DRL-Based Computation Rate Maximization for Wireless Powered Multi-AP Edge Computing

Shubin Zhang, Senlei Bao, Kaikai Chi, *Senior Member, IEEE*, Keping Yu, *Member, IEEE*, and Shahid Mumtaz, *Senior Member, IEEE*

***Abstract*—In the ongoing 5G and upcoming 6G eras, the intelligent Internet of Things (IoT) network will take increasingly important responsibility for industrial production, daily life and so on. The IoT devices with limited battery size and computing ability cannot meet many applications brought out by the data-driven artificial intelligence technique. The combination of wireless power transfer (WPT) and edge computing is regarded as an effective solution to this dilemma. IoT devices can collect radio frequency energy provided by hybrid access points (HAPs) to process data locally or offload data to the edge servers of HAPs. However, how to efficiently make offloading decisions and allocate resource is challenging, especially for the networks with multiple HAPs. In this paper, we consider the sum computation rate maximization problem for a WPT empowered IoT network with multiple HAPs and IoT devices. The problem is formulated as a mixed-integer nonlinear programming problem. To solve this problem efficiently, we decompose it into a top-problem of optimizing offloading decisions and a sub-problem of optimizing time allocation under the given offloading decisions. We propose a deep reinforcement learning (DRL) based algorithm to output the near-optimal offloading decision and design an efficient algorithm based on Lagrangian duality method to obtain the consequent optimal time allocation. Simulations verified that the proposed DRL-based algorithm can achieve more than 95 percent of the maximal computation rate with low complexity. Compared with the common actor-critic algorithm, the proposed algorithm has the substantial advantage in convergence speed, achieved computation rate and running time.**

***Index Terms*—Edge computing, wireless power transfer, resource allocation, computation rate maximization.**

I. Introduction

A increasing number of IoT devices are taking important responsibilities in industrial production and providing daily convenience, including intelligent plant, automatic drive, smart city and so on [1]. However, the demand for high intelligence brings new challenges to traditional IoT devices with limited battery capacity and poor computing capacity [2], [3]. Specifically, in the 5G and upcoming 6G eras, an increasing number of applications require the real-time data and corresponding processing results to guarantee high reliability [4]. These application scenarios put forward requirements for higher computing power, more energy storage and more efficient decision making.

S. Zhang, S. Bao and K. Chi are with the School of Computer Science and Technology, Zhejiang University of Technology, China (e-mail: {zhangshubin,2112112043,kkchi}@zjut.edu.cn).

Keping Yu is with the Graduate School of Science and Engineering, Hosei University, Tokyo 184-8584, Japan. (email: keping.yu@ieee.org).

Shahid Mumtaz is with Nottingham Trent University, UK (e-mail: dr.shahid.mumtaz@ieee.org).

In order to solve these series of challenges for intelligent IoT in the new era, a paradigm integrated with mobile edge computing (MEC) and wireless power transfer (WPT) is regarded as an efficient solution for IoT networks [5]–[8]. Using the WPT, the radio frequency (RF) energy is broadcast through the hybrid access points (HAP) to charge IoT devices in the energy transmission stage [9]–[11]. Using MEC is to deploy the server at the edge of the network lie base station [12]. Since the MEC server usually has higher computing ability, IoT devices can transmit the data to the edge server for edge computation [13], [14].

The artificial intelligence (AI) algorithms, including deep learning and deep reinforcement learning (DRL), are promising to be equipped on MECs to effectively and intelligently control all the IoT devices [15], [16]. The integration of WPT, MEC and AI gains much attention in recent years [18]. The available works mainly focus on designing DRL method to make offloading decisions and optimize resources allocation [19], [20], so as to improve the performance and efficiency of the algorithm.

To the best of our knowledge, there are very few literatures focusing on the WPT empowered MEC with multiple HAPs, which is practical for large-scale IoT networks. The available algorithms designed for single HAP scenario can hardly be directly applied to solve the offloading problem with multiple HAPs. The exponential growth of the action space with the number of WDs and HAPs makes it difficult for the available algorithms like deep Q-network (DQN) [21] and supervised learning [22] to come to converge. In addition, the resource allocation problem of multiple HAPs scenario is more complex than that of single HAP scenario.

This paper considers a WPT empowered MEC with multiple HAPs and IoT devices. Aiming to maximize the sum computation rate (SCR), we propose a DRL-based algorithm to intelligently determine the offloading decision and an efficient algorithm based on Lagrangian duality method to optimize the resource allocation.

The main contributions of this paper are summarized as follows:

- The sum computation data maximization is formulated as a mixed-integer nonlinear programming (MINLP) problem. To efficiently solve it, we decompose it into a top-problem of optimizing offloading decision and a sub-problem of optimizing time allocation under the given offloading decision.
- We propose an online deep neural network (DNN) based DRL framework to output the near-optimal offloading

decision, that is, determine whether one WD conducts the computation locally or offloads the data to one of HAPs. The proposed DRL method can greatly reduce the offloading decisions' space and converge fast during the online training process.

- For the concave sub-problem, based on Lagrangian duality method, a low-complexity algorithm is designed for obtaining the optimal time allocation.
- We demonstrate through simulations that, compared with the exhaustive search method, the proposed algorithm can achieve more than 95 percent of the maximal computation rate with low complexity.

The structure of this paper is as follows: Section II introduces related works. Section III provides the system model. Section IV formulates the problem and introduces the DRL-based algorithm. Simulation results are given in Section V. Finally, we make a conclusion in Section VI.

