Backscatter-Assisted Computation Offloading for Energy Harvesting IoT Devices via Policy-based Deep Reinforcement Learning

Yutong Xie*†, Zhengzhuo Xu‡, Yuxing Zhong‡, Jing Xu‡, Shimin Gong§, and Yi Wang¶‖
*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
†University of Chinese Academy of Sciences
‡School of Electronic Information and Communications, Huazhong University of Science and Technology, China
§School of Intelligent Systems Engineering, Sun Yat-sen University, China
¶SUSTech Institute of Future Networks, Southern University of Science and Technology, Shenzhen, China
‖Pengcheng Laboratory, Shenzhen, China

***Abstract*—Wireless Internet of Things (IoT) devices can be deployed for data acquisition and decision making, e.g., the wearable sensors used for healthcare monitoring. Due to limited computation capability, the low-power IoT devices can optionally offload power-consuming computation to a nearby computing server. To balance power consumption in data offloading and computation, we propose a novel hybrid data offloading scheme that allows each device to offload data via either the conventional RF communications or low-power backscatter communications. Such a flexibility makes it more complicated to optimize the offloading strategy with uncertain workload and energy supply at each device. As such, we propose the deep reinforcement learning (DRL) to learn the optimal offloading policy from past experience. In particular, we rely on the policy-based DRL approach for continuous control problems in the actor-critic framework. By interacting with the network environment, we can optimize each user's energy harvesting time and the workload allocation among different offloading schemes. The numerical results show that the proposed DRL approach can achieve much higher reward and learning speed compared to the conventional deep Q-network method.**

***Index Terms*—Deep reinforcement learning, DDPG, DQN, mobile data offloading, wireless backscatter.**

I. Introduction

Mobile edge computing (MEC) has emerged as a promising technique by providing the IoT devices with cloud-like computation capability at the easy-to-access and resource-rich MEC servers [1]. The edge devices are allowed to offload sensed data and computation workload (e.g., compressing, encryption, and outlier detection) to the MEC servers. Then the MEC servers return the processed data or results for fulfilling the service requests at the edge devices. On one hand, MEC offloading can support computation-intensive workload at the low-power edge devices. It reduces the edge devices' power consumption on computation. However, on the other hand, data offloading is inherently power-consuming by using the conventional RF communications. The power consumption of RF radios is typically high due to the emission of RF carrier signals [2]. Hence, data offloading based on RF communications may not be affordable by low-power IoT devices and hence prevents them from using the MEC servers. As both offloading and computation are power consuming, the user devices have to balance its power consumptions depending on the channel conditions, energy status, and the workloads.

Recently, wireless backscatter is proposed as novel communication technology with extremely low power consumption. The backscatter radios operate in *passive* mode by modulating and reflecting the incident RF signal [3]. Without using active components, the passive radios are featured with low power consumption and low data rate [4]. Whereas the active radios can transmit in a higher data rate by adapting the transmit power against the channel fading. This motivates us to design a hybrid MEC offloading scheme that allows each IoT device to switch the offloading scheme between the passive and active modes, e.g., [5]. The critical design problem is to optimize the energy harvesting time for each IoT device, as well as the transmission scheduling and workload allocation strategies among local computing, active, and passive offloading actions.

However, the joint optimization of offloading strategy becomes very challenging due to close couplings among different users and the uncertain network information, e.g., the time-varying channel conditions and random arrival of workload. To deal with such complexities, the model-free DRL-based framework has been proposed as a promising solution to learn the optimal offloading strategy based on past experience [6]. The authors in [7] considered the simple offloading scheme, in which the end user can access either the cellular network or WLAN with different costs. To minimize the user's cost, the deep Q-network (DQN) method is introduced to optimize the network selection based on the user's location and remaining data size. Considering limited capacity at the MEC server, DQN is also used in [8] to learn the optimal resource allocation and the offloading decision, based on the cost of all users and the available capacity of the MEC server.

