2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)

2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom) | 978-1-6654-4377-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSCloud-EdgeCom52276.2021.00040

A Distributed Dependency-Aware Offloading Scheme for Vehicular Edge Computing Based on Policy Gradient

Haoqiang Liu, Hongbo Zhao*, Liwei Geng, Yujie Wang, Wenquan Feng
School of Electronic and Information Engineering
Beihang University
Beijing, China
lhqbuaa@buaa.edu.cn

***Abstract*—The proliferation of smart transportation has significantly promoted explosive growth of the Internet of Vehicles (IoV). Especially, with the rapid development of the 5-th generation mobile networks (5G), amounts of computation-intensive and latency-sensitive vehicle applications impose heavy pressures to such limited vehicles. To address this issue, Multi-access Edge Computing (MEC) has served as a promising paradigm to facilitate those devices at the edge of wireless networks. Computation offloading is a critical technology that decides which tasks should be offloaded for minimizing the energy and time cost. However, conventional methods are inefficient to deal with dependency-aware subtask topologies from vehicles, which leads to low efficiency and waste of edge resources, especially for multi-vehicles scenarios. In this paper, we investigate the subtask topologies of IoV applications. Directed acyclic graphs (DAGs) are utilized to obtain the priority of task scheduling. Further, take privacy protection and optimal resource allocation into consideration, we put forward a distributed deep reinforcement learning (DRL) strategy based on policy gradient without information sharing in edge computing. Especially, both actor and critic network employ convolutional layer and transformer to learn the optimal mapping from the input states to the offloading decision for each subtasks. Numerical results show that, for various experimental settings, compared with the existing offloading methods, the proposed scheme achieves the superior performance.**

Keywords*—Multi-access Edge Computing, Dependency-Aware, Computation Offloading, Deep Reinforcement Learning, Vehicular Edge Networks

I. INTRODUCTION

Recent years have witnessed the continuous advancement of Internet of Things and communication technology. The explosive growth of new mobile applications, such as virtual reality, leads to heavy consumption of resource on mobile devices (MDs), especially for the Internet of Vehicles (IoV) [1][2]. However, many resource-constrained devices often suffer from failing to handle such applications as they need intensive resources and latency guarantees [3][4]. To alleviate this issue, Multi-access Edge Computing (MEC) is perceived as a powerful paradigm that facilitates devices in the proximity with large amounts of resources and has attracted significant attention [5]. MEC can satisfy the requirements of mobile applications and empower resource-limited devices to offload tasks to the edges of networks [6]. Compared to traditional cloud computing, MEC efficiently deals with the drawbacks of long backhaul latency and high network overload [7].

With the revolution of communication technologies and networks, IoV has been regarded as one of the most critical technologies nowadays. With the emergence of driverless vehicles, multitudes of computation-intensive and latency-sensitive applications have emerged nowadays [8]. Employing the offloading technology of MEC, vehicular edge computing networks (VECNs) has served as a potential method [9][10]. By jointing edge servers with the road side units (RSUs), VECNs provide services to vehicles. This paper concentrates on the offloading strategy in vehicular edge networks.

Despite of the virtue of MEC in augmenting computation capability of vehicles, to develop a comprehensive offloading scheme remains challenging. As the critical technology in edge computing, computation offloading refers to designing an powerful offloading method to determine which subtasks should be offloaded [11]. It is very necessary to achieve efficient offloading method for the resource limited vehicles to minimize the energy and time cost.

Current work is mainly focus on binary offloading and simple partial offloading. In practice, applications on vehicles usually consists of several components with complicated structure and the dependency among them cannot be ignored. Given a typical navigation task that can be divided into several subtasks [12], a reliable offloading is to deal with parallel subtasks at edge servers and devices concurrently for computation latency reduction. Unfortunately, existing researches often overlook subtask topologies with dependency-aware, which leads to low efficiency and waste of edge resources, especially for multi-vehicles scenarios. To guarantee the dependency-aware among subtasks, researchers begin to consider subtasks as a chain-structured or tree-structured task. Nonetheless, such topologies become unsuitable for complicated tasks. According to task topologies, directed acyclic graphs, i.e. DAGs, are utilized to elaborate the priority of task scheduling in our work.

