy $\pi$, a CMDP is proposed. In CMDP, the constrains on long-term discounted costs is augmented to an MDP [34], where cost functions $\chi_1, \chi_2, ..., \chi_n$ are added to the traditional MDP. Each $\chi_n$ is a mapping from transition tuples to costs. The aim of CMDP is to choose the policy $\pi$ to maximize the long-term reward of (21) with the constrains of $\chi_n(\pi) \leq \mu_n, \forall n \in [1, 2, 3, ..., N]$, where $\mu_n$ is the corresponding limit. The policy can be given by:

\[
\pi^\ast = \arg \max \ R_i(\pi) \quad \text{s.t.} \quad \chi_n(\pi) \leq \mu_n, \forall n \in [1, 2, 3, ..., N].
\]

(29)

B. Primal-Dual DDPG algorithm for CMDP

Compared with DQN, the DDPG algorithm mainly solves the prediction problem of continuous action spaces. The difference of implementation is mainly in the choice of the final activation function, whether the action spaces are continuous or discrete. The actor network in the DDPG algorithm outputs continuous actions, so it is necessary to process some action variables. When the agent chooses the action variable $k(i) \in [0, K]$, it needs to be discretized. If $k(i) = 0$, $k' = 1$; If $k(i) \neq 0$, $k' = \lceil k(i) \rceil$, where $\lceil \cdot \rceil$ is the ceiling operation [20]. The task offloading ratio and speed of VES are continuous action variables. All the above action variables are optimized in a joint effort to achieve the system delay minimization. Through the introduction in [34], a primal-dual DDPG algorithm is given to solve CMDPs. Different from DDPG, PD-DDPG uses the off-policy data to update the primal policy and the dual variable. The flow of the PD-DDPG algorithm is shown in Algorithm 2.

The entire training process of PD-DDPG is summarized as shown in the Fig. 3. Compared with the network structure of DDPG algorithm, PD-DDPG added a neural network to represent the long-term discount cost. The PD-DDPG’s neural networks for functional positioning are as follows:

- $Q$ online network: Iterative update of policy network parameter $\theta$. Picking the current action $A_i$ on the base of
Algorithm 2 Primal-Dual DDPG algorithm for Multi-tier Computation Network

1: **Input:** Discount factor $\gamma$; Soft-max update factor $\tau$; Learning rate $\alpha_a$ and $\alpha_c$; Episode of training $X$; Sample length of training $H$;
2: **Initialize:** Experience replay memory $M$; Network parameters $\theta$, $w_r$, $w_c$;
3: for each episode $x = 1, 2, ..., X$ do
4: Reset and obtain initial state $S_0$ in (19) from CECN function;
5: for $h = 1, 2, ..., H$ do
6: State normalization: $S_i \rightarrow S_i'$;
7: Obtain the action $A_i = \mu(S_i' | \theta) + N_i$ and the behavior noise $N_i$;
8: Execute action $A_i$ in (20), and get the reward $R_i$ from (21) and the next stage state $S_i + 1$;
9: if $M$ is not full then
10: Put $(S_i', A_i, R_i, S_{i+1}')$ into $M$;
11: else
12: Replace a transition in $M$ with $(S_i, A_i, R_i, S_{i+1})$ randomly;
13: Sample $(S_j', A_j, R_j, S_{j+1}')$, $j = 1, 2, ..., H$ randomly from $M$;
14: Set $y_i = R_i + \gamma Q_c(S_{i+1}, A_{i+1}, \omega_c')$; $z_i = R_i + \gamma Q_c(S_{i+1}, A_{i+1}, \omega_c)$;
15: Through minimizing the loss function in (31) and (32), update $\omega_c$ and $\omega_c'$;
16: Update $\theta$ through (33);
17: Update the actor policy in (36);
18: Update the dual variable with dual gradient
\[
\nabla_{\lambda} Q_c(S_i, A_i, \omega) = \frac{1}{m} \sum_{j=1}^{m} \nabla Q_c(S_i, A_i, \omega_c)_{S_i=A_i} - \mu;
\]
19: Soft update: $\omega'_c, \omega''_c$, and $\theta'$ in (30) – (32);
20: end if
21: end for
22: end for
23: **Output:** The actor network from (36).

the current state $S_i$ for interacting with the environment to generate $S_{i+1}$ and $R_i$.