However, the above-mentioned works for MEC offloading

The authors would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by an NSFC project grant (ref. no. 61872420), and the project of "PCL Future Regional Network Facilities for Large-scale Experiments and Applications (ref. no. PCL2018KP001)" Corresponding author: Shimin Gong.

generally reformulate a discrete control problem by properly quantizing the decision variables into a finite action space. This simplifies the design of learning policy by the value-based DQN framework and its variants, e.g., DDQN and dueling structure [6]. In this paper, we propose the policy-based deep deterministic policy gradient (DDPG) approach that directly searches for the optimal time and workload allocation strategies in continuous domain. Based on the channel conditions and energy status, each user device firstly optimizes its energy harvesting time to accumulate RF power. Given the power budget, the computation workload will be optimally divided among local computing, active, and passive offloading schemes to minimize the workload outage. Our numerical results demonstrate that the proposed DDPG approach for continuous control can achieve much higher reward and learning speed compared to the value-based DQN algorithm.

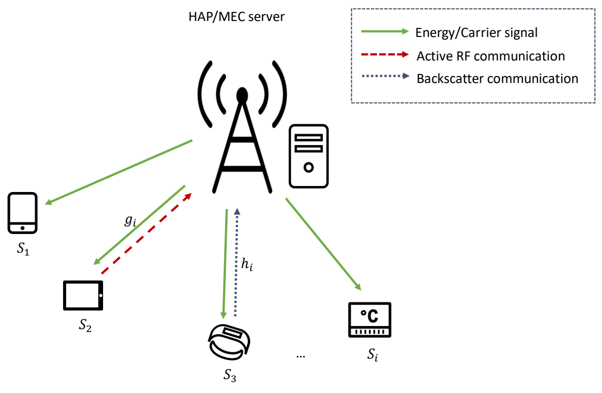
II. System Model

We consider a wireless sensor network with one hybrid access point (HAP) and N user devices that can sense and process data independently. The user devices can be envisioned as wearable devices for healthcare monitoring. This information can be sampled at different rate to save energy and maintain a certain accuracy requirement. The sensed information can be analyzed locally or remotely by machine learning algorithms for classification, prediction, and decision making, which are usually computation-intensive. To assist their data processing, the user devices can offload their sensed data and workload to an nearby MEC server, co-located with the HAP. The MEC server will return the processed data to the user devices after the completion of workload. The system model is illustrated in Fig. 1 and a similar model has been studied in [9].

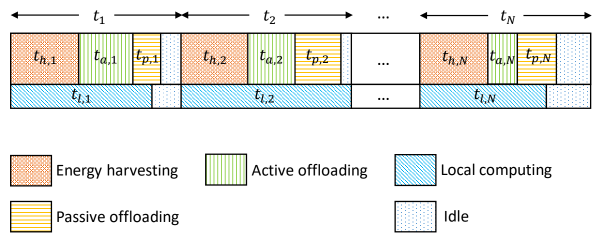
Let $\mathcal{N} = \{1, 2, ..., N\}$ denote the set of all edge nodes and S_i denote the i-th edge node for $i \in \mathcal{N}$. Each node is equipped with single antenna capable of harvesting energy from the HAP with constant transmit power. The complex uplink and downlink channels between HAP and node S_i are denoted by $h_i \in \mathcal{C}$ and $g_i \in \mathcal{C}$, respectively. Each S_i is allocated a time slot t_i for its data offloading and capable of energy harvesting in the same time slot. The workload of each edge node S_i is given by L_i, which is defined as the number of data bits to be processed either locally or remotely at the MEC server. We assume that the workload of each device is generated at the beginning of each time slot, and it has to be processed before the end of data frame.

A. Hybrid MEC Offloading

The data offloading from each user to the MEC server can be performed in either passive backscatter communications or active RF communications. In passive mode, the HAP's beamforming provides the carrier signal for the edge node to perform backscatter communications. A part of the incident RF signal is reflected back and the other part is still captured by the antenna and converted to energy. We assume that each user has only one antenna and thus it can only transmit in one radio mode or harvest energy from the HAP. The switch between passive and active mode can be achieved by tuning the load



(a) Wireless power hybrid data offloading in MEC



(b) Time allocation for MEC

Fig. 1: Hybrid MEC offloading for energy harvesting IoT devices.

impedance, e.g., [10]. As such, we further divide each time slot t_j into three sub-slots, as shown in Fig. 1(b). The first sub-slot $t_{h,j}$ is used for the user device to harvest RF power and then sustain local computing and data offloading. The following two sub-slots $t_{a,j}$ and $t_{p,j}$ are used for data offloading in active and passive modes, respectively. Besides data offloading, the user device can also perform local computation simultaneously with the data offloading [1], as shown in Fig. 1(b).