Although previous methods on computation offloading, such as game theory and intelligent optimization algorithms, have

978-1-6654-4377-7/21/$31.00 ©2021 IEEE
DOI 10.1109/CSCloud-EdgeCom52276.2021.00040

achieved improvements to a certain extent, they could not be sufficient in practice since they only take one-shot optimization into consideration. Fortunately, in recent years, reinforcement learning (RL) shows advantages in task offloading because it can obtain the best policy and maximize rewards without information of systems [13]. Further, deep reinforcement learning (DRL) enables RL to learn large state spaces with the deep neural networks [14-16]. Therefore, in our offloading scheme, DRL is employed for VECNs.

Recent works require centralized control to ensure the optimal performance, which is not practical because users may be unwilling to expose personal information. Therefore, in this paper, considering privacy protection and optimal resource allocation into consideration, we put forward a distributed DRL strategy based on policy gradient without information sharing in edge computing for minimizing the total cost in multiple vehicles situation. Simulation results reveal that, compared with the existing offloading methods, the proposed scheme can improve system utility efficiently. In this paper, there are three-fold main contributions:

1) We elaborate the priority of task scheduling with DAGs, which considers the dependency among subtasks.

2) Due to privacy protection and optimal resource allocation, a distributed offloading scheme based on policy gradient without information sharing is proposed with dependency-aware in multiple vehicles scenario.

3) Simulations are conducted to reveal the advantage of the proposed PGDCO.

The rest of the paper is organized as follows. Section II presents related works about computation offloading. Section III introduces the basic model and formulates mathematical problem. Section IV gives a DRL-based distributed offloading scheme, followed by evaluations and analysis in Section V. Then, Section VI gives the conclusion.

II. Related Works

MEC was proposed in 2014, which aims to build an open platform on the radio access network side. In recent years, MEC has been recognized as a critical solution to enhance performance and attracted large amounts of attention all over the world. As a critical issue in MEC, computation offloading is recognized as a hot topic and widely applied for resource-limited vehicles with latency-sensitive applications. The efficient offloading configurations is concentrated on minimizing the total delay of applications in VECNs.

In order to make full use of resources, extensive works have focus on how to obtain the best offloading performance in MEC with methods such as game theory and mathematical programming. For example, [17] introduced an optimization offloading method which minimized communication and energy costs, and verified that it was practical to improve Quality of Service (QoS) of users. Zhao gave a resource allocation method that was suitable for the heavy computation and limited edge resources situations. Nevertheless, those approaches could not deal with vehicular networks well as high dynamic environments impose a heavy impact in long-term performance.

Fortunately, the breakthroughs of DRL provide alternative approaches to handle those problems for edge computing. Recent researches have obtained superior performance over a long term in the time-varying network status. You performed a centralized offloading method in system with TDMA, and it minimized the energy consumption [18]. Alam put forward a deep Q-learning-based framework, which handled the resource demand from mobile devices in edge network [19]. Jiang modeled the caching problem and proposed a DRL-based strategy to coordinate the caching decisions [20]. Meanwhile, taking large action space into account, this paper gave an upper confidence bound approaches. Although those works have studied computation offloading from the perspective of both single device and multi-devices, they require centralized control with all prior knowledge of devices. This may fail to meet the interactions among devices as they take offloading actions independently. Moreover, users may be unwilling to expose personal information because of privacy protection [21], which makes centralized schemes unpractical. Therefore, the research on distributed DRL offloading schemes has attracted extensive attention [22][23]. We concentrate on distributed strategies with multiple independent offloading systems deployed.

Note that task can be mainly divided into divisible and atomic models. For atomic models, tasks must be offloaded to servers or executed locally. For divisible models, task graphs have been paid attention and the dependency relationship between subtasks should be ensured. Such dependency is described by a DAG in this paper.

III. System Model and Problem Formulation

As shown in Fig. 2, vehicular edge networks can be divided into several cellular networks according to geographic information. Each cellular network consists of a base station (BS) connected with central mobile edge computing (CMEC) and several RSUs equipped with MEC servers. Similar with previous study, suppose that each cellular network can fully cover the corresponding zone. Vehicles can communicate with the RSUs in the adjacent region through telecommunication, such as LTE or 5G. RSUs are connected to each and the BS with optical fiber, thus the corresponding transmission time can be ignored. For ease of description, we perform the system model in single cellular network.