II. Related Works

According to different task models, the task offloading modes of wireless devices (WDs) in WPT-MEC networks are mainly partial offloading and binary offloading, depending on whether the tasks of WDs are inseparable. Additionally, the considered performance metrics are mainly computation rate, energy efficiency, energy consumption, etc. Table I summarizes some related works focusing on partial offloading and binary offloading.

A. Partial Offloading

In [23], one WD relays part of the other WD's data to the server and process the remaining part. The total amount of data processed within a given time frame was maximized by jointly optimizing the time allocation, the transmit power, etc. Zhou *et al.* [24] considered TDMA-based WPT-MEC network where unmanned aerial vehicle (UAV) was used for wireless energy supply and maximized the weighted SCR for both partial offloading and binary offloading. In [25], Zhou *et al.* also considered TDMA-based WPT-MEC and proposed a soft actor-critic based UAV trajectory planning and resource allocation algorithm to maximize the computation bits of WDs in a fixed time period. In [17], we studied FDMA-based MEC networks, and proposed an online offloading algorithm based on DRL to maximize the SCR.

Shi *et al.* [26] considered the NOMA-based WPT-MEC network, and maximized the energy efficiency of the network by jointly optimizing the frequency and time resources of the MEC server and IoT devices, as well as the transmission powers of IoT devices and energy source.

[27] considered the TDMA-based WPT-MEC where AP always charges WDs through beamforming technology. By jointly optimizing beamforming, CPU frequency, offloaded data volume and user time allocation, the total energy consumption of AP was minimized while ensuring that all WDs' tasks are completed within a fixed duration. [28] considered the scenario where the dynamically arrived tasks in a fixed period must be processed in this period. The total energy consumption of the system was minimized by jointly optimizing AP's energy beamforming, local computation and task offloading.

In [29], each WD has its own power source and only harvests RF energy to supplement its power consumption. Considering the WDs' tasks with a latency constraint, the authors optimized data partitioning, transmit power and time allocation for computation offloading to minimize the transmission energy consumption, and optimized the energy beamforming of AP for wireless charging to maximize the received charging energy.

B. Binary Offloading

For TDMA-based binary offloading, [30] maximized the SCR. For NOMA-based binary offloading, Zeng *et al.* [31] proposed a channel-gain based greedy algorithm, aiming to maximize the SCR. Nguyen *et al.* [32] investigated backscatter-assisted data offloading in OFDMA-based WPT-MEC system and maximized the SCR by jointly optimizing the AP's transmit power, backscatter coefficient, time allocation, and offloading decision.

[33] considered both the TDMA and NOMA modes and both the partial offloading and binary offloading. While meeting the minimum computation amount of each WD, the minimal computing efficiency of all WDs was maximized.

Wang *et al.* [34] optimized the weighted difference between the overall computation rate and energy consumption, aiming to achieve as much computation rate and as little energy consumption as possible. In [35], the energy source forms K directional RF energy beams for K nodes. Under the given WPT duration and offloading duration, the total utility of OFDMA-based offloaded data amount and AP's energy consumption was maximized while ensuring the constraint of node task completion delay.

Liu *et al.* [36] studied the wireless-powered fog-cloud computing network, where WDs choose to offload computing tasks to fog servers, cloud servers, or full local computing. The minimum energy surplus among all WDs was maximized by jointly optimizing time allocation, CPU computing frequencies and computing mode of each WD.

[37] minimized the computation delay when each WD has a computation task to execute.

III. System Model

A. Network Model

Fig. 1 shows the considered network with N WDs denoted as $\{WD_1, WD_2, \ldots, WD_N\}$ and M HAPs denoted as $\{HAP_1, HAP_2, \ldots, HAP_M\}$. Let $\mathcal{N} = \{1, 2, ..., N\}$ and $\mathcal{M} = \{1, 2, ..., M\}$. Each HAP carries one server. HAPs are connected with the power source and integrated with energy transfer circuit for broadcasting RF energy to the WDs. Every WD can harvest the RF energy transmitted by HAPs, store the harvested energy to its battery through energy transfer circuit and provide power for the subsequent operations (local computation or data transmission). When WDs are about to process the data, they can execute the computation locally by their own computing units or choose one certain HAP to

TABLE I
SUMMARY OF SOME RELATED WORKS FOCUSING ON PARTIAL OFFLOADING AND BINARY OFFLOADING.

References	Access scheme	Offloading mode	Scenario	Technique	Objectives
He *et al.* [23]	TDMA	Partial offloading	A MEC server with two users	Convex optimization	SCR
Zhou *et al.* [25]	TDMA	Partial offloading	A UAV with multi-users	DRL + Convex optimization	SCR
Shi *et al.* [26]	NOMA	Partial offloading	A MEC server with multi-users	Convex optimization	Energy efficiency
Wang *et al.* [28]	TDMA	Partial offloading	A MEC server with multi-users	Convex optimization	Energy consumption
Zeng *et al.* [31]	NOMA	Binary offloading	A MEC server with multi-users	Greedy algorithm	SCR
Nguyen *et al.* [32]	OFDMA	Binary offloading	A MEC server with multi-users	Optimization algorithm	SCR
Zhou *et al.* [33]	TDMA+NOMA	Binary offloading + partial offloading	A MEC server with multi-users	Optimization algorithm	Energy efficiency
Wang *et al.* [34]	TDMA	Binary offloading	A MEC server with multi-users	DRL	Energy efficiency
Our paper	TDMA	Binary offloading	Multiple MEC servers with multi-users	DRL + Convex optimization	SCR

