Joint Scheduling Design in Wireless Powered MEC IoT Networks Aided by Reconfigurable Intelligent Surface

Aichen Li[1], Yang Liu[1], Ming Li[1], Qingqing Wu[2], and Jun Zhao[3]
1: School of Information and Communication Engineering, Dalian University of Technology
2: State Key Lab. of Internet of Things for Smart City, University of Macau
3: School of Computer Science and Engineering, Nanyang Technological University
Emails: {lacz@mail.dlut.edu.cn, {yangliu_613, mli}@dlut.edu.cn, qingqingwu@um.edu.mo, junzhao@ntu.edu.sg}

2021 IEEE/CIC International Conference on Communications in China (ICCC Workshops) | 978-1-6654-3944-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCWorkshops52231.2021.9538853

***Abstract*—Internet of things (IoT) technology is critical to realize universal connections of everything and pervasive intelligence for the future world. The forthcoming IoT technology will be characterized by two predominant features: energy self-sustainability, which is fueled by the recent thrilling wireless power transfer (WPT) technology, and sufficient computation power capability, which will be empowered by the mobile edge computing (MEC) networking. Very recently a promising technology named reconfigurable intelligent surfaces (RIS) has attracted much attention due to its effective beamforming capability and viable potentials to enhance wireless communication system. In this paper we consider exploiting RIS to enhance the WPT-based MEC IoT networks via boosting its energy transferring and communication efficiency. Specifically, we consider the scheduling design through jointly optimizing the WPT-time allocation, dynamic RIS phase control and all IoT mobile devices' offloading decisions to improve the entire MEC network's computation capability. This problem is very challenging due to its high dimension discrete variable space. Here we adopt a reinforcement learning (RL) based online method, which utilizes a novel double deep Q-network (DDQN) structure to effectively overcome the overestimation issue and outperforms the conventional deep Q-network (DQN) learning methods. Numerical results verify the effectiveness of our proposed algorithm and demonstrate the benefits of introducing RIS to assist the WPT-based MEC network.**

***Index Terms*—Mobile edge computing (MEC), reconfigurable intelligent surfaces (RIS), deep reinforcement learning (DRL), wireless power transfer (WPT).**

I. Introduction

The past decade has witnessed the great success of commercialization of 5G wireless communication and nowadays we have been moving forward towards exploring the paradigms of the next generation wireless networks [1]. The Internet of Things (IoT) technology [2] will play a key role in connecting everything to realize the pervasive intelligence, including smart home, smart cities, Industry 4.0 and so on.

One predominant challenge along with the emerging IoT technology is the power supply issue. Considering that huge number of IoT devices will be widely deployed in a large area, manually replacing the devices' batteries will become obviously infeasible solution. Thanks to the recent progress of the wireless power transfer (WPT) technique [3], which makes the self-sustainability of IoT devices realistic via harvesting the electromagnetic (EM) waveforms in the proximal environment.

Another notable difficulty of IoT networks is the universal computation resource deficiency among the IoT nodes, which are generally lost-cost devices with very limited signal processing capability. Fortunately, the mobile edge computing (MEC) resolves this problem via pushing computation resources to the edge of radio access network (RAN) and providing computing services to the IoT devices (offloading).

There are a multitude of research works investigating the system design of WPT based MEC IoT networks. For instance, the work [4] investigates the computing policy design under the energy harvesting and latency constraints. The paper [5] researches the joint offloading scheme and time allocation in wireless powered MEC networks. The article [6] researches edge cloud selection scheme according to the IoT devices battery levels. The authors of [7] develop low complexity solution via introducing approximation to optimize offloading policies in energy harvesting MEC network.

Very recently, the reconfigurable intelligent surfaces (RIS) has been cast with great attention in both academia and industry. RIS is a planar surface comprising an array of controllable reflecting elements. Via collaboratively controlling all reflecting elements [8], RIS can effectively adjust the EM waveforms and therefore improve the network's performance like sum rate, power transferring efficiency and so on [9]-[10].

In this paper, we consider utilizing RIS device to assist the WPT MEC communication system. It should be noted that, although there exist works studying the computation scheduling scheme in WPT MEC systems, e.g. [4]-[7] and RIS control in wireless communication network [8], [9], to the best of our authors' knowledge, the joint optimization of time allocation, RIS phase shifting control and offloading decisions of IoT devices to maximize the entire MEC network's computation capability has never been investigated in the literature, which is one major contribution of this paper. Besides, we propose an enhanced deep reinforcement learning (DRL) method based on the novel double deep Q-network (DDQN) structure to solve the above challenging problem. Numerical results will demonstrate the effectiveness of our solutions and significant performance gain by introducing RIS to the MEC network.

