Energy-Efficient Multi-UAV-Enabled Computation Offloading for Industrial Internet of Things via Deep Reinforcement Learning

Shuo Shi[1,2(✉)], Meng Wang[1], and Xuemai Gu[1,3]

[1] School of Electronic and Information Engineering, Harbin Institute of Technology, Harbin 150001, Heilongjiang, China
crcss@hit.edu.cn
[2] Network Communication Research Centre, Peng Cheng Laboratory, Shenzhen 518052, Guangdong, China
[3] International Innovation Institute of HIT in Huizhou, Huizhou 516000, Guangdong, China

Abstract. Industrial Internet of things (IIoT) has been envisioned as a key technology for Industry 4.0. However, the battery capcity and processing ability of IIoT devices are limited which imposes great challenges when handling tasks with high quality of service (QoS) requirements. Toward this end, in this paper we first use multiple unmanned aerial vehicles (UAVs) equipped with computation resources to offer computation offloading opportunities for IIoT devices due to their high flexibility. Then we formulate the multi-UAV-enabled computation offloading problem as a mixed integer non-linear programming (MINLP) problem and prove its NP-hardness. Furthermore, to obtain the energy-efficient solutions for IIoT devices, we propose an intelligent algorithm called multi-agent deep Q-learning with stochastic prioritized replay (MDSPR). Simulation results show that the proposed MDSPR converges fast and outperforms the normal deep Q-learning (DQN) method and other benchmark algorithms in terms of energy-efficiency and tasks' successful rate.

Keywords: UAV-enabled IIoT · Deep reinforcement learning · Computation offloading

1 Introduction

Industrial Internet of things (IIoT), as an emergying communication paradigm, is expected to evolutionize manufacturing and also drive growth in productivity across various types of applications such as smart factories and smart grids [1]. It is important to provide the required quality of service (QoS) in terms of reliability and real-time to IIoT applications [2]. However, the battery capacity and processing ability of IIoT devices are limited, which imposes great challenges when handling tasks with characteristics of computation-sensitive and latency-sensitive.

S. Shi et al. (Eds.): AICON 2020, LNICST 356, pp. 295–305, 2021.
https://doi.org/10.1007/978-3-030-69066-3_26

Recently, with the rapid developement of mobile edge computing in IIoT, local devices can choose to offload the latency-sensitive tasks to edge servers through wireless access which can effectively alleviate the computation stress in local [3]. Due to the high flexibility and controllbility, unmanned aerial vehicles (UAVs) equipped with computation capabilities can act the role of edge servers to enable edge computing even in the absence of wireless infrastructures [4]. Furthermore, the communication links between devices and UAVs are largely line of sight (LoS) links [5], which will improve the reliability in both uplink and downlink transmission. The studies of energy-efficient computation offloading in IIoT with UAVs assisted are still remain as an open issue. In [4], a single UAV servers as a moving cloudlet for mobile users, aiming to minimize the total energy consumptions by jointly optimizing the bits allocation for uplink and downlink communications. [5] considers a stochastic task arrival model and obtains an energy-efficient offloading solution through Lyapunov optimization [6]. Considers a multi-UAV scenario and solving the non-convex computation offloading problem with iterative optimization algorithm. [7] studies the computation rate maximization problem under both partial and binary computation offloading modes subject to the energy constraints.

However, the solutions to the complex computation offloading problems in the aforementioned works [5–7] are based on one-shot optimization and fail to maximize the long-term performance of computation offloading. Moreover, these solutions are obtained under a specific model and require a priori knowledge of the network model, if the channel fading is fast, they cannot meet the requirement of real-time decision. Furthermore, the computation capabilities in devices are limited in IIoT, algorithms with high time complexity cannot run efficiently in the processor of devices. To obtain long-term rewards and real-time offloading decisions, deep reinforcement learning (DRL), as an emerging artificial intelligence technique, has been introduced to complex computation offloading problems with low complexity [8,9] proposes a deep dynamic scheduling algorithm to solve the offloading decisions problem with minimum energy costs using deep Q-learning (DQN). [10] considers a computation offloading problem with stochastic vehicle traffic, dynamic computation requests and time-varying communication conditions, and uses Q-learning as offline solution, DQN as online solution.