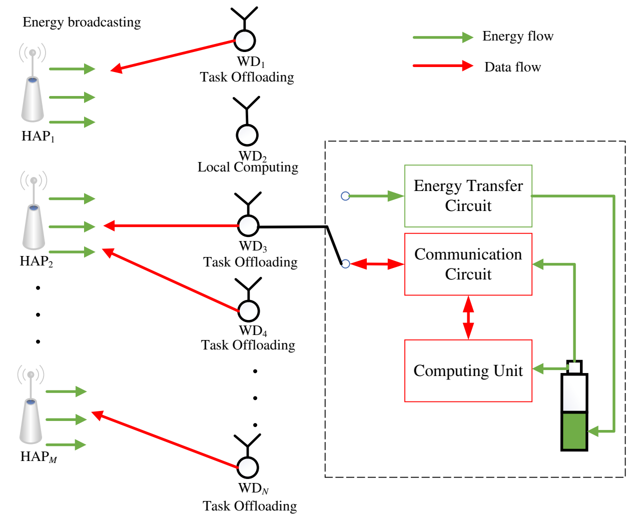


Fig. 1. The considered multi-HAP WPT empowered edge computing.

offload data by communication circuit. When HAPs receive data from WDs, HAPs will execute the computation by their servers and transmit the result to WDs by communication circuit.

In the considered scenario, for the WDs associated to the same HAP, they communicate with HAP by using TDMA protocol. The WDs associated to different HAPs use orthogonal channels. WDs adopt harvest-then-transmit (HTT) protocol which means they need to harvest energy before offloading data to HAP. This network system can represent the IoT network empowered by WPT and MEC, which is with the requirements of long lifetime and complex computing task, such as air quality monitoring [38], smart home appliance [39] and so on.

B. Channel Model

In our scenario, system time is divided into time slots with duration T, and the channel gain remains the same in one time slot but varies in different time slots. h_{ij} represents the channel gain between WD_i and HAP_j, which is expressed as:

$$h_{ij} = |g_{ij}|^2 \alpha_{ij}, \tag{1}$$

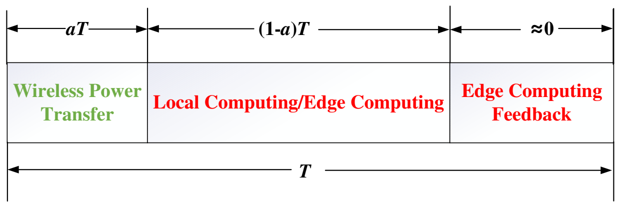


Fig. 2. Time allocation for the WPT empowered edge computing.

where α_{ij} represents large-scale fading component and g_{ij} represents the small-scale fading between WD_i and HAP_j.

C. Energy Harvesting Model

One time slot mainly consists of two parts (refer to Fig. 2): WPT phase and data offloading phase. In the WPT phase, a WD can harvest RF energy from every HAP and store the energy in its battery. The amount of energy captured by WD_i is:

$$E_i = \sum_{j=1}^{M} \mu P h_{ij} a T, \tag{2}$$

where μ represents the energy capture efficiency, P represents the energy transmit power of each HAP and a presents energy harvest time ratio. In data offloading phase, each WD can choose to offload data to one HAP or conduct the computation locally. The detail of both computing modes is introduced below.

D. Local Computing Model

If one WD processes its data locally by its own CPU, it can harvest the RF energy while processing the data during the whole time slot. ϕ represents the number of CPU cycles consumed to process 1-bit data for WDs. f_i represents WD_i's CPU calculation speed in the unit of cycles/second, whose value can be adjusted. So the amount of processed data can be expressed as $f_i t_i/\phi$, where t_i is the computation time of WD_i. The CPU energy consumption is modeled as $cf_i^3 t_i$, where c is the computation energy efficiency coefficient. Furthermore, the consumed energy should be smaller than the harvested energy, i.e., $cf_i^3 t_i \leq E_i$. It is not hard to know that to maximize the processed data within a time slot, the optimal value of t_i is T and the optimal value of f_i is $(\frac{E_i}{cT})^{1/3}$. Thus, the local computing rate of WD_i is:

$$\begin{aligned} r_{l,i} &= \frac{f_i t_i}{\phi T} \\ &= \frac{(\sum_{j=1}^{M} \mu P h_{ij} a/c)^{1/3}}{\phi} \\ &= \eta_1 \left(\sum_{j=1}^{M} h_{ij} a \right)^{1/3}, i \in \mathcal{N}, \end{aligned} \tag{3}$$

where $\eta_1 \triangleq \frac{(\mu P/c)^{\frac{1}{3}}}{\phi}$.

E. Edge Computing Model

In the edge computing mode, a WD needs to choose a HAP to offload data. All the WDs connected to the same HAP share the $(1-a)T$ time duration to offload data. Denote the time allocated for WD_i connected with HAP_j as $\tau_{ij}T$. Notice that for WD_k which is not associated to HAP_j, $\tau_{kj} > 0$ just wastes the time resource of HAP_j and is clearly not the optimal solution.