Consider a set of vehicles, i.e. N=[1, 2, ..., N], and each vehicle has an application that requires to be completed. There are multiple edge servers connected to RSUs denoted by M=[1, 2, ..., M]. Considering that vehicles in VECNs have only one radio, although there exists multiple edge servers, only one of them can be adopted by vehicles during execution. For vehicles, various applications can be offloaded to edge servers or executed locally. If the required computing resources beyond what edge servers or the local unit can support, the request will be sent over the RSU to the CMEC.

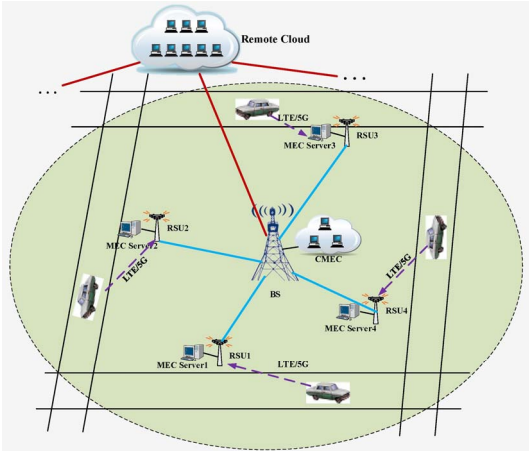

Fig. 1. The basic architecture of vehicular edge networks

Applications can be divided into many subtasks and illustrated as DAGs (denoted by $\mathbf{G}=(\boldsymbol{V},\boldsymbol{E})$). As depicted in Fig. 2, the edge (k, l) shows that the subtask l is based on the subtask k, and the dependency relationship between subtasks should not be reversed.

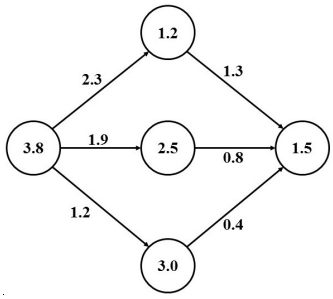

Fig. 2. Directed acyclic task graph

A. Basic Model

Consider a scenario with multiple servers and multiple vehicles. The j-th subtask of the i-th vehicle, i.e., $v_{i,j}$, may be expressed as $v_{i,j}=(d_{i,j}, c_{i,j}, \eta_{i,j})$, where $d_{i,j}$ means the size of input data, $c_{i,j}$ is the required CPU cycles, $\eta_{i,j}$ is the maximum execution time allowed. Let $a_{i,j}=\{0,-1\}\cup\boldsymbol{M}$ be the whole offloading of $v_{i,j}$. $a_{i,j}$=0 reveals the i-th vehicle i executes $v_{i,j}$ locally, and $a_{i,j}$=-1 or m means that $v_{i,j}$ is offloaded to the CMEC or the m-th server. Define the offloading decision vector as $\boldsymbol{A}_i=\{a_{i,1}, a_{i,2}, \ldots, a_{i,|vi|}\}$ for the i-th vehicle. We use $|V_i|$ to show the total number of subtasks. As defined above, $\boldsymbol{A}_*$ is utilized to show where the *-th vehicle process its subtasks. We can obtain the execution cost for local and edge computing.

1) Local Computing

While $a_{i,*}=0$, the task is computed locally. Denote the local CPU frequency of the i-th vehicle as f_i^l. Then the local execution time of $v_{i,j}$ can be expressed as:

$$t_{i,j}^{l}=c_{i,j}/f_i^{l}. \tag{1}$$

The corresponding energy consumption is:

$$e_{i,j}^{l}=\kappa(f_i^{l})^2 c_{i,j}, \tag{2}$$

where κ is the switched capacitance.

2) MEC computing

f_m^{MEC} is the computation capability of the m-th MEC server. Assume that, the first-come-first-served (FCFS) is utilized, the response delay should consider the delay in queue in the same server and the computation latency:

$$t_{i,j}^{exe}=c_{i,j}/f_m^{MEC}, \tag{3}$$

$$t_{i,j}^{MEC}=t_{i,j}^{exe}+t_{i,j}^{trans}+t_{i,j}^{queue}. \tag{4}$$

Here, $t_{i,j}^{exe}$ denotes the execution time in server. $t_{i,j}^{trans}$ represents the time for transmission:

$$t_{i,j}^{trans}=d_{i,j}/r_{i,m}^{up}, \tag{5}$$

where $r_{i,m}^{up}$ is the uplink data rate between the i-th vehicle and the m-th RSU. And it can be obtained by:

$$r_{i,m}^{up}=\frac{B}{n}log_2\left(1+\frac{p_{i,m}\cdot g_{i,m}^{up}}{\sigma^2}\right). \tag{6}$$