B. Workload Allocation

The workload generated in each time slot can be allocated among local computation, active and passive offloading. Note that different computation schemes have different processing capabilities and power consumption. Hence, the design of MEC offloading scheme aims to optimally divide the workload into three schemes, according to the dynamics in workload, channel conditions, and the energy supply of each edge device.

1) Active Offloading Scheme: Let $p_{a,i}$ denote the transmit power of user S_i in active offloading. The received signal at HAP is $y = \sqrt{p_{a,i}} h_i s(t) + \nu_d$, where $s(t)$ represents the information with unit power and $\nu_d \sim \mathcal{CN}(0, \sigma^2)$ is the noise at the HAP. Then, the data rate in the active mode is expressed by

$$r_{a,i} = B \log_2 \left(1 + p_{a,i}|h_i|^2/\sigma^2\right), \tag{1}$$

where B denotes the bandwidth of active data transmission. The relationship between $p_{a,i}$ and $r_{a,i}$ is denoted by:

$$p_{a,i} = \beta(r_{a,i}) \triangleq \left(2^{r_{a,i}/B} - 1\right) \sigma^2/|h_i|^2. \tag{2}$$

Hence, the total power consumption in active mode is given by $\tilde{\beta}\left(r_{a,i}\right) \triangleq \beta\left(r_{a,i}\right) + p_{c,i}$, where $p_{c,i}$ represents the constant power to excite the circuit.

2) Passive Offloading Scheme: For passive offloading, the backscattered signal at HAP can be expressed as $y(n,i) = \alpha g_i b(n) h_i x(n)$, where $x(n)$ denotes the carrier signal emitted by the HAP and $b(n) \in \{0,1\}$ is the binary backscatter information [11]. Here α denotes the reflection coefficient of the backscatter transmitter, which is determined by the antenna's load impedance. Assuming perfect interference cancelation, the direct transmission from the HAP can be subtracted from the received signal. Hence, similar to that in [12], we can simply approximate the data rate in the passive mode as $r_{p,i} = B\log(1 + |\alpha g_i h_i|^2/\sigma^2)$. Typically the backscatter rate $r_{p,i}$ is less than that of the active RF communications. However, power consumption for backscatter communications can be neglectable and sustainable by wireless energy harvesting. This implies that the edge device prefers to use high rate RF communications when energy is sufficient, and turns to backscatter communications if energy becomes insufficient.

3) Local Computing: The edge device can also perform local computing in parallel with data offloading. We allow different edge devices to have different computation capability. Let f_i denote the processor's computing speed (CPU cycles per second) of the i-th user device. The power consumption per CPU cycle can be characterized by kf_i^2 [13], where the constant coefficient k denotes the energy efficiency of computation. Let $0 \le t_{l,i} \le 1$ represent the time allocation for local computing. Then, the total energy consumption in local computing can be modeled by $e_{l,i} = kf_i^3 t_{l,i}$. Let $\phi > 0$ denote the number of cycles required to process one unit workload. Hence, the number of information bits that can be processed locally is given by $\ell_{l,i} = r_{l,i} t_{l,i}$, where $r_{l,i} = f_i/\phi$ can be viewed as the data processing rate in local computing. We assume that the parameters f_i and ϕ are fixed for different user devices. Thus the energy consumption $e_{l,i}$ in local computing only relates to the time and workload allocation $(t_{l,i}, \ell_{l,i})$.

C. Price for MEC Offloading

By offloading the workload to MEC server, the edge nodes consume the MEC server's channel resource to receive workload and return results, as well as the CPU resource to execute computation tasks. The MEC server also charges a price for each user using its MEC offloading service. Let $\ell_{o,i} = \ell_{a,i} + \ell_{p,i}$ represents the total workload offloaded to the MEC server in both active and passive mode. The price for MEC offloading service contains two parts. The first part accounts for the use of the channel resource, which is proportional to the workload offloading rate. A higher offloading rate implies that more channel resource (e.g., bandwidth and energy consumption) will be allocated to receive the workload. The other part depends on the total number of workload, denoting the cost of computing resource, e.g., CPU cycles and storage. In particular, the MEC server sets its price as follows:

$$p_{o,i} = \frac{\mu_o \ell_{o,i}}{t_{a,i} + t_{p,i}} + \rho_o \ell_{o,i},$$

where μ_o denotes the unit channel price [14] and ρ_o represents the unit computing price. The unit prices μ_o and ρ_o can be adjusted properly to ensure that all offloaded workload can be processed within its resource limitation.