II. System Model and Problem Formulation

A. System Model

This paper considers a RIS-assisted MEC system as shown in Fig. 1. The MEC server can communicate with K single antenna mobile devices and provide them with computing services. Let $\mathcal{K} \triangleq \{1, 2, \cdots, K\}$ represent the set of all mobile devices. An RIS with N reflecting elements is deployed to enhance the performance of the network. Let $\mathbf{H}_{\rm br} \in \mathbb{C}^{N\times M}$, $\mathbf{h}_{{\rm ru},k} \in \mathbb{C}^{N\times 1}$, and $\mathbf{h}_{{\rm d},k} \in \mathbb{C}^{M\times 1}$ denote the wireless channels between the MEC server and the RIS, the RIS between the k-th mobile device, and the MEC server between the k-th mobile device, respectively.

In this context, we assume that the mobile devices are energy self-sustainable and utilize WPT technology to harvest energy from the EM waves emitted from the MEC server. The mobile devices utilize the harvested energy to maintain their regular operations including data collection, communication, computation and so on.

Assume that the communication system operates in a frequency division multiplexing mode, where each mobile device is assigned with a dedicated subchannel with bandwidth B to communicate with the MEC server. In virtue of the first-charge-then-work nature, the mobile devices should repeatedly switch between two working modes: the energy harvesting mode and the task execution mode. Here it is assumed that the entire MEC network works under a frame-based scheduling protocol [6], [12], where the timeline is divided into consecutive frames with each frame being an interval of duration T. Each frame comprises two stages: *i)* the first stage in which the MEC server transfers energy to all mobile devices via broadcasting EM signals; *ii)* the second stage where all mobile devices execute the computation tasks that have arrived during the past T time interval. The two stages of each frame will be specified in detail in the following.

a) Stage-1: Energy harvesting

In the first stage, which is assumed to be of length αT, mobile devices charge their batteries via harvesting the EM waves from the MEC server. Here we assume that the system operates in a TDD mode and the time period T is shorter than the coherence time. Therefore the downlink and uplink channels are reciprocal and remain constant within one frame duration. The MEC server can communicate with the K mobile devices and provide them with computing services. The received signal of the k-th mobile device is given as

$$y_{1,k} = (\mathbf{h}_{{\rm ru},k}^H \mathbf{\Phi}_1 \mathbf{H}_{\rm br} + \mathbf{h}_{{\rm d},k}^H)\sqrt{p}s_k, \forall k \in \mathcal{K}. \quad (1)$$

where $\mathbf{\Phi}_1 = {\rm diag}(e^{j\theta_1^1}, e^{j\theta_1^2}, \cdots, e^{j\theta_1^N})$, $\theta_1^n \in \mathcal{F} \triangleq \{0, \cdots, \frac{2\pi m}{2^M}, \cdots, \frac{2\pi(2^M-1)}{2^M}\}, \forall n \in \mathcal{N} \triangleq \{1, \cdots, N\}$, s_k is the information symbol transmitted by the k-th mobile device satisfying $\mathbb{E}[s_k^2] = 1$, and p is the transmit power of the MEC server. Here we assume the RIS exploits M-bit quantization phase-shifters and consequently each reflecting element can take 2^M possible phase shifts.

The channel between the MEC server and the RIS follows

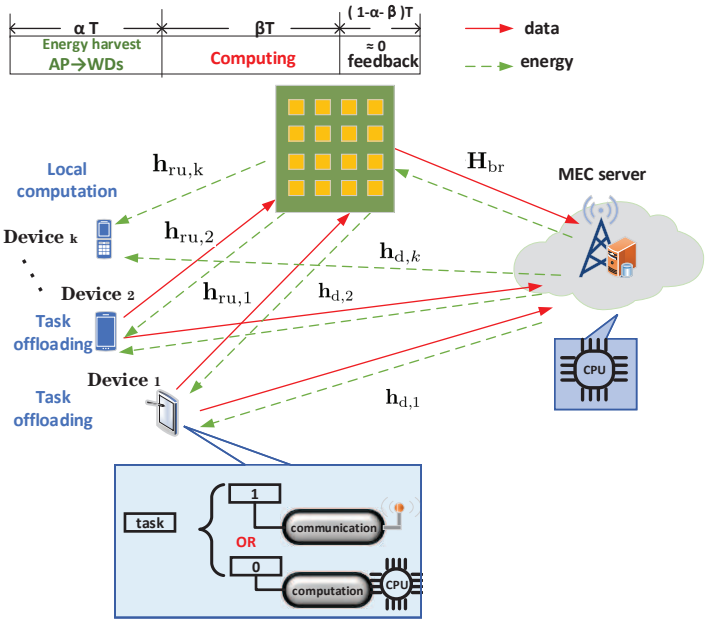

Fig. 1. RIS-assisted mobile edge computation system.