Motivated by the aforementioned issues, in this paper, we focus on the energy-efficient binary computadtion offloading problem in IIoT with UAV assited. IIoT devices can choose to execute the tasks locally or offload the tasks to UAVs through LoS links or offload the tasks to the remote cloud center through base stations. Compared with [4,5,7] which focus only on a single UAV, we use multiple moving UAVs as edge servers to provide computation offload opportunities, which is more practical due to the limited battery capacity of UAVs. We first propose the multi-UAV-enabled network model and formulate the computation offloading problem as a mixed integer non-linear programming (MINLP) problem. Branch and bound (BB) method can be used to solve the problem [8] but with prohibitively high computational complexity. [9] proposes an iterative optimization method to solve the MINLP problem, however, the complexity grows rapidly as the size of the network increases. To address this, we propose our multi-agent deep Q-learning with stochastic prioritized replay (MDSPR) to solve the

problem distributedly and efficiently. Compared with [9,10] using normal DQN and DDQN, our method choose training experiences with high priorities instead of choosing randomly, which can accelerate the training process and improve convergence stability.

2 System Model and Problem Formulation

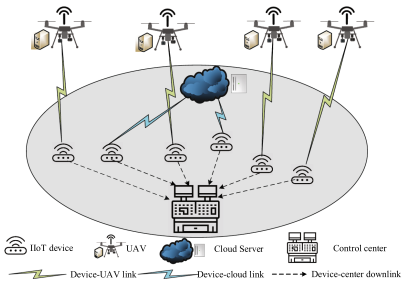

Fig. 1. Multi UAV-Enabled IIoT devices computation offloading network model.

We consider a three-layer UAV-Enabled IIoT computation offloading network model where IIoT devices process the tasks assigned from IIoT control centers, each IIoT device can choose to execute the tasks locally or offload the tasks to the UAVs or offload the tasks to the remote cloud center through terristrial base stations, and send the execution results back to control centers, as shown in Fig. 1. UAVs with communication and computation abilities act the role of MEC servers. $\mathcal{M} = \{1, 2, ..., M\}$, $\mathcal{N} = \{1, 2, ..., N\}$ denote the devices' set and UAVs' set, respectively. IIoT devices are distributed in an area with a radius of K and UAVs hover above the devices with a height of H. The locations of devices and UAVs during a single time interval $t \in \{1, 2, ..., T\}$ can be expressed as $\mathbf{q}_i(t) = (x_i(t), y_i(t)), i \in \mathcal{M}$ and $\mathbf{p}_j(t) = (X_j(t), Y_j(t), H), j \in \mathcal{N}$, respectively. Moreover, tasks can be modelled as $W_i(t) = (s_i(t), c_i(t), d_i(t))$ according to [3], where $s_i(t)$ denotes the total bits of the task, $c_i(t)$ denotes the total CPU circles required by the task and $d_i(t)$ denotes the tolerant delay of the task. In this paper, we assume UAVs flying with a random speed v in the fixed area, $\mathbf{q}_i(t)$, $\mathbf{p}_j(t)$ remain unchanged in a single time slot and there is a coming task for each IIoT device in a single time interval.

2.1 Communication and Computation Model

According to [8] the line of sight (LoS) channel gain between i-th IIoT device and j-th UAV can be modelled as

$$h_i^j(t) = \beta_0 d^{-2}(t) = \frac{\beta_0}{H^2 + ||\mathbf{q}_i(t) - \mathbf{p}_j(t)||^2}, \tag{1}$$

where β_0 denotes the channel gain at the reference distance $d_0 = 1\,\mathrm{m}$. Tasks can be offloaded to UAVs or the cloud center through wireless channle. Each IIoT device is associated with UAVs using orthogonal frequency-division multiplexing (OFDM). Tasks can be successfully offloaded to UAVs if the distance d between UAVs and devices is less than R, where R denotes the communication range of UAVs.

Since the size of the processed tasks can be negligible compared with the original tasks, in this paper, we only consider the uplink transmission. The uplink transmission rate between i-th IIoT device and j-th UAV can be expressed as

$$r_i^j(t) = B\log_2\Big(1 + \frac{Ph_i^j(t)}{\sigma^2}\Big), \tag{2}$$

where B is the allocated bandwith for IIoT devices, P is the uplink transmission power of IIoT devices, σ^2 denotes the background noise power, and we assume that it is constant on the slow fading channel [7]. The uplink transmission rate of an IIoT device i that chooses to offload the task to the cloud through wireless link can be denoted as