- **Q target network:** Choose the next action $S_{i+1}$ based on the next state $A_{i+1}$ sampled from the experience replay pool and $\theta \rightarrow \theta'$.
- **Reward critic online network:** Iterative update of $Q$ network parameter $\omega_r$ and calculate $Q_i(S_i, A_i, \omega_r)$. The $Q$ target value is $y_i = R_i + \gamma Q_{i+1}(S_{i+1}, A_{i+1}, \omega_c')$.
- **Reward critic target network:** Update $\omega \rightarrow \omega'_r$ and calculate the $Q_{i+1}(S_{i+1}, A_{i+1}, \omega'_r)$.
- **Cost critic online network:** Iterative update of $Q$ network parameter $\omega_c$ and calculate $Q_i(S_i, A_i, \omega_c)$. The $Q$ target value is $z_i = R_i + \gamma Q_{i+1}(S_{i+1}, A_{i+1}, \omega_c)$.
- **Cost critic target network:** Update $\omega_c \rightarrow \omega'_c$ and calculate the $Q_{i+1}(S_{i+1}, A_{i+1}, \omega'_c)$.

Replication of DDPG from online network to target network is not the same as DQN, which directly copy the parameters of the $Q$ online network to the $Q$ target network. DDPG chooses soft updating rather than $\omega' = \omega$, which means that only a small part of the parameters are updated each time. It can be expressed as:

\[
\begin{align*}
\theta' & \leftarrow \theta + \tau \theta_C + (1-\tau)\theta, \\
w_r' & \leftarrow w_r + \tau w_r + (1-\tau)w_r, \\
w_c' & \leftarrow w_c + (1-\tau)w_c,
\end{align*}
\]

where $\tau$ is the update coefficient generally denoted as a small value. At the same time, to add randomness in the learning process to increase the coverage of learning, a certain amount of noise $N_i$ is added to the selected action $A_i$. The expression of the action that finally interacts with the environment is:
\[ A_i = \pi_\theta(S_i) + \mathcal{N}_i. \]  

To minimize the loss function, an optimizer is used for updating \( Q \) value in the critic network. Similar to the DQN, the loss function of reward critic network can be expressed in the form of mean squared error:

\[ J(\omega_r) = \frac{1}{m} \sum_{i=1}^{m} (y_i - Q_r(\phi(S_i), A_i, \omega_r))^2. \]  

Similarly, the loss function of cost critic network is given by:

\[ J(\omega_c) = \frac{1}{m} \sum_{i=1}^{m} (z_i - Q_c(\phi(S_i), A_i, \omega_c))^2. \]  

PD-DDPG executes a determined strategy so the loss function of actor network can be expressed as:

\[
\nabla_{\pi}(\theta, \lambda) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\pi}[Q_r(S_i, A_i, \omega_r)]_{s=s_i, A=\pi_\theta(s_i), \lambda} - \lambda Q_c(S_i, A_i, \omega_c)]_{s=s_i, A=\pi_\theta(s_i)}.
\]

If two different actions \( A_1 \) and \( A_2 \) are output for the same state, there can be two feedback \( Q \) values from the online network of Critic, which are \( Q_1 \) and \( Q_2 \) respectively. \( Q_1 > Q_2 \) means taking action \( A_1 \) can get more reward than \( A_2 \). According to the policy gradient, it is no doubt to increase the occurrence probability of \( A_1 \) and decrease the occurrence probability of \( A_2 \), which means actor network is desire for a larger \( Q \) value as much as possible. Therefore, the smaller the feedback \( Q \) value the agent obtained, the greater the loss it can get. Therefore, the loss function is calculated by:

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [Q_r(S_i, A_i, w_r) - \lambda Q_c(S_i, A_i, w_c)]. \]  

\[ \text{TABLE I} \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )</td>
<td>4</td>
<td>( \omega_r )</td>
<td>10 kg</td>
</tr>
<tr>
<td>( I )</td>
<td>20</td>
<td>( t_{drive} )</td>
<td>10 s</td>
</tr>
<tr>
<td>( B )</td>
<td>1 MHz</td>
<td>( P_{NLOS} )</td>
<td>20 dB</td>
</tr>
<tr>
<td>( E )</td>
<td>500 KJ</td>
<td>( f_{t,k} )</td>
<td>0.6 GHz</td>
</tr>
<tr>
<td>( L, W )</td>
<td>1000 m</td>
<td>( T )</td>
<td>200 s</td>
</tr>
<tr>
<td>( v_{\text{max}} )</td>
<td>10 m/s</td>
<td>( \alpha_0 )</td>
<td>-50 dB</td>
</tr>
<tr>
<td>( \sigma^2_1, \sigma^2_2 )</td>
<td>-100 dBm</td>
<td>( s )</td>
<td>1000 cycles/bit</td>
</tr>
<tr>
<td>( P_k )</td>
<td>0.1 W</td>
<td>( M )</td>
<td>9.65 KG</td>
</tr>
<tr>
<td>( f_{c,k} )</td>
<td>1.2 GHz</td>
<td>( f_{s,k} )</td>
<td>240 GHz</td>
</tr>
</tbody>
</table>

\[ \text{V. Simulations and Discussions} \]

In this section, the feasibility of a multi-tier computation offloading PD-DDPG framework is demonstrated in the CECN system through simulations under different parameters. It begins with the simulation parameters settings. Then, the implementation performance of the PD-DDPG algorithm is validated under various settings. The algorithm is shown to be significantly better than other methods.

\[ \text{A. Simulation Settings} \]

In CECN, a square area 1000 \( \times \) 1000 \( m^2 \) is designed. The whole communication time period is \( T = 200 \) s, which is divided into 20 time slots. The maximum drive speed of VES is 10 \( m/s \). The communication bandwidth is 1 MHz. The noise power of the receiver is -50 dBm. In the VES computing mode, \( b_t \) means the signal is blocked between UE \( t \) and VES. The transmission power of UE is 0.1 W, and the CPU cycles is 1000 cycle/bit. It is assumed that the energy capacity is 5000 KJ. The computational power of different tier is 0.6 GHz, 1.2 GHz, and 240 GHz. The detailed parameter settings are shown in Table 1. Simulation is set as above to compare the advantages and disadvantages of the algorithm under the condition of simulating the real situation.

The multi-tier computation offloading algorithm based on PD-DDPG is compared with other methods:

- **Without offloading**: Assumed that all computing tasks are processed locally without other computing tiers.
- **PD-DDPG without cloud tier**: Assumed that partial computing tasks can only be offloaded to VES.
- **PD-DDPG without VES tier based on**: Assumed that partial computing tasks can only be offloaded to VES.
- **DQN based on computation offloading**: A discrete action spaced based DQN computation offloading algorithm is discussed, where vehicle speed and offloading ratio are divided into ten small discrete action spaces, and they can be expressed as \( v(i) \in \{0, \frac{v_{\text{max}}}{10}, \frac{v_{\text{max}}}{9}, ..., \frac{v_{\text{max}}}{1} \} \), \( r_{e,k}(i) \in \{0, 0.1, ..., 1\} \), and \( r_{c,k}(i) \in \{0, 0.1, ..., 1\} \).

\[ \text{B. Results and Discussions} \]

A series of simulation experiments are executed to demonstrate the algorithm performance and the optimal values of hyper parameters under different conditions, respectively. Fig. 4 shows the performance results among the PD-DDPG
algorithm and other schemes. The algorithm is trained for a total of 1000 episodes, where both DQN and PD-DDPG can converge with increasing the number of iterations. The result shows that the utility of PD-DDPG is significantly better than DQN, for both DQN and DDPG contain a policy network and a target network. The dual-network structure can find the best action policy by isolating the correlation between training data. Different from the DQN algorithm, the PD-DDPG algorithm can output continuous actions, which is more advantageous in selecting action spaces $r_{f,k}(i), r_{c,k}(i)$. Simultaneously, the situation of only considering the local tier and not considering the local tier is also compared, and the results after the convergence of the PD-DDPG algorithm are obviously better than the delay results of the two.