III. Policy-based DRL Approach for Hybrid MEC Offloading

Individual user can make offloading decisions based on its local observations. In the following, we focus on a single user and aim to maximize its long-term performance. In the i-th time slot, let $\mathbf{t}_i \triangleq [t_{h,i}, t_{a,i}, t_{p,i}, t_{l,i}]^T$ denote the the time allocation among energy harvesting and different computing schemes. Let $\boldsymbol{\ell}_i \triangleq [\ell_{l,i}, \ell_{a,i}, \ell_{p,i}]^T$ denote the workload allocation among local computing, active, and passive offloading, respectively. We aim to optimize the user's overall performance by optimizing the time and workload allocation strategies $(\mathbf{t}_i, \boldsymbol{\ell}_i)$ in each time slot. In particular, we define the performance metric as follows:

$$R_i(\mathbf{t}_i, \boldsymbol{\ell}_i) \triangleq \frac{L_i}{\tilde{\beta}(r_{a,i})\, t_{a,i} + kf_i^3 t_{l,i}} - wp_{o,i}. \qquad (3)$$

Here the constant weight w represents the user's preference for using MEC offloading service. The first term in (3) represents the energy efficiency in MEC offloading and local computing, defined as the total workload over the total energy consumption. The second term is the cost for using MEC offloading service.

A. Joint Time and Workload Allocation

The edge user's workload in each time slot has to be completed before a fixed delay bound. Assuming that each time slot has unit length, we simply require $t_{h,i}+t_{a,i}+t_{p,i} \le 1$. The workload allocation in three schemes have to fulfill the user's service requirement:

$$\ell_{a,i} + \ell_{p,i} + \ell_{l,i} \ge L_i, \qquad (4)$$

where we have $\ell_{a,i} = t_{a,i} r_{a,i}$ and $\ell_{p,i} = t_{p,i} r_{p,i}$. Workload outage may happen if the workload constraint (4) does not hold, which implies that the workload generated in the i-th time slot cannot be successfully processed within the delay bound. As the computation capability varies in different schemes, this requires an optimal allocation of the workload to minimize the workload outage probability.

Different computation schemes also vary in their energy consumptions. In particular, local computation consumes power in CPU cycles. The active offloading consumes high power in RF communications, while the power consumption in passive offloading is much less than that of RF communications and can be omitted [15]. Hence, the total energy consumption in one time slot is given by $e_i = kf_i^3 t_{l,i} + t_{a,i}\widetilde{\beta}(\ell_{a,i}/t_{a,i})$, corresponding to local computing and active offloading, respectively. Let E_i denote the available energy in the i-th time slot. Hence, the residual energy in the next time slot can be simply denoted as follows:

$$E_{i+1} = \min\left(E_{\max}, \left(E_i + \eta p_0 |g_i|^2 t_{h,i} - e_i\right)^+\right), \qquad (5)$$

where $E_{\max}$ denotes the battery capacity, η denotes energy conversion efficiency, and p_0 denotes the transmit power of HAP. Till this point, we can formulate the performance optimization problem as follows:

$$\max_{\mathbf{t}_i, \boldsymbol{\ell}_i} \quad \mathbb{E}\left[\frac{1}{T}\sum_{i\in\mathcal{T}} R(\mathbf{t}_i, \boldsymbol{\ell}_i)\right] \tag{6a}$$

$$s.t. \quad t_{h,i} + t_{a,i} + t_{p,i} \leq 1, \tag{6b}$$

$$\ell_{a,i} + \ell_{p,i} + \ell_{l,i} \geq L_i, \tag{6c}$$

$$t_{a,i}\widetilde{\beta}(\ell_{a,i}/t_{a,i}) + kf_i^3 t_i \leq E_i + \eta p_0 |g_i|^2 t_{h,i}, \tag{6d}$$

$$\mathbf{t}_i \succeq 0 \text{ and } \boldsymbol{\ell}_i \succeq 0, \quad \forall\, i \in \mathcal{T} \triangleq \{1, 2, \ldots, T\}. \tag{6e}$$

The expectation in the objective function is taken over all instances of the random workload and channel realization. It is clear that problem (6) is very difficult to solve due to the stochastic nature and non-convex structure. In particular, the objective function and the constraints are all non-convex. The battery dynamics in (5) implies a dynamic optimization approach with very high complexity. The uncertainty in workload also makes the optimization impractical for real-time implementation.