$$r_i^c(t) = \omega\log_2\Big(1 + \frac{P_c H_i(t)}{\sigma^2}\Big), \tag{3}$$

where w, P_c, $H_i(t)$ denote the bandwith, transmission power , the channel gain between i-th IIoT device and cloud, respectively. For UAV computation offloading, the task transmission time can be expressed as

$$L_i^j(t) = \frac{s_i(t)}{r_i^j(t)}, \tag{4}$$

for cloud computation offloading, according to [7] the transmission time can be denoted as

$$L_i^c(t) = \frac{s_i(t)}{r_i^c(t)} + \tau, \tag{5}$$

where τ denotes the uplink propogation delay factor due to congestion of the bachhaul link. To simplify the analysis, all the UAVs are assumed to have the same computation capacity, denoted as C_u. The computatioin capacity of IIoT devices and the remote cloud center are denoted as C_l and C_c, respectively. Thus the local task computation time can be expressed as

$$L_l(t) = \frac{c_i(t)}{C_l}, \tag{6}$$

according to [7], the energy consumption of local computation can be given as

$$E_i^l(t) = \theta c_i(t)^2, \tag{7}$$

where θ is the factor denoting the consuming energy per CPU cycle. The task computation time in UAV can be denoted as

$$D_u(t) = \frac{c_i(t)}{C_u}. \tag{8}$$

Note that,the cloud processing time can be negligible since the computation capacity is enormous in cloud center. Thus, for UAV offloading cases, the total communication energy can be expressed as

$$E_i^j(t) = P(L_i^j(t) + D_u(t)), \tag{9}$$

for cloud offloading cases, the total energy consumptions can be modelled as

$$E_i^c(t) = P_c(L_i^c(t) + \tau), \tag{10}$$

2.2 Problem Formulation

In each time interval IIoT devices should make offloading decisions, we define $\mathbf{I} = (I_1(t), I_2(t), ..., I_M(t))$ as the offloading indicated vector in a single time interval, where $I_i(t) \in \left\{I_i^{-1}, I_i^0, I_i^1, ..., I_i^M\right\}, \forall t \in T, \forall i \in \mathcal{M}$ is an indicator which can be used to describe an offloading decision, $I_i^k \in \{0, 1\}, k \in \{-1, 0, 1, 2, ..., N\}$. Let $E(t) \in \left\{E_i^c(t), E_i^j(t), E_i^l(t)\right\}$, $L(t) \in \left\{L_i^c(t), L_i^j(t), L_i^l(t)\right\}$. The energy consumption of a singe task can then be derived as

$$E_i(t) = I_i(t)E(t) = \begin{cases} E_i^c(t), & k = -1 \\ E_i^l(t), & k = 0 \\ E_i^j(t), & k \in \mathcal{M} \end{cases} \tag{11}$$

Our objective is to minimize the total energy cosumptions of all IIoT devices in a time period T through optimizing the offloading indicated vector while satisfying the basic constraints. Thus, we can formulate the energy-efficient multi-agent computation offloading problem as follows

$$\begin{aligned} \min_{\boldsymbol{I},\boldsymbol{p},\boldsymbol{q}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{M} I_i(t)E_i(t). \\ \text{s.t.} \quad & C1: I_i^k \in \{0, 1\}, \forall i \in \mathcal{M}, k \in \{-1, 0, 1, 2, ..., N\} \\ & C2: \sum_{k=1}^{N} I_i^k \leq 1, \forall i \in \mathcal{M}, k \in \{-1, 0, 1, 2, ..., N\} \\ & C3: \sum_{i=1}^{M} I_i(t) \leq W, \quad I_i(t) \in \left\{I_i^1, ..., I_i^M\right\} \\ & C4: I_i(t)L(t) \leq d_i(t), \forall i \in \mathcal{M}, \forall t \in \mathcal{T} \\ & C5: H^2 + ||\mathbf{q}_i(t) - \mathbf{p}_j(t)||^2 \leq R^2, \forall i \in \mathcal{M}, \forall t \in \mathcal{T} \end{aligned} \tag{12}$$

C1 indicates that I_i^k is a binary variable. C2 shows that if a task is choosen to offload to UAVs, it can only be offloaded to a single UAV. C3 gives the constraints that tasks need to be executed in UAVs cannot exceed the maximum limit. C4 guarantees that the task can be finished within the tolerant delay. C5 means if i-th IIoT devices offload the task to j-th UAV, the distance between the device and UAV should less than the communication range of the UAV.