Rician fading, i.e.,

$$\mathbf{H}_{\rm br} = \sqrt{P_l}\left(\sqrt{\frac{l_1}{l_1+1}}\mathbf{H}_{\rm br}^{\rm LoS} + \sqrt{\frac{1}{l_1+1}}\mathbf{H}_{\rm br}^{\rm NLoS}\right), \quad (2)$$

where l_1 is the Rician factor. The path loss P_l is modelled as $P_l = P_{l_0} - 10\delta\log_{10}(d/d_0)$ [17], where d is the distance between the MEC server and the RIS, δ is the path loss exponent, $\mathbf{H}_{\rm br}^{\rm LoS}$ denotes the deterministic LoS component, and $\mathbf{H}_{\rm br}^{\rm NLoS}$ follows Rayleigh fading. Similarly, other channels, $\mathbf{h}_{{\rm ru},k}$ and $\mathbf{h}_{{\rm d},k}$, follow Rician fading model with different Rician factors and pathloss exponents. To simplify the following discussion, we denote the effective channel between the MEC server and the k-th device as

$$\mathbf{h}_{1,k}^H = \mathbf{h}_{{\rm ru},k}^H \mathbf{\Phi}_1 \mathbf{H}_{\rm br} + \mathbf{h}_{{\rm d},k}^H, \forall k \in \mathcal{K}. \quad (3)$$

The received signal in (1) is sent into a rectifier circuit and turned into direct current to charge battery. The amount of energy that the k-th mobile device harvests is given by $E_k = \eta p||\mathbf{h}_{1,k}||_2^2 \alpha T$, where η is the energy harvesting efficiency [15].

b) Stage-2: Computing task execution

During the second stage, the mobile devices perform computations that emerge with various applications like the data fusion, signal processing and so on. To this end, mobile devices can either execute the computing task locally or upload the data to the MEC server, where the task will be done and the associated result will be feedback to the mobile device once the computation is finished. The latter way is namely the *offloading* procedure. For simplicity, here we assume that each mobile device has exactly one computation task in each frame.

In the context of computation offloading, there are two prevalent models to describe offloading tasks—divisible and

indivisible tasks. In the former case, a task can be partitioned into multiple fractions with each fraction being processed in parallel. While for the latter model, any task is regarded as an indivisible atom job and should be executed as a whole. Here we adopt the indivisible model, which applies to most generic scenarios. Assume that the MEC server is equipped with sufficient computation capability and therefore the associated execution time is negligible [6], [7]. Besides, considering that the computation result is usually quite short message, we just assume that the feedback delay from the edge cloud is negligible compared to the time of offloading process, which implies that the duration of the second stage is approximately $\beta = (1-\alpha)T$.

Once the mobile device makes the offloading decision, the computation task will be accordingly executed locally or at the edge cloud end. We introduce the offloading decision variable $v_k \in \{0,1\}$ to denote the k-th mobile device's choice for the task execution manner, with $v_k = 1$ denoting that the computation task is offloaded and $v_k = 0$ otherwise. Both of the computing manners will be elaborated in the following.

1) Local computing

We apply the dynamic voltage and frequency scaling (D-VFS) technique to analyze computation performance [13]. Let f denote CPU's computing frequency (cycles per second), c denote the computation energy efficiency coefficient, and t_k represent the computation time with $0 \le t_k \le (1-\alpha)T$. In terms of the DVFS model, the energy consumption can be represented as $cf^3 t_k$. Under the condition that the power supply is limited, i.e., $cf^3 t_k \le E_k$ [16], to maximize the computation rate, the mobile device should exhaust all the energy and work throughout the whole duration $(1-\alpha)T$ with the optimal frequency $f^* = \sqrt[3]{\frac{E_k}{c(1-\alpha)T}}$ [5]. Here the energy limit is actually the energy harvested during the first stage, i.e., $E_k = \eta p ||\mathbf{h}_{1,k}||_2^2 \alpha T$. We assume that D_k is the number of cycles needed to process one bit of task data, the computation rate (in bits per second) can be represented as

$$\begin{aligned} r_k &= \frac{f^*(1-\alpha)T}{D_k(1-\alpha)T} = \frac{\sqrt[3]{\eta p ||\mathbf{h}_{1,k}||_2^2 \alpha T}}{\sqrt[3]{c(1-\alpha)T D_k}} \\ &= \left(\frac{\eta p}{c}\right)^{\frac{1}{3}} \frac{1}{D_k} ||\mathbf{h}_{1,k}||_2^{\frac{2}{3}} \left(\frac{\alpha}{1-\alpha}\right)^{\frac{1}{3}}, \forall k \in \mathcal{K}. \end{aligned} \tag{4}$$