B. MDP Reformulation of MEC Offloading Problem

With above practical challenges, the conventional model-based optimization techniques become very inflexible and inefficient. In the following, we resort to a model-free DRL approach to optimize the MEC offloading decisions under uncertain network environment. DRL expands the conventional reinforcement learning approaches for solving Markov decision process (MDP) with large action and state space. The MDP framework for the MEC offloading problem can be defined by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$.

$\mathcal{S}$ represents the system state, denoting the set of observations of the network environment. For each edge user, the system state $s = (\ell, e, c) \in \mathcal{S}$ includes the random workload $\ell \in \{0, 1, \ldots, L\}$ at the beginning of each time slot, the energy $e \in \{0, 1, \ldots, E\}$ in battery, and the finite-state channel condition $c \in \{0, 1, \ldots, C\}$. $\mathcal{A}$ is the continuous action space defined as $\mathcal{A} = \{(\mathbf{t}, \boldsymbol{\ell})\}$, where $\boldsymbol{\ell} = (\ell_l, \ell_a, \ell_p)$ denotes the workload allocation, and $\mathbf{t}$ corresponds to the time allocation $\mathbf{t} = (t_h, t_l, t_a, t_p) \in (\mathbf{0}, \mathbf{1})$ among energy harvesting, local computing, active, and passive offloading. $\mathcal{P}$ is the state transition probability function denoting the distribution of the next state $s_{i+1} \in \mathcal{S}$ given the current state $s_i \in \mathcal{S}$ and the offloading action $a_i \in \mathcal{A}$. This information is typically uncertain to the decision maker and has to be learnt during the interaction with the environment. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that evaluates the quality of action on each state, defined as follows

$$r(s_i, a_i) = \frac{L_i x_i}{\widetilde{\beta}(r_{a,i})t_{a,i} + kf_i^3 t_{l,i}} - wp_{o,i}, \tag{7}$$

We have $x_i = 1$ as the workload is completed successfully, otherwise $x_i = 0$ and there is a waste of computing resources.

Given the dynamics of channel conditions, energy status, and workload, each user device will choose its action accordingly to maximize the accumulated reward $V(s) = \mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^i r(s_i, a_i)|s_0 = s\right]$, where γ denotes the discount factor. Reinforcement learning has provided a solution to find the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maps each network state

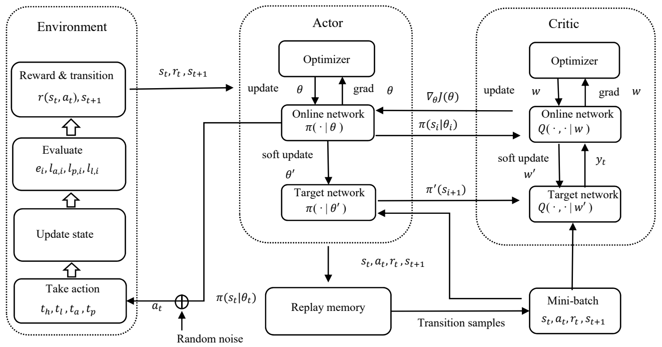


Fig. 2: Workflow of the DDPG algorithm.

$s \in \mathcal{S}$ to an action $a \in \mathcal{A}$ such that the state-value function $V(s)$ is maximized. With small and finite state and action spaces, the optimal policy can be obtained by the Q-learning algorithm. In particular, the optimal action on each state is to maximize the Q-value function $a_i^* = \arg\max_{a\in\mathcal{A}} Q(s_i, a)$, and then we update the Q-value by the difference between the current Q-value and its target y_i as follows:

$$Q_{i+1}(s_i, a_i) = Q_i(s_i, a_i) + \tau_i\Big[y_i - Q_i(s_i, a_i)\Big],$$

where τ_i can be viewed as a step-size and the target value y_i is evaluated by $y_i = r(s_i, a_i) + \gamma \max_{a_{i+1}} Q_i(s_{i+1}, a_{i+1})$.