3 Proposed Intelligent Computation Offloading Solution : MDSPR

In this section, we present our solutions using multi-agent deep Q-learning with stochastic prioritized replay (MDSPR) for the energy-efficent multi-agent computation offloading problem.

3.1 Markov Decision Process Modeling

We first model the problem as a markov decision process (MDP). A MDP involves a agent that repeatedly observes the current state s_t during a time period. Agents interact with the environment through making actions a among all the available actions in that state, and the environment gives a feedback reward r_t to agents. Then, the agent will transfer to a new state s_{t+1} with a transition probability $P(s_{t+1}|s_t, a)$. A typical MDP tuple can be modeled as $< \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{S}' >$.

State Space. A state vector should reflect the agents' perception of the environment. According to our network model, we define $s_t = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$ as the state vector. $\mathcal{S}_1 = \{s_i(t), c_i(t), d_i(t)\}$ reflects the task properties in each time interval. $\mathcal{S}_2 = \{x_i(t), y_i(t)\}$, $\mathcal{S}_3 = \{X_1(t), Y_1(t), X_2(t), Y_2(t), ..., X_N(t), Y_N(t)\}$ denotes the current locations of the IIoT device and UAVs.

$\mathcal{S}_4 = \left\{H_i(t), h_i^1(t), h_i^2(t), ..., h_i^N(t)\right\}$ gives the channel conditions between i-agent and different offloading objectives. The dimension of a state vector is $3N+6$.

Action Space. $\mathcal{A} = \{-1, 0, 1, 2, 3, ..., N\}$ represents the action space. Here $a = -1$ means to offload the task to cloud center, $a = 0$ means to execute the task locally, $a = \{1, 2, 3, ..., N\}$ denotes that the agent chooses to offload the task to UAVs.

Transition Probability. $\mathcal{F}$ denotes the probability distribution $P(s_{t+1}|s_t, \{a\}_{a\in\mathcal{A}}) \in [0, 1]$ when state transition happens. Note that, in our network model, the agent cannot predict the state and reward of next state before making actions, so the MDP problem needs to be solved by model-free reinforcement learning method, in which the transition probabiliy distribution remains unknowm.

Reward Function. The reward function is a immediate feedback for agents' actions from the environment. In our optimization problem, the objective is to minimize the total energy consumptions of agents, so the reward function should be inverse proportional to energy consumptions, which can be defined as

$$r(t) = \begin{cases} \dfrac{1}{I_i(t)E(t)}, & I_i(t)E(t) \le d_i(t) \\ \dfrac{1}{I_i(t)E(t)} - m, & I_i(t)E(t) > d_i(t) \end{cases} \tag{13}$$

where m is a punishment factor when the delay constraint is not satisfied.

3.2 Deep Q-Learning Structure

In this paper, we use deep Q-learning structure to solve the MDP. Deep Q-learing (DQN) is a promising reinforcement learning technique which combines deep learning with Q-learing aiming to solve complex problems in communication and networking [9]. Compared with Q-learning, DQN uses deep neural networks to replace the traditional Q-table, so that DQN can overcome the curse of dimensionality in Q-learning. The key idea of DQN is to use neural network as a function approximator. Given input state s_t, the output of neural network is $Q(s, a_i; \mathbf{w})$, where $\mathbf{w}$ is the weights of DQN.

Experience replay is an important technique to improve training efficiency in DQN. Since there is no Q-table in DQN, in a series of actions, the training samples may have strong correlations which will make learning process fall into local optimum. To solve this, DQN introduces a replay memory $\mathcal{D}$, once the agent obtain a new experience $< s_t, a_t, r_t, s_{t+1} >$, the experience is stored in $\mathcal{D}$. Each training step a mini-batch with size L is randomly sampled from D instead of a one step experience.