Fig. 5 shows the convergence of PD-DDPG at various learning rates. PD-DDPG can converge and achieve better effect when $lr_a = 0.001$, $lr_c = 0.002$, and $lr_{lam} = 0.002$. When $lr_a = 0.01$, $lr_c = 0.02$, and $lr_{lam} = 0.02$, PD-DDPG can converge with lower reward. When $lr_a = 0.0001$, $lr_c = 0.0002$, and $lr_{lam} = 0.0002$, PD-DDPG cannot converge. The reason is that a higher learning rate will cause both the actor network and critic network to take larger updating steps, and a lower learning rate will cause the neural network to update slower, which requires more iterations to obtain better results.

Fig. 6 shows the effect of inappropriate state normalization or behavioral noise on training. When there is no behavior noise, the convergence of the algorithm is obviously not as good as adding behavior noise. When the algorithm is trained without state normalization, the PD-DDPG algorithm cannot converge normally for the missing scaling factor and the larger value of state space. In this way, the neural network is randomly initialized to output a larger value, which can be regarded as a greedy algorithm. Fig. 7 shows the cases where only offloading to the cloud and the VES, and it can be obviously seen that the convergence effect of the algorithm considering both offloading approaches is significantly better than only considering one case. In addition, PD-DDPG without VES converges better than PD-DDPG without cloud. This is because the cloud servers have more powerful computing resources than VES. The better utility of the PD-DDPG algorithm can be achieved only when multiple offloading modes are considered at the same time.

Fig. 8 shows the average delay size of PD-DDPG algorithm, DQN algorithm, locally, and without considering one computing case under different task sizes. In Fig. 8, the total latency of PD-DDPG algorithm is significantly lower than the DQN algorithm for tasks of the same size, for it is necessary to explore the non-negligible space between the discrete action space and the available actions. It is difficult to find better offloading strategy in the DQN algorithm. Moreover,
PD-DDPG can explore the continuous action space to obtain appropriate policies by taking precise actions. Furthermore, with increasing the task size, latency of the PD-DDPG algorithm grows much slower than other schemes, which further demonstrates the advantages of the PD-DDPG algorithm. Fig. 9 shows the total delay under different transmit power levels. Similarly, the proposed algorithm is superior to other algorithms.

Fig. 10 shows the total delay among the proposed algorithm, DQN algorithm, and without computation offloading under different UE CPU frequencies. Under different UE computing capabilities, the PD-DDPG algorithm achieves lower delay, for the offloading ratio is a precise factor that has a great influence on the delay with continuous action controlling of PD-DDPG.

VI. CONCLUSION

Aiming at the problem of multi-user joint user scheduling, VES mobility, and task offloading optimization in multi-tier computing networks, user scheduling, cloud offloading ratio, VES offloading ratio, and VES mobility are considered. Discrete variables and energy consumption are constraints to minimize the system delay in the entire communication period. The proposed multi-tier computation offloading PD-DDPG algorithm abstracted system resources and offloading strategies into environment states and actions vectors with CMDP, and applies DNN to offloading decision prediction. The appropriate offloading strategy is gained through multiple iterations. Simulation experiments were designed to evaluate the convergence and operation of the algorithm. Simulations indicated that the PD-DDPG algorithm converges under different hyper parameters and outperforms other offloading strategies under different experimental conditions. Therefore,
the proposed algorithm can effectively reduce the latency of the CECC system.