Then the overall average local computing rate is

$$\begin{aligned} r_{l,k} &= \frac{r_k(1-\alpha)T}{T} \\ &= \left(\frac{\eta p}{c}\right)^{\frac{1}{3}} \frac{1}{D_k} ||\mathbf{h}_{1,k}||_2^{\frac{2}{3}} (\alpha)^{\frac{1}{3}} (1-\alpha)^{\frac{2}{3}}, \forall k \in \mathcal{K}. \end{aligned} \tag{5}$$

2) Computation offloading

If the mobile device decides to offload its task to the edge cloud, it will transmit the data associated with the task to the MEC server using all the energy that has been harvested. The received signal at the MEC server is given as

$$y_{2,k} = (\mathbf{h}_{\mathrm{ru},k}^H \mathbf{\Phi}_2 \mathbf{H}_{\mathrm{br},k} + \mathbf{h}_{\mathrm{d},k}^H)\sqrt{p} s_k + n_0, \forall k \in \mathcal{K}. \tag{6}$$

where $\mathbf{\Phi}_2 = \mathrm{diag}(e^{j\theta_2^1}, e^{j\theta_2^2}, \cdots, e^{j\theta_2^N})$, $\theta_2^n \in \mathcal{F}$ represents the phase shift of the second stage, and $n_0 \sim \mathcal{CN}(0, \sigma^2)$ denotes the noise. The equivalent channel can be rewritten as

$$\mathbf{h}_{2,k}^H = \mathbf{h}_{\mathrm{ru},k}^H \mathbf{\Phi}_2 \mathbf{H}_{\mathrm{br},k} + \mathbf{h}_{\mathrm{d},k}^H, \forall k \in \mathcal{K}. \tag{7}$$

In this case, since the delay due to the computation and feedback are negligible, as previously discussed, the equivalent computation rate is approximately the communication rate given by

$$r_{o,k} = (1-\alpha) B \log_2 \left(1 + \frac{\eta p ||\mathbf{h}_{1,k}||_2^2 ||\mathbf{h}_{2,k}||_2^2 (\frac{\alpha}{1-\alpha})}{n}\right), \forall k. \tag{8}$$

B. Problem Formulation

According to the aforementioned discussion, here we aim to jointly optimize the RIS phase shiftings associated with the two stages $\mathbf{\Phi}_1$, $\mathbf{\Phi}_2$, computing the task assignments $\mathbf{v} \triangleq [v_1, \cdots, v_K]$, and the time allocation α to maximize the utility of the overall MEC network, i.e., the sum computation rate of all mobile devices, which is

$$R_{sum} = \sum_{k=1}^{K} ((1-v_k) r_{l,k} + v_k r_{o,k}). \tag{9}$$

Therefore the joint scheduling problem for a specific frame can be formulated as the following optimization problem

$$\begin{aligned} (\mathsf{P1}): R = \max_{\mathbf{v},\alpha,\theta} \; & R_{sum} \\ \text{s.t. } & 0 \le \alpha \le 1, \\ & v_k \in \{0,1\}, \forall k \in \mathcal{K}, \\ & \theta_1^n, \theta_2^n \in \mathcal{F}, \forall n \in \mathcal{N}. \end{aligned} \tag{10}$$

The problem (P1) is very challenging due to its high dimension variable space. To solve it, we adopt DRL solution, which will be elaborated in the subsequent section.

III. DRL Based Algorithm Design

In this section, we will discuss our proposed DDQN algorithm in detail.

A. The Plain DRL Solution

Model-free RL is a viable dynamic programming tool to learn decision making policy via interacting with environment. Agent and environment are two parts of the RL system. Within each frame, the agent observes environment state s_t after executing an action according to a policy π. The environment transfers to the next state s_{t+1} and feedbacks the agent with a reward r_t which measures the system's utility associated with the current action a_t and the environment state s_t. The agent then stores $\{s_t, a_t, r_t, s_{t+1}\}$ in the experience replay.

To accommodate the RL framework to our problem, we treat the RIS-aided MEC communication system as the environment, other key elements of RL are specified as follows:

State Space: Let $\mathcal{S}$ denote the system state space. The current system state $s \in \mathcal{S}$ includes a set of observations characterizing the environment, which is defined as

$$s = \{\{\mathbf{h}_{1,k}, \mathbf{h}_{2,k}\}_{k\in\mathcal{K}}\}. \tag{11}$$

Action Space: Let $\mathcal{A}$ denote the system action space. According to the observed system state s, the agent chooses the RIS phase shifting, time allocation and the offloading decision $\mathbf{v}$. Hence, the action $a \in \mathcal{A}$ can be defined by

$$a = \{\alpha, \mathbf{v}, \mathbf{\Phi}_1, \mathbf{\Phi}_2\}. \tag{12}$$

Policy: Let $\pi(s_{t+1}|s_t, a_t)$ denote the policy that maps the current state to a probability distribution over corresponding actions.