C. Policy-based DRL for MEC Offloading

The Q-learning algorithm becomes unstable and even fails to converge when the state and action spaces are large. In this part, we introduce DRL to learn the optimal MEC offloading policy, by using DNNs as the approximator for the Q-value function. There are mainly value-based and policy-based DRL approaches. In general, the value-based approaches, such as DQN and its variants, are applicable to discrete action space, while the continuous action space is more preferably tackled by policy-based methods.

Considering the continuous MEC offloading decisions, we adopt the policy-based DRL approach to learn the optimal time and workload allocation strategies. The deep deterministic policy gradient (DDPG) algorithm combines DQN and deterministic policy gradient in the actor-critic framework to make the learning more stable and robust [16], by using the experience replay and target Q-network for DNN training. The policy-based DRL updates the parametric policy in a gradient direction to improve the value function directly, which can be rewritten as $J(\theta) = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a)$, where $d^\pi(s)$ is the stationary state distribution with the policy π_θ. DDPG relies on the deterministic policy gradient theorem that simplifies the gradient evaluation $\nabla_\theta J(\theta)$ as follows

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^\pi(s)}[\nabla_a Q^\pi(s, a)\nabla_\theta \pi_\theta(s)|_{a=\pi_\theta(s)}], \tag{8}$$

where $\pi_\theta(s)$ produces a single deterministic action on the state s, instead of a distribution over the action space. Hence, the estimation $\nabla_\theta J(\theta)$ can be performed efficiently by sampling the historical trajectories.

The policy gradient in (8) motivates the actor-critic framework. The actor network corresponds to the update of policy parameter θ in gradient direction:

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_a Q(s_t, a_t|w_t)\nabla_\theta \pi_\theta(s)|_{a=\pi_\theta(s)},$$

Algorithm 1 DDPG for Hybrid MEC offloading

```
Input: Initial workload, channel and energy conditions
Output: Convergent hybrid MEC offloading policy $\pi^*$
  Initialize actor/critic online network parameters: $\theta$, $w$
  Copy parameters to target networks: $\theta' \leftarrow \theta$, $w' \leftarrow w$
  Initialize replay memory
  for episode $k \leq K$ do
    Reset environment parameters.
    for $t = \{1, 2, ..., T\}$ do
      Actor network sets action $a_t = \pi(s_t|\theta) + \mathcal{N}_t$
      Execute action $a_t$, collect the tuple$(s_t, a_t, r_t, s_{t+1})$
      Sample a mini-batch from replay memory
      Update critic network by minimizing (9)
      Update policy gradient in (8) by the mini-batch
      Update $\theta'$ and $w'$ for the target networks
    end for
  end for
```

where $Q(s_t, a_t|w_t)$ denotes the parameterized Q-function with the DNN weight w_t. For a better exploration in learning, we construct the action by adding a random noise $\mathcal{N}_t$ to $\pi(s_t|\theta_t)$. The critic network estimates the Q-value by updating the DNN weight as follows:

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q(s_t, a_t|w),$$

where $\delta_t = y_t - Q_w(s_t, a_t|w_t)$ denotes the temporal-difference error between $Q_w(s_t, a_t|w_t)$ and its target y_t. The parameters α_θ and α_w are viewed as step-sizes for parameter updating. It is obvious that both the actor and critic networks can be approximated by DNNs and mutually dependent in the learning process.

For the critic network, the training of Q-network is similar to the DQN method by sampling a mini-batch from the experience replay memory. DDPG also adopts the online and target networks to ensure the stability of learning [17]. The DNN training aims to minimize the loss function:

$$L(w) = \mathbb{E}\left[(y_t - Q(s_t, a_t|w))^2\right], \tag{9}$$

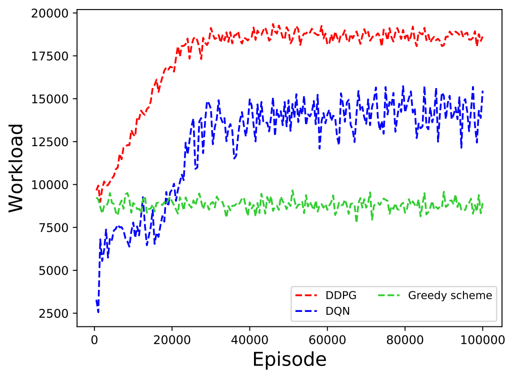
where y_t is updated by $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}|\theta'_t)|w'_t)$ and the training sample (s_t, a_t, r_t, s_{t+1}) is taken from a mini-batch. For a small update rate τ, the parameters w'_t and θ'_t of the target networks are updated by the following rules:

$$w'_{t+1} = \tau w_t + (1-\tau) w'_t, \tag{10}$$
$$\theta'_{t+1} = \tau \theta_t + (1-\tau) \theta'_t, \tag{11}$$