Here B denotes the channel bandwidth, $p_{i,k}$ means the upload power of the i-th vehicle, $g_{i,m}^{up}$ denotes channel gain. σ^2 represents the background noise power. Assume that the system with OFDMA is considered, therefore interval interference can be ignored. Meanwhile, the bandwidth is divided equally to the connected vehicles at the same time. The energy consumption of edge computing contains the data transmission energy and the maintaining energy in the idle state. Thus, we have

$$e_{i,j}^{MEC}=p_i^{up}t_{i,j}^{trans}+p_i^{static}t_{i,j}^{exe} \tag{7}$$

3) CMEC computing

$v_{i,j}$ can also be sent to CMEC for execution over RSU. In general, there are more abundant resources on CMEC server. Due to the high-speed fiber, the transmission delay between RSUs and CMEC can be ignored. Similar to (4) and (7), the corresponding time and energy cost can also be obtained. At each time slot, each vehicle generates a computation-intensive task with dependency-aware. Note that, vehicles can only choose one of edge servers to be offloaded. In order to alleviate network congestion, CMEC can be offloaded in case that deadlines of tasks beyond what edge servers or the local unit can support. The best offloading strategy is to minimize the entire cost in VECNs.

B. Problem Description

From the above description, we can formulate the offloading problem as minimizing latency and energy cost. As mentioned above, each vehicle can offload subtasks to the m-th $(1\le m\le M)$ MEC server or the CMEC, or process subtasks locally. Thus, for each vehicle, we have $(M+2)$ offloading decisions. According to the DAG-task offloading model, we aim to minimize the whole time and energy cost for all applications in vehicles subject to several constraint, i.e.,

$$\begin{aligned}&\min_{\mathbf{A}}\ \beta_t\sum_{i=1}^{N}t_{i,total}(i,|V_i|,\mathbf{A}_i)+\beta_e\sum_{i=1}^{N}e_{i,total}(i,|V_i|,\mathbf{A}_i),\\ &s.t.\ \ \mathrm{C1}: a_{i,j}\in\{-1,0,1,2,\ldots,M\},\\ &\qquad\mathrm{C2}: t_{i,j}\le\eta_{i,j},\ \forall i\in N, j\in\{1,2,\ldots,|V_i|\}\end{aligned}, \tag{8}$$

where β_t and β_e denote the weights of time and energy consumption, respectively. We have $\beta_t+\beta_e=1$. The problem above is NP-hard, i.e. achieving the optimal solution is time

consuming. In following parts, DRL is adopted to solve this problem. Furthermore, we will consider load balancing factor.

IV. DRL-BASED MULTI-VEHICLES OFFLOADING SCHEME WITH DEPENDENCY GUARANTEES

DRL can learn high dimensional states with the deep neural networks efficiency and maximize long-term expected benefits. In follow parts, a DRL-based offloading scheme is introduced to address the joint optimization for multi-vehicles scenarios. Firstly, the priority of DAG structure subtask scheduling for single vehicle is described.

A. Priority of Subtask Scheduling

As defined above, $\boldsymbol{V}$ is the set of subtasks. Let $y_{i,j}$ denote the latency of subtask $v_{i,j} \in \boldsymbol{V}_i$,:

$$y_{i,j} = \min\{t_{i,j}^{local}, t_{i,j}^{MEC}\}. \tag{9}$$

Thus, the corresponding priority $P(v_{i,j})$ can be recursively expressed by:

$$P(v_{i,j}) = \max_{v_{i,j'} \in succ(v_{i,j})} P(v_{i,j'}) + y_{i,j}, \tag{10}$$

where succ($v_{i,j}$) denotes successors of $v_{i,j}$. When $v_{i,j}$ is the exit subtask, we have

$$P(v_{i,j}) = y_{i,j}, v_{i,j} \in \mathbf{exit}(G). \tag{11}$$

The priority of subtask on the *i*-th vehicle can be obtained recursively. According to $P(v_{i,j})$, we can order the scheduling sequence and denote it as:

$$V_i = (v_{i,1}, v_{i,2}, ..., v_{i,|v_i|}). \tag{12}$$

B. A Distributed Offloading Scheme Based on Policy Gradient

In centralized control scenarios, to achieve best performance, a centralized network requires the information of all vehicles, such as the size of computation data for different subtasks. Unfortunately, to acquire those knowledge is impractical since users may be unwilling to reveal parameters due to privacy protection. In this part, a DRL-based offloading scheme is put forward. To be specific, we describe the whole process as a multi-agent Markov Decision Process (MDP) and introduce a **P**olicy **G**radient-based **D**istributed **C**omputation **O**ffloading algorithm (PGDCO). Each vehicle has a DRL-based offloading agent. For each agent, the core elements are illustrated as follows.

State ($\mathbf{S}_i^t$): Because of privacy protection, each agent only observe the past strategy and its own information. State is composed of three components, i.e., DAG-tasks structure information of the *i*-th vehicle (denoted by $\mathbf{S_1}$), basic information denoted by $\mathbf{S_2}$ (including load factor of each MEC server and its current the radio bandwidth), and the offloading decisions of other vehicles in the previous time slot (denoted by $\mathbf{S_3}$). We represent the information of subtasks as a set of tensors in the same way as in [24]. Assume that the *m*-th edge server has *k* subtasks in the task queue currently, then its load factor (*LF*) is defined as the ratio of the CPU cycles required in queue to its computation capacity:

$$LF_m^t = \sum_{i \in queue} c_i / f_m. \tag{13}$$

$\mathbf{S_2}$ can be expressed as:

$$\mathbf{S}_2(t) = [LF_1^t, LF_2^t, ..., LF_M^t, b_i^t]. \tag{14}$$

Also we can have the past strategy set of other vehicles $\mathbf{S_3}$ in the following way:

$$\mathbf{S}_3(t) = [\boldsymbol{x}_1^{t-1}, ..., \boldsymbol{x}_{i-1}^{t-1}, \boldsymbol{x}_{i+1}^{t-1}, ..., \boldsymbol{x}_N^{t-1}]^T. \tag{15}$$

Action ($\mathbf{A}_i^t$): Define action as an offloading decision for the vehicle:

$$\mathbf{A}_i^t = [a_{i,1}^t, a_{i,2}^t, ..., a_{i,|v_i|}^t] \tag{16}$$

Our model has constraints that each vehicle only select one of the servers as the destination server during the whole process.

Reward ($\mathbf{R}_i^t$): After taking actions, we can calculate its current reward. To maximize utilization of resource on servers, we hope to ensure the load on each server balanced. Therefore, reward function considers computation cost and load balancing factor. The load balancing factor of the system depicts the variance of the load of each server as:

$$\mathrm{var}(LF) = \sum_{m \in M} (LF_m - E(LF))^2 / M. \tag{17}$$

We can express the single step reward by the performance improvement with the current action:

$$R_i^t = [\mu(t_{local} - t(A_i)/t_{local}) + (1-\mu)(e_{local} - e(A_i)/e_{local})] - \delta\,\mathrm{var}(LF). \tag{18}$$

Here, μ and δ are the corresponding weights respectively. R_i^t can be obtained after the joint action $(\mathrm{A}_1^t, \mathrm{A}_{2,...,}^t \mathrm{A}_N^t)$ taken by different agents. System utility is the average of total rewards of different agents.

The proposed DRL-based multi-agent offloading scheme is illustrated in following part. For each agent, we have a critic network and an actor network with optimized network structure. The actor network gives actions according to state by a multiple neural network. The critic network obtains the corresponding value of the state. More precisely, to achieve learning efficiently, the multi-layers convolutional neural network and the transformer are employed to extract knowledge of $\mathbf{S_1}$ and $\mathbf{S_3}$, respectively. Transformer is a special neural network with encoder and decoder modules, which enable to learn the characteristics of past states. The state value can be estimated with the state history precisely with it.