REFERENCES


Haijun Zhang (M’13, SM’17) is currently a Full Professor and Associate Dean in School of Computer and Communications Engineering at University of Science and Technology Beijing, China. He was a Postdoctoral Research Fellow in Department of Electrical and Computer Engineering, the University of British Columbia (UBC), Canada. He serves/served as Track Co-Chair of WCNC 2020/2021, Symposium Chair of Globecom’19, TPC Co-Chair of INFOCOM 2018 Workshop on Integrating Edge Computing, Caching, and Offloading in Next Generation Networks, and General Co-Chair of GameNets’16. He serves as an Editor of IEEE Transactions on Communications, and IEEE Transactions on Network Science and Engineering. He received the IEEE CSIM Technical Committee Best Journal Paper Award in 2018, IEEE ComSoc Young Author Best Paper Award in 2017, and IEEE ComSoc Asia-Pacific Best Young Researcher Award in 2019.

Lizhe Feng received the B.S. degree from the School of Computer and Communication Engineering, University of Science and Technology Beijing, China, in 2020, where he is currently pursuing the M.S. degree. His research interests include wireless communication, deep reinforcement learning, and multi-tier computing.

Xiangnan Liu received the B.S. degree from the School of Computer and Communication Engineering, University of Science and Technology Beijing, China, in 2019. He is currently pursuing his Ph.D. degree at University of Science and Technology Beijing, China. His research interests include access control, beamforming, and resource allocation in 6G wireless communication.

Keping Long (SM’06) received the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, in 1995 and 1998, respectively. From September 1998 to August 2000, he was a Postdoctoral Research Fellow at the National Laboratory of Switching Technology and Telecommunication Networks, Beijing University of Posts and Telecommunications (BUPT), China. From September 2000 to June 2001, he was an Associate Professor at BUPT. From July 2001 to November 2002, he was a Research Fellow with the ARC Special Research Centre for Ultra Broadband Information Networks (CUBIN), University of Melbourne, Australia. He is currently a professor and Dean at the School of Computer and Communication Engineering, University of Science and Technology Beijing. He has published more than 200 papers, 20 keynote speeches, and invited talks at international and local conferences. His research interests are optical Internet technology, new generation network technology, wireless information networks, value added services, and secure technology of networks. Dr. Long has been a TPC or ISC member of COIN 2003/05/06/07/08/09/10, IEEE IWCN 2010, ICON2004/06, APOC2004/06/08, Co-Chair of the organization Committee for IWCMC2006, TPC Chair of COIN 2005/08, and TPC Co-Chair of COIN 2008/10. He was awarded by the National Science Fund for Distinguished Young Scholars of China in 2007 and selected as the Chang Jiang Scholars Program Professor of China in 2008. He is a member of the Editorial Committees of Sciences in China Series F and China Communications.

George K. Karagiannidis (Fellow, IEEE) was born in Pithagorion, Samos Island, Greece. He received the University Diploma (five years) and Ph.D. degrees in electrical and computer engineering from the University of Patras in 1987 and 1999, respectively. From 2000 to 2004, he was a Senior Researcher at the Institute for Space Applications and Remote Sensing, National Observatory of Athens, Greece. In June 2004, he joined the Faculty of the Aristotle University of Thessaloniki, Greece, where he is currently a Professor with the Electrical and Computer Engineering Department and the Head of the Wireless Communications and Information Processing (WCIP) Group. He is also an Honorary Professor at Southwest Jiaotong University, Chengdu, China. His research interests are in the broad area of digital communications systems and signal processing, with emphasis on wireless communications, optical wireless communications, wireless power transfer, and applications and communications and signal processing for biomedical engineering. Dr. Karagiannidis has been involved as the general chair, the technical program chair, and a member of technical program committees in several IEEE and non-IEEE conferences. He is one of the highly-cited authors across all areas of electrical engineering, recognized from Clarivate Analytics as the Web of Science Highly-Cited Researcher in the six consecutive years 2015–2020. In the past, he was an editor in several IEEE journals. From 2012 to 2015, he was the Editor-in-Chief of IEEE Communications Letters. He serves as the Associate Editor-in-Chief for IEEE Open Journal of the Communications Society.