In the training stage, despite the current network $Q(s, a_i; \mathbf{w})$, DQN introduces another target network $Q(s, a_i; \mathbf{w'})$ to produce target value. The weights of $Q(s, a_i; \mathbf{w'})$ are updated every Z training steps. The target value is defined as

$$y_t = r_t + \gamma \max_{\boldsymbol{a'} \in \boldsymbol{\mathcal{A}}} Q(s', a'; \mathbf{w'}), \tag{14}$$

where γ denotes the reward decay in DQN. However, recent research has proved that the greedy iteration method may cause overestimation. Double DQN (DDQN) is proposed to eliminate overestimation which can be expressed as

$$y_t = r_t + \gamma Q(s', \underset{\boldsymbol{a}}{\operatorname{argmax}}\, Q(s, a_i; \mathbf{w}); \mathbf{w'}), \tag{15}$$

DDQN eliminates the problem of overestimation by decoupling the selection of target Q action and the calculation of target Q into two neural networks. Temporal difference error (TD-error) is defined as the difference between target value and current value which can be denoted as

$$\delta_t^j = |y_t - Q(s_t, a_1, a_2, ..., a_N)|, \forall t \in \mathcal{T}, \forall j \in \mathcal{N} \tag{16}$$

The loss function is defined as the least squared loss of TD-error denoted as

$$\mathbb{L}(\mathbf{w}) = \mathbb{E}[(y_{target} - Q(s, a; \mathbf{w}))^2]. \tag{17}$$

$\mathbb{L}(\mathbf{w})$ can be minimized through back propogation methods in deep learning, in this paper we use the classic gradient descent method

$$\mathbf{w} = \mathbf{w} - \alpha \bigtriangledown_{\mathbf{w}} \mathbb{L}(\mathbf{w}), \tag{18}$$

where α is the learning rate.

3.3 Stochastic Prioritized Replay

Traditional experienced replay method selects L experiences randomly from D, however, the importance of experiences in replay memory are different. Experiences with larger TD-errors may have better training effect. Furthermore, simply choosing L experiences with the largest TD-errors (greedy prioritized) in each training step will cause high time complexity since the size of $\mathcal{D}$ is large. Therefore, according to [2], in this paper we use deep Q-learning with stochastic prioritised replay. We adopt the binary tree structure to store and sample experiences with priorities to accelerate the training process with relative low time complexity. The top node stores the sum of all priorities in $\mathcal{D}$, in sampling stage, the sum is divided equally into L intervals. In sampling stage, each interval generates a random count in its range, and then starts to search downwards with the top node as the parent node. If the input value is less than the left child node, the left child node is regarded as the parent node to continue the downward search, otherwise, the input value is subtracted by the value of left child node, then, the right child node is denoted as the parent node, and the difference is used as an input to continue searching downwards until the last layer, and the experience corresponding to the parent node is the output sample. The chosen probability distribution of s_k can be denoted as

$$P(k) = \frac{p_k}{\sum_{d=1}^{O} p_d}. \tag{19}$$

where O denotes the capacity of $\mathcal{D}$. (19) shows that the higher the priority, the greater the probability of being selected. Moreover, the iterations of stochastic prioritized replay during each sampling process is $L * log_2(O)$, compared with greedy prioritized $L * O$.

4 Simulation Results

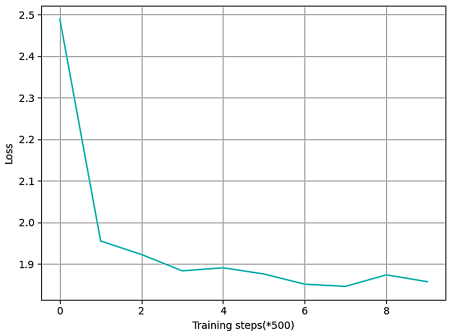

Fig. 2. The convergence performance of the proposed MDSPR.

First, we show the convergence performance of the MDSPR. Figure 2 shows the changes of loss over training steps, it is observed that the loss reduce rapidly at the beginning. This is because at the beginning of the training, agents take actions randomly and the approximation is not precise. After 600 steps the loss become stable, which indicates the convergence of MDSPR. We then compare the performance of our MDSPR with three benchmark algorithms, namely random algorithm, DQN algorithm and DDQN algorithm. Figure 3 shows the average energy consumptions of a mobile user when dealing with a single task. Since the tasks are generated with random s_i, c_i, d_i, an inappropriate offloading decision may cause huge additional energy comsumptions. For example, if a task with high computation requirement and small size is executed in local, the cost will increase significantly compared with that the task offloaded to the remote cloud center. In the first n training steps, the agent makes offloading decisions according to ϵ-greedy policy, after the learning stage, the average energy consumptions become stable due to the convergence of neural networks. Figure 4 shows the total energy consumptions under different number of tasks after the learning stage. From Fig. 4, it can be seen that the proposed MDSPR outperforms the three benchmark algorithms. The reason is that in each training step, MDSPR chooses more valuable experiences to learn, thus maximizing the overall long-time reward. Furthermore, it is observed that with the increase of tasks, the energy consumptions increase approximately linearly. This is because agents make offloading decisions upon each task arrival according to the maximum Q-value. After the convergence of the neural network, the action with highest Q-value has a high probability to obtain minimize energy consumptions.