The amount of feedback after HAP processed the data offloaded from WD_i is denoted by $\rho_i b_i$, where $0 < \rho \ll 1$ indicates the ratio of output/input in downlink transmission [30]. We denote f_0 and P_0 as CPU frequency and transmission power of HAPs respectively. Then the time spent by HAP_j on data processing and feedback to WD_i can be expressed as: $t_0 = \frac{\phi b_i}{f_0} + \frac{\rho b_i}{B log_2\left(1+\frac{P_0 h_{ij}}{N_0}\right)}$. Considering the CPU frequency and the transmit power of HAPs is more than three orders of magnitude stronger than WDss [27], it's reasonable to neglect the time spent on data processing and result feedback. Hence a time slot is mainly divided as two parts as shown in Fig. 2 and the following condition must hold:

$$\sum_{i=1}^{N} \tau_{ij} + a \le 1, \forall j \in \mathcal{M}. \tag{4}$$

The amount of offloaded data b_i of WD_i is:

$$b_i = \frac{B\tau_{ij}T}{v_u} \log_2 \left(1 + \frac{P_i h_{ij}}{N_0}\right), \tag{5}$$

where B represents the bandwidth, $v_u \ge 1$ represents the factor that represents communication overhead in data offloading, including the encryption cost and data header, P_i is transmit power of WD_i, and N_0 is additive white Gaussian noise (AWGN) power.

In order to ensure that WD_i can offload as much data as possible, the energy it harvested in WPT phase should be exhausted, i.e., $P_i = \frac{E_i}{\tau_{ij}T}$, thus the maximal computation rate of edge computing for WD_i can be expressed as:

$$\begin{aligned} r_{o,ij} &= \frac{b_i}{T} \\ &= \frac{B\tau_{ij}}{v_u} \log_2 \left(1 + \frac{\mu P a h_{ij} \sum_{k=1}^{M} h_{ik}}{\tau_{ij} N_0}\right) \\ &= \epsilon \tau_{ij} \ln \left(1 + \frac{\eta_2 h_{ij} \sum_{k=1}^{M} h_{ik}}{\tau_{ij}} a\right), \end{aligned} \tag{6}$$

where $\epsilon \triangleq \frac{B}{v_u \ln 2}$ and $\eta_2 \triangleq \frac{\mu P}{N_0}$.

IV. Problem Formulation and Efficient Algorithm

In this section, we formulate the SCR maximization problem, propose an offloading algorithm based on DRL and design a Lagrangian duality based algorithm for time allocation.

A. Problem Formulation

Let $\boldsymbol{h} = [\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_N}]$ where $\boldsymbol{h_i} = [h_{i1}, h_{i2}, \ldots, h_{iM}]$ is the channel gain of WD_i. Let $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}]$ where $\boldsymbol{x_i} = [x_{i,0}, x_{i,1}, \ldots, x_{i,M}]$ is the offloading decision indicator of WD_i. $x_{i,0} = 1$ represents WD_i process task by local computing, $x_{i,j} = 1 (j \neq 0)$, represents that WD_i offloads data to HAP_j. Let $\boldsymbol{\tau} = [\boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \ldots, \boldsymbol{\tau_N}]$ where $\boldsymbol{\tau_i} = [\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,M}]$.

Then combining (3) and (6), the SCR of one time slot is given by:

$$Q(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\tau}, a) \triangleq \sum_{i=1}^{N} \boldsymbol{x_i} \cdot [r_{l,i}, r_{o,i1}, r_{o,i2}, \ldots, r_{o,iM}]^{\mathrm{T}}, \tag{7}$$

In the scenario we considered, our goal is to maximize the SCR achieved in one time slot. So the problem is formulated as (P1):

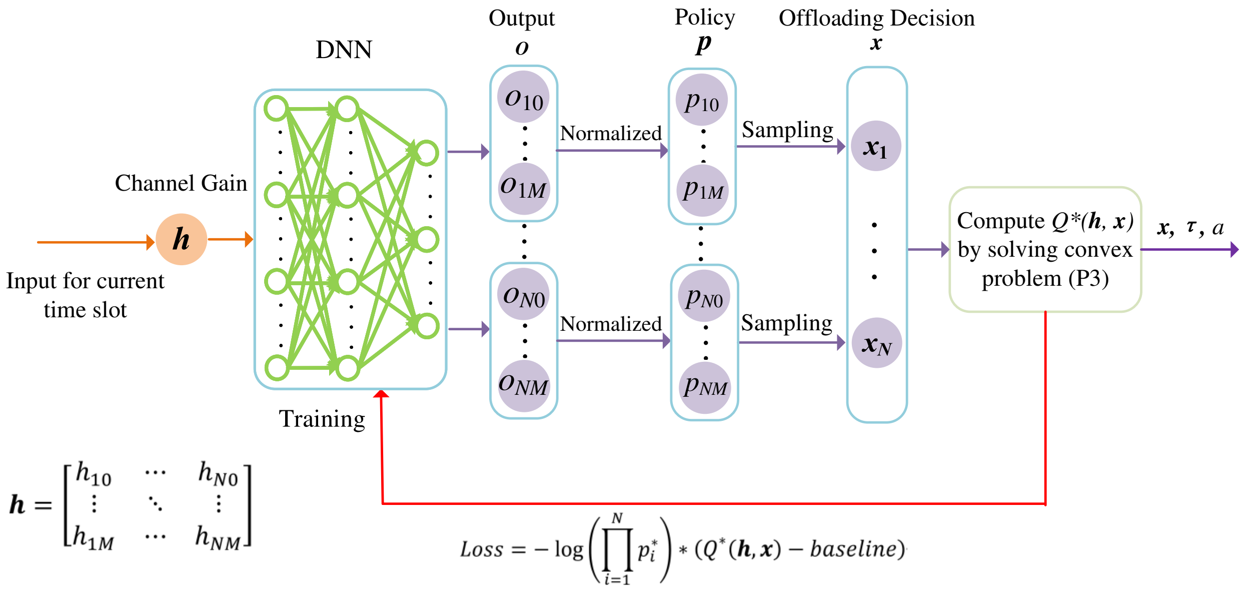
$$\begin{aligned} (P1): Q^*(\boldsymbol{h}) = \ & \underset{\boldsymbol{x}, \boldsymbol{\tau}, a}{\text{maximize}} \quad Q(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\tau}, a) && \text{(8a)} \\ \text{s.t.} \quad & 0 \le \tau_{ij} \le 1, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}, && \text{(8b)} \\ & 0 \le a \le 1, && \text{(8c)} \\ & \sum_{i=1}^{N} \tau_{ij} + a \le 1, \forall j \in \mathcal{M}, && \text{(8d)} \\ & \sum_{j=0}^{M} x_{i,j} = 1, \forall i \in \mathcal{N}, && \text{(8e)} \\ & x_{i,j} \in \{0, 1\}, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}. && \text{(8f)} \end{aligned}$$