Reward Function: In RL, the reward acts as a mark to evaluate how beneficial the policy is as the agent executes action given state s. Thus, it is important to design an efficient reward function. In this paper, our goal is to maximize the total computation rate of all mobile devices. Therefore it is reasonable to define the reward function as the computation rate, i.e., $r_t = R_{sum}$.

RL can be utilized to learn strategies maximizing the long term cumulative reward $G_t = \sum_{\tau=0}^{\infty} \gamma^\tau r_{(t+\tau+1)}$, where $\gamma \in (0, 1]$ denotes the discount factor. The Q function $Q_\pi(s_t, a_t)$ is defined as $Q_\pi(s_t, a_t) \triangleq \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$ and represents the expected long-term reward when executing action a_t in state s_t under the policy π, which reflects the impact of selected actions on the objective. It can be shown that the Q function follows Bellman equation that is given by

$$\begin{aligned} Q_\pi(s_t, a_t) =& \mathbb{E}_\pi[r_{t+1} + \gamma \sum_{s_{t+1}\in S} P(s_{t+1}|s_t, a_t) \\ & \sum_{a_{t+1}\in\mathcal{A}} \pi(s_{t+1}, a_{t+1}) Q_\pi(s_{t+1}, a_{t+1})]. \end{aligned} \tag{13}$$

where $\rho \in (0, 1)$ is the learning rate and $P(s_{t+1}|s_t, a_t)$ is the transition probability from state s_t to the next state s_{t+1} after taking action a_t.

In a standard RL algorithm, the agent applies an ϵ-greedy strategy to explore constantly and make decisions. The ϵ-greedy policy has two important ingredients. One is to select the actions associated with the maximum Q-value with probability $1 - \epsilon$. The other is to select an action randomly with probability ϵ to avoid the searching procedure being tracked to local sub-optimal solutions.

One drawback of RL is that in the scenario where the state space and action space have extremely large dimension, e.g., the problem considered here, it is usually unrealistic to construct and update the Q-tables. To address this difficulty, fueled by the recent success of artificial intelligence technique, we can utilize a deep neutral network (DNN) to replace the conventional Q-table via approximating its function. This method significantly decreases the computation complexity [14]. Under the policy π the state-action value function can be rewritten as

$$Q_\pi(s_t, a_t; \theta) = Q_\pi(s_t, a_t), \tag{14}$$

where the $Q_\pi(s_t, a_t; \theta)$ is obtained by approximating the true value of the Q-function, θ represents the parameters of the DNN. The aforementioned improved RL algorithm utilizing deep Q-network (DQN) is called DRL method.

Algorithm 1 Double DQN Algorithm

Input: The empty replay buffer $\mathcal{D}$, the initial network parameters θ, θ', the copy of θ, the learning rate α, the discount factor γ, the soft update coefficient τ, the replacement steps N_r and the batchsize N_B.

Output: The phase shift $\theta_1^1, \theta_1^2, \cdots, \theta_1^N, \theta_2^1, \theta_2^2, \cdots, \theta_2^N$, time allocation α, and mode selection v.

Initialize the replay buffer as 0.
for episode $i = 1, 2, ..., N$ **do**
  for each time step $t = 0, 1, 2, ..., T$ **do**
    Select action a_t based ϵ-greedy policy.
    $a_t = \arg\max_{a_t\in\mathcal{A}} Q(s_t, a_t)$, with probability $1 - \varepsilon$.
    $a_t = \text{random}\{a_i\}_{a_i\in\mathcal{A}}$, with probability ε.
    Observe new state s_{t+1} given action $a_{(t)}$.
    Obtain instantaneous reward r_{t+1}.
    Store the s_t, a_t, s_t, a_t in replay buffer.
    Sample a minibatch of N_B tuples from D.
    Construct target values.
    Define $a_{\max} = \arg\max_{a\in\mathcal{A}} Q(s_{t+1}, a; \theta)$.
    Set $y_j = \begin{cases} r_j, & \text{for terminal } s_{j+1}, \\ r_j + \gamma Q(s_{j+1}, a_{\max}; \theta') & \text{, otherwise.} \end{cases}$
    Perform gradient descent and update Q-network.
    Replace the target network parameters θ with θ' every N_r steps.
  end for
end for

B. Proposed DDQN-based Enhancement

It should be pointed that one notable shortcoming of the DQN algorithm is its overestimation issue. That is the DQN algorithm tends to overestimate the action values, which would consequently degrade the obtained reward severely. Here to overcome this issue, we apply an enhancement scheme, namely the double deep Q-network (DDQN) method [14] to tackle the problem (P1) and further boost the DRL algorithm's performance.