With the policy gradient, the long term expectation rewards of agent *n* under different scenarios can be maximized. Policy gradient can be denoted as:

$$\begin{aligned}\nabla J(\theta_n) &= \mathbb{E}_{\pi_{\theta_n}, \rho(s_n)}[\nabla_{\theta_n} \log \pi_{\theta_n}(A^t|S^t)(Q^{\pi_{\theta_n}}(S^t, A^t) - V(S^t))] \\ &\approx \mathbb{E}_{\pi_{\theta_n}, \rho(s_n)}[(\frac{\pi_{\theta_n}(A^t|S^t)}{\pi_{\hat{\theta}_n}(A^t|S^t)}) \nabla_{\theta_n} \log \pi_{\theta_n}(A^t|S^t)(Q^{\pi_{\theta_n}}(S^t, A^t) - V(S^t))]\end{aligned}, \tag{19}$$

where θ_n is the actor network parameter. To accelerate the convergence, the proximal policy optimization (PPO) is employed. The training objective with policy π_θ is as follows:

$$\max_{\theta_n} J(\theta_n) = \max_{\theta_n} E_{\pi_\theta}[(Q^{\pi_{\theta_n}}(S^t, A^t) - V(S^t)) \cdot \log \pi_{\theta_n}(A^t|S^t)], \tag{20}$$

The actor network can be updating through mini-batch stochastic gradient ascent:

$$\theta_n(t+1) = \theta_n(t) + \alpha \cdot \nabla J(\theta_n). \tag{21}$$

For updating the critic network, we have the loss function:

$$L_n(\omega_n) = \mathbb{E}_{s_n \sim \rho(s_n)}[\mathbb{E}_{s_n', a_n}[r + V_{\omega_n}(s_n')] - V_{\omega_n}(s_n)]^2, \tag{22}$$

where s_n' represents the next state of s_n, $\omega_n$is the critic network parameter. We can update the critic network as:

$$\omega_n(t+1) = \omega_n(t) - \beta \cdot \nabla L_n(\omega_n). \tag{23}$$

α and β are actor learning rate and critic learning rate. The k-step learning is utilized in our scheme, and experiments validates that the parameters are updated towards superior results in terms of the long-term reward. The corresponding pseudocode is shown below.

Algorithm: Distributed offloading scheme based on policy gradient

Input: N, M, $\mathbf{G}_i$=($\mathbf{V}_i$, $\mathbf{E}_i$), f_i^l, f_m^{MEC}, f^{CMEC}, α, β, γ
Output: The offloading strategy of different vehicles.
Initialize: network parameters θ_n, ω_n, replay buffer D_n.
for each episode g = 1, 2,… **do**
  for each time slot t = 1, 2,… **do**
    for agent $n \in \mathcal{N}$ **do**
      Obtain the state s_t and take action with policy π_{θ_n}.
      Obtain the reward after actions taken by multiple agents.
      Put transitions $\{s_t, a_t, r_t, s_{t+1}, ..., s_{t+k}, a_{t+k}, r_{t+k}, s_{t+k+1}\}$ into D_n.
      Input s_{t+k+1} into the actor network and take the action.
      Obtain its reward by (18).
      if k % D_n == 0 **then**
        Randomly sample a mini-batch of data in D_n.
          Obtain (22) and update ω_n by (23).
          Obtain (20) and update θ_n by (21).
        Clear up buffer D_n.
      end if
    end for
  end for
end for

V. Numerical Results and Discussions

We consider multi-vehicles scenario that, each of vehicle has a DAG-task that needs to be accomplished at the beginning of the game. Different vehicles may experience automotive applications with different DAG structure. For each subtask, the input data size is drawn from [100 KB, 150 KB] randomly, and the maximum latency allowed is between 0.3 s and 1 s. The CPU cycles of subtasks ranges from 20 Megacycles to 30 Megacycles. We set the computation capacity of CMEC, MEC servers and vehicles to 20 GHz, 10 GHz and 1GHz, respectively. We set the background noise power to -100 dBm. For RSUs, the wireless bandwidth is set to 30 MHz. μ and δ in (18) are set to 0.6 and 0.05. Extensive numerical simulations are performed to evaluate the performance of the PGDCO. In different settings, we compare our scheme with some baseline methods, i.e. the centralized actor critic algorithm (AC), the DEFO [12] and random algorithm (Random). The DEFO is a distributed method which performs task scheduling in edge server and share prior knowledge of vehicles.