Constraints (8b), (8c) and (8d) mean that each WD's offloading duration allocated from each HAP, the WPT duration, and the total allocated time from each HAP cannot exceed the slot duration, respectively. Constraint (8e) means that each WD can be associated to at most one HAP.

It's obvious that problem (P1) is an MINLP problem which is hard to solve. However, once the value of $\boldsymbol{x}$ is given, (P1) can be transformed to the following concave sub-problem (P2).

$$\begin{aligned} (P2): Q^*(\boldsymbol{h}, \boldsymbol{x}) = \ & \underset{\boldsymbol{\tau}, a}{\text{maximize}} \quad Q(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\tau}, a) && \text{(9a)} \\ \text{s.t.} \quad & 0 \le \tau_{ij} \le 1, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}, && \text{(9b)} \\ & \sum_{i=1}^{N} \tau_{ij} + a \le 1, \forall j \in \mathcal{M}, && \text{(9c)} \\ & 0 \le a \le 1. && \text{(9d)} \end{aligned}$$

The algorithm for solving sub-problem (P2) will be introduced later. Below we first introduce the DNN based DRL algorithm, which is able to output the near-optimal offloading decisions according to $\boldsymbol{h}$ of WDs in the current time slot.



DNN
Output $\boldsymbol{o}$
Policy $\boldsymbol{p}$
Offloading Decision $\boldsymbol{x}$
Channel Gain
Input for current time slot
$\boldsymbol{h}$
o_{10}
o_{1M}
o_{N0}
o_{NM}
Normalized
Normalized
p_{10}
p_{1M}
p_{N0}
p_{NM}
Sampling
Sampling
$\boldsymbol{x}_1$
$\boldsymbol{x}_N$
Compute $Q^*(\boldsymbol{h}, \boldsymbol{x})$ by solving convex problem (P3)
$\boldsymbol{x}, \tau, a$
Training
$\boldsymbol{h} = \begin{bmatrix} h_{10} & \cdots & h_{N0} \\ \vdots & \ddots & \vdots \\ h_{1M} & \cdots & h_{NM} \end{bmatrix}$
$Loss = -\log\left(\prod_{i=1}^{N} p_i^*\right) * (Q^*(\boldsymbol{h}, \boldsymbol{x}) - baseline)$

Fig. 3. The framework of the proposed algorithm.

TABLE II
NOTATIONS USED THROUGHOUT THE PAPER

Notation	Description
N	The number of WDs
M	The number of HAPs
T	The length of time slot
$\mathcal{N}$	The set of WDs
$\mathcal{M}$	The set of HAPs
h_{ij}	The channel gain between WD_i and HAP_j
α_{ij}	The large-scale fading component between WD_i and HAP_j
g_{ij}	The small-scale fading between WD_i and HAP_j
E_i	The amount of energy captured by WD_i
μ	The energy capture efficiency
P	The energy transmit power of HAPs
a	The energy harvest time ratio
$r_{l,i}$	The local computation rate of WD_i
ϕ	The number of CPU cycles consumed to process 1-bit data
ρ	The ratio of output/input in downlink transmission
f_0	The CPU calculation speed of HAPs
P_0	The transmit power of HAPs
t_0	The computation and result feedback time used by HAP in edge computing
f_i	The CPU calculation speed of WD_i
t_i	The computation time used by WD_i in local computing
c	The computation energy efficiency coefficient
τ_{ij}	The offloading time allocated ratio for WD_i connected with HAP_j
b_i	The amount of offloaded data offloaded by WD_i
B	The communication bandwidth
v_u	The communication overhead in data offloading
P_i	The transmit power of WD_i
N_0	The additive white Gaussian noise power
$r_{o,ij}$	The computation rate of edge computing for WD_i offloads data to HAP_j
$Q(\cdot)$	The sum computation rate function
x_{ij}	The offloading indicator for WD_i
$\boldsymbol{\tau_i}$	The vector of offloading time allocated ratio for WD_i
π	The policy function
θ	The parameters of DNN
$\boldsymbol{o}$	The origin output of DNN
p_{ij}	The probability of WD_i executes offloading decision
p_i^*	The probability of WD_i actually executing offloading decision
β	The learning rate of DNN