DDQN also has two Q-networks, an evaluation network and a target network. We perform ϵ-greedy search based on the output of the evaluation network, then use the target network to evaluate Q-value, such that there are two network parameters $\theta_t, \theta_t^{'}$. The θ_t in evaluation network is utilized to estimate the value of the current state, and its counterpart $\theta_t^{'}$ in the target network is used for fairly evaluating the value of this policy, which is actually a copy of the θ_t every specific (N_r) steps. In contrast to the DQN algorithm, the key of DDQN lies in the *argmax* operation, which selects the action maximizing the Q-value. Then the updating Q-value function is

$$\begin{aligned} & Q(s_t, a_t; \theta_t) \leftarrow Q(s_t, a_t; \theta_t) + \rho(r_t + \gamma \\ & Q(s_{t+1}, \arg\max_{a\in\mathcal{A}} Q(s_{t+1}, a; \theta_t); \theta_t^{'}) - Q(s_t, a_t; \theta_t)). \end{aligned} \tag{15}$$

where $\arg\max_a Q(s_{t+1}, a)$ denotes the action associated with the maximal Q-value for the next state s_{t+1}.

We calculate the loss between the evaluation network and the target network, then obtain the policy gradients via the Adam iterative algorithm to update the parameters $(\theta_t, \theta_t^{'})$ of DNN. The DDQN algorithm is summarized in Algorithm 1.

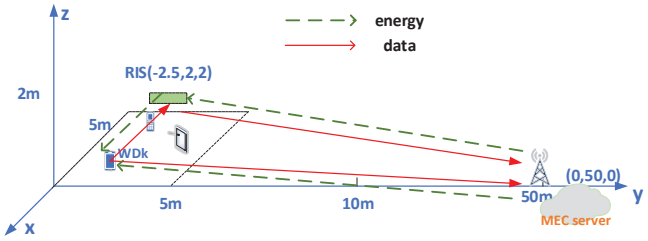

Fig. 2. Simulation setup.

IV. Simulation Results

In this section, we evaluate the performance of the RIS-assisted MEC computing system shown in Fig. 2. We assume that the RIS exploits M = 2 bit quantization phase-shifters. Other settings of the numerical experiment is specified as follows: the energy harvesting efficiency $\eta = 0.7$, the equal computing efficiency $c = 10^{-23}$, the number of cycles needed to process one bit of task data $D_k = 100, \forall k$ [5], each device's data offloading bandwidth is B = 15 MHz, and the received noise power $n_0 = 10^{-12}$ dBm. We set the number of devices as 6 in most testing scenarios, the Rician factor $l_1 = 100$. The path loss is defined as $P_l = P_{l_0} - 10\delta \log_{10}(d/d_0)$ dB, where $d_0 = 1$ m, P_{l_0} = 30 dB, The path loss exponents of the MEC-RIS, RIS-device, and MEC-device links are set to $\delta_{br} = 3.5, \delta_{ru} = 3$ and $\delta_{bu} = 3$, respectively. The learning rate is set to $\rho = 0.0001$ and the discount factor is set to $\gamma = 0.9$. Meantime, the exploration rate is equal to 0.9. In this context, the DNN is a fully connected network which comprises one input layer, two hidden layers and one output layer.

Fig. 3 shows the impact of hyper-parameters of the DDQN on its learning performance. To simplify comparison, we plot in Fig. 3 the normalized computation rate $\hat{R}_{sum} \in [0, 1]$ that is defined as [5]

$$\hat{R}_{sum} = \frac{R_{sum}}{\max\ R'_{sum}}. \tag{16}$$

where the denominator is obtained across all trials using different learning rates. We try to take the learning rate as an example to investigate the impact of hyper-parameters on the performance and convergence rate. As shown in Fig. 3, various learning rates settings can lead to quite different convergence rates. It can be seen that larger learning steps generally result in faster converge but at the same time render the algorithm more easily stuck by local optimal solutions. According to our experiments, 0.0001 is generally a reasonable choice that can always give rise to satisfying converged objective value.

Figure 4 shows that the performance of the two proposed DRL algorithms, DQN and DDQN, with different number of mobile devices deployed when the number of RIS reflecting elements is fixed as $N = 36$. The reward is obtained by running the learned policy. It can be seen that, the DQN's performance, though which usually exhibits advantageous in the very beginning phase, will eventually be outperformed by DDQN, which reflects that DDQN can be used at scale to effectively suppress the overestimation and consequently result in a more advantageous policy.