A. Convergence Analysis

Without loss of generality, the convergence of the PGDCO is first considered. In this simulation, there are 10 vehicles and 4 edge servers. Number of subtasks in each vehicle varies from 6 to 10. As shown in Fig. 4, system utility of the proposed algorithm converges to nearly 0.82. It can be observed that, PGDCO significantly exceeds other methods by achieving the best system utility. Specifically, as compared with PGDCO, AC decreases the system utility by about 0.07, and DEFO decreases the system utility by about 0.09. Random performs the worst and decreases the system utility by 0.5. Meanwhile, our scheme takes about 550 time slots to converge to the stable state with small shock, obtaining the fastest convergence speed. However, AC takes about 900 episodes and DQN takes about 1100 episodes for convergence. And both methods easily falls in the local optimum. The superiority of our scheme lies primarily in the optimal network structure and k-step learning.

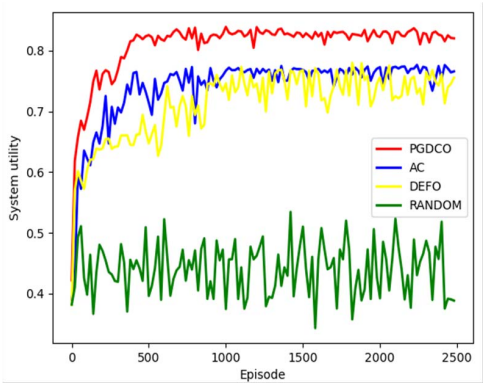

Fig. 3. System utility under different episode

B. Performance Under Different Numbers of Subtasks

We then explore the influence of different average subtasks of vehicles on performance. In this experiment, there are 10 vehicles and 4 edge servers. For easy analysis, assume that each vehicle has a same task graph with chain-based structure. We change the average number of subtasks on vehicles from 4 to 12. We compare the PGDCO with two representative benchmarks, i.e. AC and the DEFO, in terms of the time and energy cost under different average number of subtasks. Each point in the figure is the average performance of 100 independent realizations. We evaluate the cost after approaches have converged.

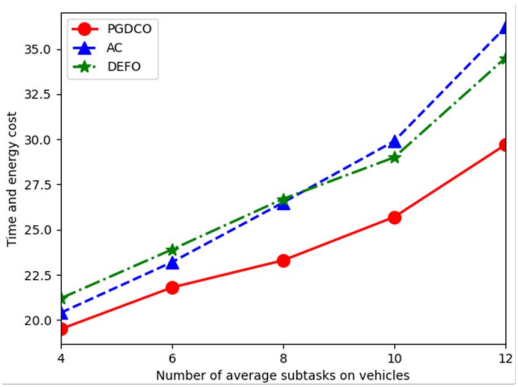

Fig. 4. Cost under different average numbers of subtasks

As illustrated in Fig. 5, PGDCO outperforms other methods as achieving the least time and energy cost in different average numbers of subtasks. For instance, compared with PGDCO, AC increases the cost by about 10 percent. Also, we can observe that, as the number of subtasks grows, total costs of three methods increase since there are much more subtasks competing for constrained resources. Furthermore, it can be found that, the time and energy cost gap between the proposed PGDCO and the existing algorithms gets larger gradually as the number of subtasks increases.

C. Performance Under Different Numbers of MEC Servers

Thirdly, we compare the time and energy cost with different numbers of MEC servers. The number of subtasks on each vehicle and the number of vehicles are set to 6 and 15, respectively.

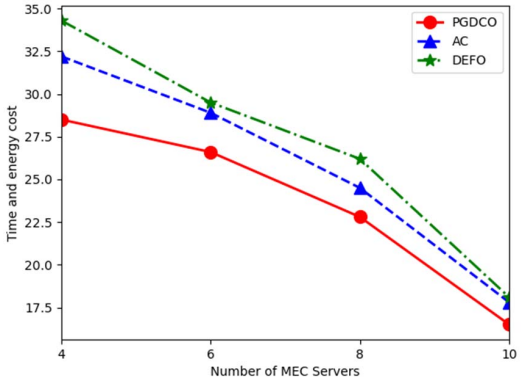

Fig. 5. Cost under different numbers of MEC servers

It can be observed from Fig. 6 that, the PGDCO can significantly reduce the overall cost compared with AC and DEFO in different numbers of MEC servers. Similarly, as the number of servers grows, the entire cost of the system decreases since more servers can be chosen without conflict.