B. The DRL Algorithm for Offloading Decision

The whole framework of our algorithm is shown in Fig. 3. Denote the DNN by π_θ, where θ represents the weights and bias in the DNN. The input of the DNN is the channel gain $\boldsymbol{h}$ and the output is a vector $\boldsymbol{o}$ consisting of $N \times (M+1)$ elements, i.e.,

$$\boldsymbol{o} = \pi_\theta(\boldsymbol{h}). \tag{10}$$

In the beginning of a time slot, $\boldsymbol{h}$, which consists of all the $N \times M$ WD-HAP channel gains, is input into DNN π_θ. Then the DNN outputs $\boldsymbol{o} = [o_{10}, ..., o_{1M}, ..., o_{N0}, ...o_{NM}]$. For the sake of easy understanding, we can see $\boldsymbol{o}$ as a matrix with dimension $(M+1) \times N$:

$$\boldsymbol{o} = \begin{bmatrix} o_{10} & o_{20} & \cdots & o_{N0} \\ o_{11} & o_{21} & \cdots & o_{N1} \\ \cdots & \cdots & \cdots & \cdots \\ o_{1M} & o_{2M} & \cdots & o_{NM} \end{bmatrix}. \tag{11}$$

Then, every column vector of $\boldsymbol{o}$ is normalized and a new matrix $\boldsymbol{p}$ is obtained:

$$\boldsymbol{p} = \begin{bmatrix} p_{10} & p_{20} & \cdots & p_{N0} \\ p_{11} & p_{21} & \cdots & p_{N1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{1M} & p_{2M} & \cdots & p_{NM} \end{bmatrix}, \tag{12}$$

where $p_{ij} = \frac{o_{ij}}{\sum_{k=0}^{M} o_{ik}}$.

It is obvious that, the sum of the elements in one column is 1, i.e., $\sum_{j=0}^{M} p_{ij} = 1, \forall i \in \mathcal{N}$. $\boldsymbol{p}$ is the policy to make offloading decision for every WD. Every column vector $\boldsymbol{p_i}$ of matrix $\boldsymbol{p}$ represents the offloading policy for WD_i, i.e., the probabilities of offloading data to different HAPs. Its element p_{i0} represents the probability that WD_i conducts the computation locally, and $p_{ij}(j \neq 0)$ represents the probability that WD_i offloads data to HAP_j. Then we randomly sample offloading decision according to the obtained probability distribution $\boldsymbol{p_i}$ and use the sampling result $\boldsymbol{x}_i$ as the offloading decision of WD_i. Note that the sampling only leads to one of following cases for WD_i: local computing ($x_{i,0} = 1$), offloading data to HAP_1 ($x_{i,1} = 1$), ..., offloading data to HAP_M ($x_{i,M} = 1$).

Let $p_i^* = p_{ij}$ satisfying $x_{i,j} = 1$. p_i^* will be used to calculate the loss of obtained offloading decision for training the DNN.

We use one example to illustrate how to obtain $\boldsymbol{p}$ and $\boldsymbol{x}$ from $\boldsymbol{o}$. In a network containing 3 WDs and 2 HAPs, suppose that the original output of the DNN is:

$$\boldsymbol{o} = \begin{bmatrix} 0.04 & 0.02 & 0.04 \\ 0.28 & 0.12 & 0.02 \\ 0.02 & 0.2 & 0.10 \end{bmatrix}. \tag{13}$$

Then after normalization, we have the following $\boldsymbol{p}$:

$$\boldsymbol{p} = \begin{bmatrix} 0.12 & 0.059 & 0.25 \\ 0.82 & 0.353 & 0.125 \\ 0.06 & 0.588 & 0.625 \end{bmatrix}. \tag{14}$$

Suppose that after random sampling, WD_1, WD_2 and WD_3 choose the computation mode as local computing, offloading data to HAP_2 and offloading data to HAP_1, respectively. Then we can get the offloading decision $\boldsymbol{x_1} = [1, 0, 0]$, $\boldsymbol{x_2} = [0, 0, 1]$, and $\boldsymbol{x_3} = [0, 1, 0]$. So $p_1^* = p_{10} = 0.12$, $p_2^* = p_{22} = 0.625$, and $p_3^* = p_{31} = 0.125$.

The training of this proposed DRL algorithm for offloading decision has two important parts: setting a baseline of offloading decision's utility and updating DNN.

- Setting baseline: For every time slot, we need a baseline utility to evaluate the goodness of obtained offloading decision, such that we can give reward (positive value) for a good decision and give punishment (negative value) for a bad decision. We set the average SCR of the most recent K time slots as the baseline utility. The appropriate value of K will be investigated later.
- Updating DNN: We adopt a policy gradient-like updating method. Considering that the occurrence probabilities of WDs' offloading decisions are independent of one another, the probability of obtained offloading decision $\boldsymbol{x}$ is $\prod_{i=1}^{N} p_i^*$. Hence, the loss function can be expressed as:

$$Loss = -\log \left(\prod_{i=1}^{N} p_i^* \right) \times \left(Q^* \left(\boldsymbol{h}, \boldsymbol{x} \right) - baseline \right). \tag{15}$$

The policy provided by DNN will be improved with time slots going by and converges after a number of time slots. Note that the proposed DRL algorithm is different from supervised learning. For the supervised learning, a number of samples with labels must be prepared in advance for training the DNN. For our algorithm, there is no need to prepare the samples with labels, which will generate the samples during its running in the environment. The proposed algorithm is a kind of policy-based DRL. Compared with the common policy-based DRL such as the actor-critic (AC) algorithm, we abandon critic network in the original stochastic policy gradient algorithm and use the reward function to judge the actions' goodness. The reason is that our scenario only considers a single time slot, and the objective function (or can be seen as a reward function) can accurately measure the value of the current action without bias by the real value directly obtained in the environment.