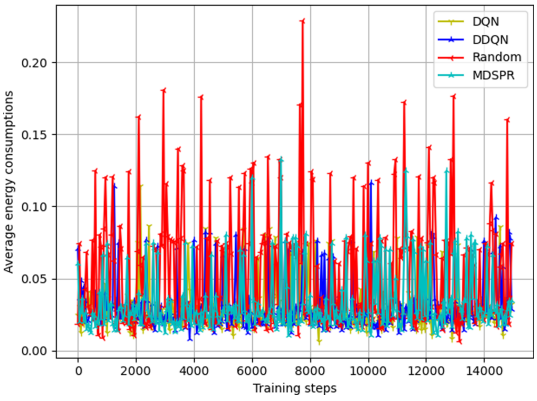

Fig. 3. The average energy consumptions of a mobile user when dealing with a single task.

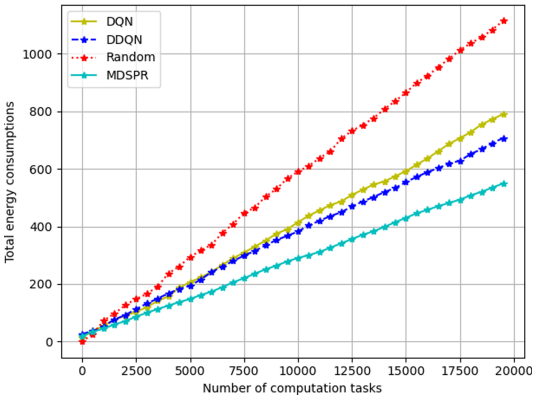

Fig. 4. The average energy consumptions of a mobile user when dealing with a single task.

5 Conclusion

In this paper, we study the energy-efficient computation offloading problem in UAV-enbaled IIoT network. To release the computation burden of IIoT devices, we use UAVs as edge servers to provide computation offloading opportunities and formulate the multi-UAV-enabled computation offloading problem as an MINLP problem and prove its NP-hardness. To obtain energy-efficent decisions, we propose our MDSPR solution. Simulation results show that the proposed MDSPR converges fast and outperforms the normal DQN method and other benchmark algorithms in terms of energy-efficiency and tasks' successful rate. Partial computation offloading and resource allocation issues in IIoT will be left as our future works.

References

1. Ojo, M.O., Giordano, S., Adami, D., Pagano, M.: Throughput maximizing and fair scheduling algorithms in industrial internet of things networks. IEEE Trans. Industr. Inf. **15**(6), 3400–3410 (2019)
2. Willig, A.: Recent and emerging topics in wireless industrial communications: a selection. IEEE Trans. Industr. Inf. **4**(2), 102–124 (2008)
3. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A Survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutorials **19**(4), 2322–2358 (2017)
4. Jeong, S., Simeone, O., Kang, J.: Mobile edge computing via a UAV-mounted cloudlet: optimization of bit allocation and path planning. IEEE Trans. Veh. Technol. **67**(3), 2049–2063 (2018)
5. Zhang, J., et al.: Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing. IEEE Internet Things J. **6**(2), 3688–3699 (2019)

6. Zhang, J., et al.: Computation-efficient offloading and trajectory scheduling for Multi-UAV assisted mobile edge computing. IEEE Trans. Veh. Technol. **69**(2), 2114–2125 (2020)
7. Zhou, F., Wu, Y., Hu, R.Q., Qian, Y.: Computation rate maximization in UAV-Enabled wireless-powered mobile-edge computing systems. IEEE J. Sel. Areas Commun. **36**(9), 1927–1941 (2018)
8. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
9. Wang, Y., Wang, K., Huang, H., Miyazaki, T., Guo, S.: Traffic and computation Co-offloading with reinforcement learning in fog computing for industrial applications. IEEE Trans. Industr. Inf. **15**(2), 976–986 (2019)
10. Liu, Y., Yu, H., Xie, S., Zhang, Y.: Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. IEEE Trans. Veh. Technol. **68**(11), 11158–11168 (2019)