In Fig. 5, we examine the loss performance of our DDQN

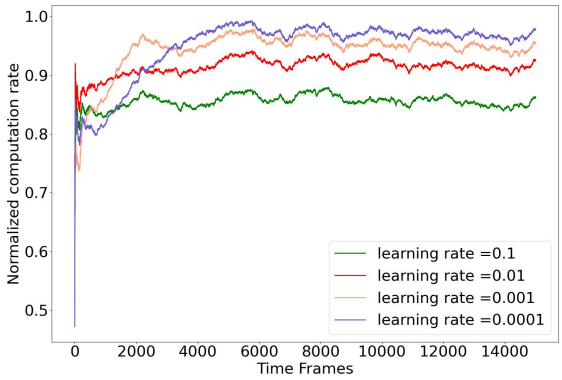

Fig. 3. Normalized computation rate in different learning rate.

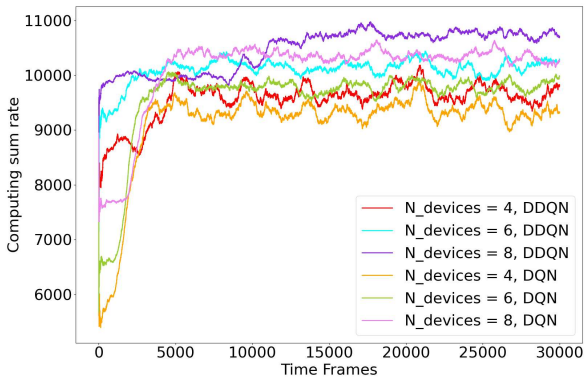

Fig. 4. The sum computation rate in different RL algorithm.

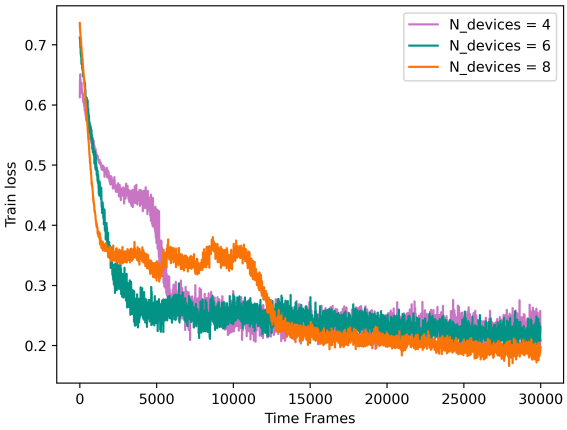

Fig. 5. The training loss of the DNN.

networks under various system settings. As illustrated in the figure, the training loss generally decreases significantly within the first 5000 frames. Under various circumstances, the loss value will always converge to satisfactorily low values, which proves that DNN can achieve a very accurate approximation.

Fig. 6 illustrates the impact of the RIS elements number N and MEC server's transmit power p for the system's computation rate. It can be seen that when the transmit power p increases from 24 dBm to 36 dBm, the computation rate increases correspondingly as p grows. For comparison, we

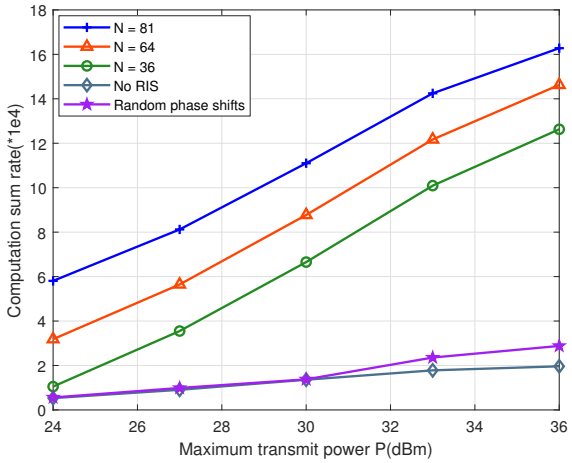

Fig. 6. Performance comparisons versus the transmit power and RIS elements.

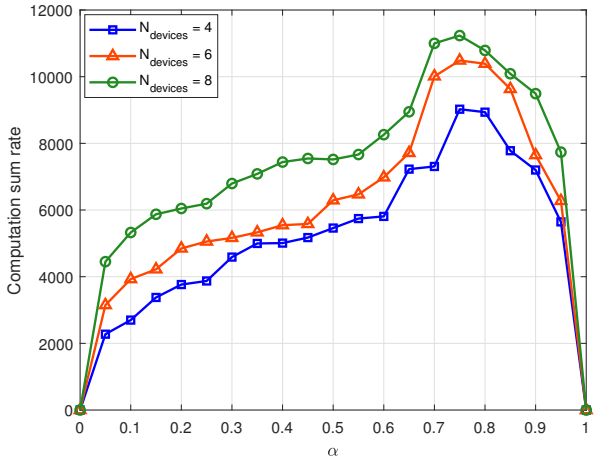

Fig. 7. System rate versus time allocation α.

also plot in Fig. 6 the performance corresponding to no-RIS and random phase RIS scenarios as benchmarks. Clearly, as suggested by the numerical results, the deployment of RIS can significantly boost the system's performance.