Algorithm 1 DRL based offloading algorithm

```
Require: $\boldsymbol{h}$, number of time slots $num$;
Ensure: $\boldsymbol{x}$;
1: Initialize parameters $\theta$ of DNN;
2: while $t < num$ do
3:     Generate output $\boldsymbol{o}$ by DNN with input of $\boldsymbol{h}$;
4:     Normalize every $M+1$ elements in $\boldsymbol{o}$ and obtain policy $\boldsymbol{p}$;
5:     Do sampling according to $\boldsymbol{p}$: $\boldsymbol{x} = sampling(\boldsymbol{p})$;
6:     With the obtained $\boldsymbol{x}$, solve Problem (P3) and get $Q^*(\boldsymbol{h}, \boldsymbol{x})$;
7:     Calculate the loss of DNN network: $Loss = -\log\left(\prod_{i=1}^{N} p_i^*\right) * (Q^*(\boldsymbol{h}, \boldsymbol{x} - baseline)$;
8:     Update the parameters of the DNN $\theta$: $\theta^{t+1} = \theta^t - \beta\left(\frac{\partial Loss}{\partial \theta}\right)$;
9: end while
```

C. Optimization Algorithm for Time Resource Allocation

After obtaining the offloading decision $\boldsymbol{x}$, we optimize time allocation to maximize the SCR of the network under a certain offloading decision.

Below, let $\mathcal{N}_0$ be the set of WDs' indices which conduct the computation locally and $\mathcal{N}_k$ be the set of WDs' indices which offload data to HAP_k. $\mathcal{N}_k = \phi$ if no WD is associated to HAP_k. So, $\mathcal{N}_0 + \mathcal{N}_1 + ... + \mathcal{N}_M = \mathcal{N}$. Let $\mathcal{M}'$ be the set of HAPs' indices which have at least one associated WD.

If $\mathcal{M}' = \phi$ (i.e., all WDs conduct the computation locally), then it is clear that the sub-problem is simple and the optimal value of a is 1. Below, we consider the case $\mathcal{M}' \neq \phi$. Without loss of generality, suppose that HAP_1 has at least one WD to offload data.

When we get $\boldsymbol{x}$, (9) can be expressed as:

$$(P3): Q^*(\boldsymbol{h},\boldsymbol{x}) =$$

$$\underset{\boldsymbol{\tau},\boldsymbol{a}}{\text{maximize}} \quad \sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M}h_{iq}\right)^{\frac{1}{3}}a^{\frac{1}{3}}+ \sum_{k\in\mathcal{M}'}\sum_{j\in\mathcal{N}_k}\epsilon\tau_{jk}\ln\left(1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}}{\tau_{jk}}a\right) \tag{16a}$$

$$\text{s.t.} \quad 0\le\tau_{jk}\le 1, \forall j\in\mathcal{N}_k, k\in\mathcal{M}', \tag{16b}$$

$$\sum_{j\in\mathcal{N}_k}\tau_{jk}+a\le 1, \forall k\in\mathcal{M}', \tag{16c}$$

$$0\le a\le 1. \tag{16d}$$

Lemma 1. *The optimization problem (16) is concave with respect to* $(a,\boldsymbol{\tau})$.

Proof. For the first term of (16a), i.e., the computing rate of local computing. It is obvious that exponential function $f(a)=a^{1/3}$ is concave function with respect to $(a,\boldsymbol{\tau})$. According to the fact $\eta_1\left(\sum_{q=1}^{M}h_{iq}\right)>0$ and the addition property of concavity, the first term is concave with respect to $(a,\boldsymbol{\tau})$. For the second term of (16a), i.e., the offloading rate of WDs. Since the logarithmic function $g(a)=\ln(1+a)$ is concave, according to chapter 3.2.6 in [40], its perspective function $\tau_{jk}g(a/\tau_{jk})=\tau_{jk}\ln(1+a/\tau_{jk})$ is concave with respect to (a,τ_{jk}). Since the value of $\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}$ and ϵ is greater than 0, they will not influence the concavity of the second term. According to the addition property of concavity, the second term of (16a) is concave respect to $(a,\boldsymbol{\tau})$. Considering that the constraints (16b), (16c), (16d) are affine, the optimization problem (16) is concave with respect to $(a,\boldsymbol{\tau})$. □

Since (P3) is concave, it can be solved using the available algorithms. However, they usually take much time to obtain the optimal solution. Hence, using the available algorithms are not efficient enough in the fast-fading environment. To efficiently solve problem (P3), we design an algorithm based on the Lagrangian duality method.

By introducing Lagrangian multipliers, Lagrangian of (16) can be expressed as:

$$L(a,\boldsymbol{\tau},\boldsymbol{\lambda}) = \sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M}h_{iq}\right)^{\frac{1}{3}}a^{\frac{1}{3}}+ \sum_{k\in\mathcal{M}'}\sum_{j\in\mathcal{N}_k}\epsilon\tau_{jk}\ln\left(1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}}{\tau_{jk}}a\right)+ \sum_{k\in\mathcal{M}'}\lambda_k\left(1-a-\sum\nolimits_{j\in\mathcal{N}_k}\tau_{jk}\right) \tag{17a}$$

$$\text{s.t.} \quad 0\le\tau_{jk}\le 1, \forall j\in\mathcal{N}_k, k\in\mathcal{M}', \tag{17b}$$

$$0\le a\le 1, \tag{17c}$$

where $\boldsymbol{\lambda}=\{\lambda_1,\lambda_2,\ldots,\lambda_M\}\succeq 0$ are Lagrange Multipliers associated with the constraints (16c).