VI. Conclusion

Considering the VECNs with multiple vehicles, this paper proposes a distributed offloading scheme based on policy gradient, i.e. PGDCO, to achieve the offloading decision with dependency guarantees. The goal is to minimizing the weighted sum of vehicles' time and energy cost and the load balancing factor. Priority of subtask scheduling is introduced firstly for organizing subtasks. Due to privacy protection and optimal resource allocation, the distributed DRL framework is proposed, of which the state contains abundant information. Besides, the framework employs optimal networks to learn and improve system utility from experiences, such as CNN and transformer, which makes training efficiency. Further, we utilize *k*-step learning to achieve superior estimation in term of long-term reward. Simulation results have demonstrated that the proposed scheme show superiority over the representative benchmarks.

Acknowledgment

This work was fully supported by the National Natural Science Foundation of China under Grant No. 61901015 and No. 91438116.

References

[1] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing inwireless systems: A survey, some research issues, andchallenges," IEEE Communications Surveys & Tutorials.vol. 20, no. 1, pp. 7-38, 2018.

[2] M. Muniswamaiah, T. Agerwala and C. C. Tappert, "Green computing for Internet of Things," 2020 7th IEEE International Conference on CSCloud/ 2020 6th IEEE International Conference on EdgeCom. IEEE, pp. 182-185, 2020.

[3] Z. Ning, J. Huang, and X. Wang, "Vehicular Fog Computing: Enabling Real-time Traffic Management for Smart Cities," IEEE Wireless Communications, vol. 26, no. 1, pp. 87-93, 2019.

[4] Y. Li, A. C. Orgerie, I. Rodero, B. L. Amersho, M. Parashar, and M. Menaud, "End-to-end energy models for edge cloud-based IoT platforms: Application to data stream analysis in loT," Future Generation Computer Systems, vol. 87, pp. 667-678, 2018.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Communications Surveys &Tutorials, vol. 19, no. 4, pp. 2322-2358, 2017.

[6] Z. Hu, J. Niu and T. Ren, "A Resource Management Model for Real-time Edge System of Multiple Robots," 2020 7th IEEE International Conference on CSCloud/2020 6th IEEE International Conference on EdgeCom. IEEE, pp. 222-227, 2020.

[7] L. Yang, H. Zhang, and M. Li, "Mobile Edge Computing Empowered Energy Efficient Task Offloading in 5G," IEEE Transactions on Vehicular Technology, 67(7), pp. 6398-640, 2018.

[8] X. Wang, et al., "Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4568-4578, 2018.

[9] J. Du, E. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp.1079-1092, 2019.

[10] X. Wang, et al., "Optimizing Content Dissemination for Real-time Traffic Management in Large-scale Internet of Vehicle Systems," IEEE Trans. on Vehicular Technology, vol. 68, no. 2, pp. 1093-1105, 2019.

[11] W. Ma, L. Mashayekhy, "Truthful Computation Offloading Mechanisms for Edge Computing," 2020 7th IEEE International Conference on CSCloud/2020 6th IEEE International Conference on EdgeCom. IEEE, pp. 199-206, 2020.

[12] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," IEEE Internet of Things Journal, vol. 7, no. 3, 2019.

[13] T. Alfakih, M. Hassan, and A. Gumaei, "Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA," IEEE Access, 8, pp. 54074-54084, 2020.

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

[15] Q. Qi, J. Wang, Z. Ma, et al, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," IEEE Trans on Vehicular Technology, vol. 68, no. 5, 2019.

[16] J. Y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," 2019, arXiv: 1901.10023.

[17] W. Li, X. You, Y. Jiang, J. Yang, and L. Hu, "Opportunistic computing offloading in edge clouds," Journal of Parallel and Distributed Computing, vol. 123, pp. 69-76, 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] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for loT applications," Future Gener. Comput. Syst., vol. 90, pp. 149-157, 2019.

[20] W. Jiang, G. Feng, S. Qin, et al, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," IEEE Trans. Wireless Commun, vol. 18, no. 3, pp. 1610-1622, Mar 2019.

[21] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," IEEE Trans. Veh. Technol., vol. 68, no. 1, pp. 856-868, Jan. 2019.

[22] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretio machine learning approach," IEEE Trans. Veh. Technol, vol. 67, no. 1, 2018.

[23] G. Barth-Maron et al., "Distributed distributional deterministic policy gradients," 2018, arXio: 1804.08617.

[24] J. Wang J , J. Hu, G. Min, et al, "Computation Offloading in Multi-Access Edge Computing Using a Deep Sequential Model Based on Reinforcement Learning," IEEE Communications Magazine, 2019, 57(5):64-69.