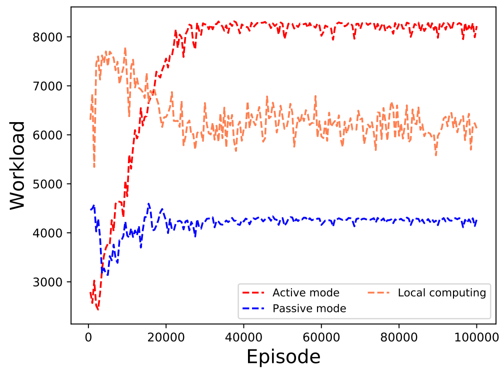
The work flow of the DDPG-based MEC offloading algorithm is organized in Fig. 2.

IV. Numerical Evaluation

In this section, we numerically evaluate the performance of the DDPG algorithm. The transmit power of HAP is set to $p_0 = 10$ mW and energy conversion efficiency is $\eta = 0.6$. The channel remains static within one time slot and follows a finite-state Markov chain in different time sots. We assume that the workload of each user is randomly generated among 0 and 50 kbits. The constant circuit power is set as $p_c = 1$ μW. The noise power is $\sigma^2 = -110$ dBm and the bandwidth is given by $B = 400$ kHz.



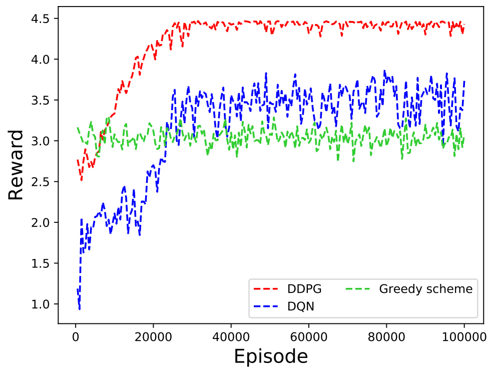
(a) Workload completed by different algorithms



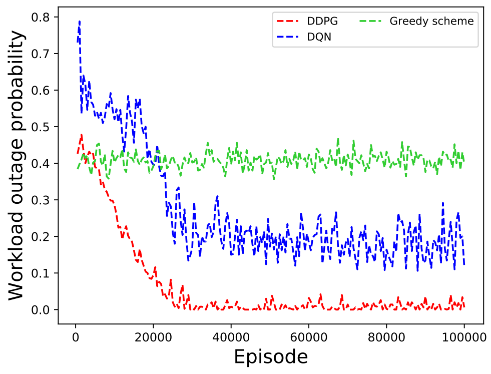
(b) Workload attributed to different computation schemes

Fig. 3: Performance comparison between DDPG and DQN methods.

Figure 3(a) shows the total workload completed by different MEC offloading algorithms. The greedy algorithm means that the user always chooses the myopic action to maximize its instant reward. At convergence, both the DDPG and DQN based MEC offloading schemes perform much better than the greedy algorithm. Besides, the total reward by the DDPG algorithm is always higher than that of the DQN method. We observe that both DDPG and DQN methods reach the convergence after training in 30k episodes. Though the DDPG algorithm does not show better learning speed, it generally has a more stable learning result, as illustrated in Fig. 3(a). Fig. 3(b) shows the workload allocated to different computation schemes, including local computing, active and passive offloading. Before 25k episodes, the workloads in three schemes change dynamically and thus the workload outage probability will be high in the early stage. Initially, local computing completes the largest part of workload. Besides, the passive offloading scheme takes more workload than that of the active offloading scheme. This implies that the edge device initially has insufficient energy supply and prefers local computing and passive offloading scheme with low energy consumption. After 25k episodes, the workload attributed to the active offloading scheme continues to rise. That is because the edge node gradually improves its time allocation strategy and harvests more RF power to sustain its active offloading.



(a) Rewards performance of different algorithm



(b) Outage performance of different algorithm

Fig. 4: Performance comparison of different algorithm.