In Fig. 7, we analyze the influence of energy harvesting time allocation α on the system performance. It can be seen that system performance achieves optimally when the α is nearly about 0.75 as the time allocation plays the role of trade-off between harvesting energy and computing task execution. A short α can lead to less harvesting energy and lower efficiency of transmission and computation, however, too large α will limit the transmission and computation time, which decreases the computation rate.

V. Conclusions

We consider the joint scheduling problem in an RIS-assisted WPT based MEC communication system. Specifically we investigate the collaborative optimization of WPT time allocation, RIS phase control associated with both the uplink and downlink stages and all mobile devices' offloading decisions. To attack this complicated problem with extremely large discrete valued variable space, we adopt the cutting-edge DDQN algorithm, which utilizes DNN to abstract the high dimension state space and can effectively overcome the overestimation issue that comes along with the conventional DQN method. Extensive numerical results have shown the effectiveness of our proposed algorithm and demonstrated the significant performance improvement via introducing and designing the RIS device.

References

[1] I. F. Akyildiz, A. Kak, and S. Nie, "6G and Beyond: The Future of Wireless Communications Systems," *IEEE Access, vol*. 8, pp. 133995-134030, July. 2020.

[2] X. You, Y. Hao, and H. Wu, "On 6G and Wide-Area IoT," *Chinese J. Internet Things*, vol. 4, no. 1, Mar. 2020.

[3] K. W. Choi, P. A. Rosyady, L. Ginting, A. A. Aziz, D. Setiawan, and D. I. Kim, "Theory and experiment for wireless-powered sensor networks: How to keep sensors alive," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 430-444, Jan. 2018.

[4] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian and D. Niyato, and Y. Zhang, "A joint energy and latency framework for transfer learning over 5G industrial edge networks," *IEEE Trans. Ind. Informat.*, pp. 1-1, Apr. 2021.

[5] L. Huang, S. Bi, and Y. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581-2593, Nov. 2020.

[6] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930-1941, Feb. 2019.

[7] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: a learning method," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4436-4447, June. 2019.

[8] B. Yang, X. Cao, C. Huang, C. Yuen, L. Qian and M. D. Renzo, "Intelligent spectrum learning for wireless networks with reconfigurable intelligent surfaces," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3920-3925, Apr. 2021.

[9] Q. Wu, and R. Zhang, "Joint active and passive beamforming optimization for intelligent reflecting surface assisted SWIPT under QoS constraints," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1735-1748, Aug. 2020.

[10] X. Cao, Bo. Yang, C. Huang, C. Yuen, M. Renzo, Z. Han, D. Niyato, H. Vincent Poor, and L. Hanzo, "AI-assisted MAC for reconfigurable intelligent-surface-aided wireless networks: challenges and opportunities," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 21-27, Jun. 2021.

[11] X. Cao, Bo. Yang, H. Zhang, C. Huang, C. Yuen, and Z. Han, "Reconfigurable intelligen surface-assisted MAC for wireless networks: protocol design, analysis, and optimization," *IEEE Internet Things J.*, pp. 1-1, Mar. 2021.

[12] L. Yu, R. Wang, M. Shi, and J. Wu, "Dynamic offloading design in time-varying mobile edge networks with deep reinforcement learning approach," Mar. 2021. [Online]. Available: https://arxiv.org/abs/2103.02174

[13] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 17, no. 3, pp. 1784-1797, Mar. 2018.

[14] C. Huang, G. Chen, Y. Gong, M. Wen, and J. A. Chambers, "Deep reinforcement learning based relay selection in intelligent reflecting surface assisted cooperative networks," *IEEE Commun. Lett.*, pp. 1036-1040, Feb. 2021.

[15] S. Bi, C. K. Ho, and R. Zhang, "Wireless powered communication: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 117-125, Apr. 2015.

[16] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," *IEEE INFOCOM.*, vol. 53, no. 4, pp. 1-9, Apr. 2016.

[17] K. Feng, Q. Wang, X. Li, and C. Wen, "Deep reinforcement learning based intelligent reflecting surface optimization for MISO communication systems," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 745-749, May 2020.