Figure 4(a) shows the total reward (i.e., the energy efficiency minus the price for MEC offloading service) in different algorithms. It is clear that the DDPG algorithm for continuous time and workload allocation achieves the highest reward, compared to the greedy algorithm and the conventional DQN method for discrete control problem. Typically, the DQN method has to approximate the continuous action space by a finite discrete set, which inevitably brings quantization errors and results in reduced reward performance. Compared to the DQN method, the DDPG algorithm can have a more precise control over the continuous decision variables. This is verified by the stable learning curves in both reward and the outage probability as shown in Fig. 4(b). The precise control in the DDPG algorithm can minimize its outage performance, i.e., nearly all workload in each time slot can be finished successfully.

V. Conclusion

In this paper, considering a continuous control problem, we have proposed a policy-based deep reinforcement learning algorithm to optimize the time and workload allocation in a novel backscatter-assisted hybrid MEC offloading scenario. Comparing with the value-based deep Q-network algorithm, the policy-based deep deterministic policy gradient method achieves significant improvement in terms of reward performance, stability, and the learning speed.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “Mobile edge computing: Survey and research outlook.” [Online]. Available: https://arxiv.org/abs/1701.01090v3

[2] J. Li, J. Xu, S. Gong, X. Huang, and P. Wang, “Robust radio mode selection in wirelessly powered communications with uncertain channel information,” in *proc. IEEE GLOBECOM*, Dec. 2017.

[3] C. Boyer and S. Roy, “Backscatter communication and RFID: Coding, energy, and MIMO analysis,” *IEEE Trans. Commun.*, vol. 62, no. 3, pp. 770–785, March 2014.

[4] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, “Ambient backscatter: Wireless communication out of thin air,” in *Proc. ACM SIGGOMM*, New York, NY, USA, Aug. 2013.

[5] S. Gong, J. Xu, D. Niyato, X. Huang, and Z. Han, “Backscatter-aided cooperative relay communications in wireless-powered hybrid radio networks,” *IEEE Network*, 2019.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, “Applications of deep reinforcement learning in communications and networking: A survey,” *CoRR*, vol. abs/1810.07862, 2018. [Online]. Available: http://arxiv.org/abs/1810.07862

[7] C. Zhang, Z. Liu, B. Gu, K. Yamori, and Y. Tanaka, “A deep reinforcement learning based approach for cost-and energy-aware multi-flow mobile data offloading,” *IEICE Trans. Commun.*, vol. E101-B, no. 7, pp. 2017–2025, 2018.

[8] L. Ji, G. Hui, L. Tiejun, and L. Yueming, “Deep reinforcement learning based computation offloading and resource allocation for mec,” in *proc. IEEE WCNC*, 2018, pp. 1–5.

[9] L. Huang, S. Bi, and Y. A. Zhang, “Deep reinforcement learning for online offloading in wireless powered mobile-edge computing networks,” *CoRR*, vol. abs/1808.01977, 2018. [Online]. Available: http://arxiv.org/abs/1808.01977

[10] J. Li, J. Xu, S. Gong, C. Li, and D. Niyato, “A game theoretic approach for backscatter-aided relay communications in hybrid radio networks,” in *proc. IEEE GLOBECOM*, Dec. 2018.

[11] J. K. Devineni and H. S. Dhillon, “Ambient backscatter systems: Exact average bit error rate under fading channels,” *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, 2019.

[12] W. Chen, W. Liu, L. Gao, S. Gong, C. Li, and K. Zhu, “Backscatter-aided relay communications in wireless powered hybrid radio networks,” in *proc. IEEE WCNC*, 2019.

[13] S. Guo, B. Xiao, Y. Yang, and Y. Yang, “Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing,” in *IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[14] M. Liu and Y. Liu, “Price-based distributed offloading for mobile-edge computing with computation capacity constraints,” *IEEE Wireless Communications Letters*, vol. 7, pp. 420–423, 2018.

[15] D. Niyato, D. I. Kim, M. Maso, and Z. Han, “Wireless powered communication networks: Research directions and technological approaches,” *IEEE Wireless Commun.*, vol. PP, no. 99, pp. 2–11, 2017.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *Proc. Int. Conf. Learning Representations (ICLR)*, May 2016.

[17] H. v. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. AAAI Conf. Artificial Intelligence*, Feb. 2016, pp. 2094–2100.