Then, the dual problem is

$$\min_{\boldsymbol{\lambda}\succeq 0}\max_{a,\boldsymbol{\tau}} L(a,\boldsymbol{\tau},\boldsymbol{\lambda}). \tag{18}$$

Next we investigate the relationship between a^*,$\boldsymbol{\tau}^*$ and $\boldsymbol{\lambda}^*$ which are the optimal values of a,$\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$, respectively.

Utilizing the Karush-Kuhn-Tucker (KKT) optimality condition, we can get following equations:

$$\left.\frac{\partial L}{\partial\tau_{jk}}\right|_{\tau_{jk}=\tau_{jk}^*,a=a^*} = \epsilon\ln\left(1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}}{\tau_{jk}^*}a^*\right) - \frac{\epsilon\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}a^*{\tau_{jk}^*}^{-1}}{1+\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}a^*{\tau_{jk}^*}^{-1}}-\lambda_k^*=0, \quad \forall j\in\mathcal{N}_k, k\in\mathcal{M}', \tag{19}$$

$$\left.\frac{\partial L}{\partial a}\right|_{\tau_{jk}=\tau_{jk}^*,a=a^*} = \sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M}h_{iq}\right)^{\frac{1}{3}}\frac{1}{3}(a^*)^{-\frac{2}{3}} + \sum_{k\in\mathcal{M}'}\left[\sum_{j\in\mathcal{N}_k}\left(\epsilon\frac{\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}}{1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}}{\tau_{jk}^*}a^*}\right)-\lambda_k^*\right]=0, \tag{20}$$

$$1-a^*-\sum\nolimits_{j\in\mathcal{N}_k}\tau_{jk}^*=0, \ \forall k\in\mathcal{M}'. \tag{21}$$

In the following lemma, we show that a^* can be expressed as a function of $\lambda_k^*(\forall k\in\mathcal{M}')$ and τ_{jk}^* can be expressed as a function of a^* and λ_k^*.

Lemma 2. *The optimal* $\{a^*,\boldsymbol{\tau}^*,\boldsymbol{\lambda}^*\}$ *satisfy:*

$$\tau_{jk}^* = \eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}a^*\psi(\lambda_k^*), \ \forall j\in\mathcal{N}_k, k\in\mathcal{M}', \tag{22}$$

$$a^* = \frac{1}{1+\eta_2\sum_{j\in\mathcal{N}_k}h_{jk}\sum_{q=1}^{M}h_{jq}\psi(\lambda_k^*)}, \forall k\in\mathcal{M}', \tag{23}$$

where

$$\psi(\lambda_k^*) = \left[-\left(W\left(-\frac{1}{\exp\left(1+\frac{\lambda_k^*}{\epsilon}\right)}\right)\right)^{-1}-1\right]^{-1}, \tag{24}$$

and $W(\cdot)$ *is Lambert W function.*

Proof. From (19) we have the following equation:

$$\ln\left(1+\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}{\tau_{jk}^*}^{-1}a^*\right) = \left(1+\frac{\lambda_k^*}{\epsilon}\right)-\frac{1}{1+\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}{\tau_{jk}^*}^{-1}a^*}. \tag{25}$$

Taking natural exponential operation at both sides:

$$x\exp\left(\frac{1}{x}\right) = \exp\left(1+\frac{\lambda_k^*}{\epsilon}\right), \tag{26}$$

where $x=1+\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}a^*{\tau_{jk}^*}^{-1}$.

We further have

$$(-\frac{1}{x})(e^{-\frac{1}{x}}) = -\frac{1}{\exp\left(1+\frac{\lambda_k^*}{\epsilon}\right)}. \tag{27}$$

So we can solve the equation for τ_{jk}^* by:

$$\frac{1}{x} = -W\left(\frac{-1}{\exp\left(1+\frac{\lambda_k^*}{\epsilon}\right)}\right) = \left(1+\eta_2 h_{jk}\sum_{q=1}^{M} h_{jq} a^* {\tau_{jk}^*}^{-1}\right)^{-1} \tag{28}$$

and τ_{jk}^* can be expressed as:

$$\tau_{jk}^* = \eta_2 h_{jk}\sum_{q=1}^{M} h_{jq} a^* \psi(\lambda_k^*), \tag{29}$$

where $\psi\left(\lambda_k^*\right)$ is given above.

According to (21), $\forall k \in \mathcal{M}'$, we have

$$a^* = \frac{1}{1+\eta_2\sum_{j\in\mathcal{N}_k} h_{jk}\sum_{q=1}^{M} h_{jq}\psi(\lambda_k^*)}. \tag{30}$$

The proof is completed. □

Lemma 3. *The relationship between λ_1^* and $\lambda_k^*, \forall k \in \mathcal{M}' - \{1\}$ is:*

$$\lambda_k^* = \left(\ln\left(1+H_k\psi^{-1}(\lambda_1^*)\right) + \left(1+H_k\psi^{-1}(\lambda_1^*)\right)^{-1} - 1\right)\epsilon, \tag{31}$$

where $H_k \triangleq \frac{\sum_{i\in\mathcal{N}_k} h_{ik}\sum_{q=1}^{M} h_{iq}}{\sum_{j\in\mathcal{N}_1} h_{j1}\sum_{q=1}^{M} h_{jq}}$.

Proof. According to (23), we can get:

$$\begin{aligned} a^* &= \frac{1}{1+\eta_2\sum_{i\in\mathcal{N}_k} h_{ik}\sum_{q=1}^{M} h_{iq}\psi\left(\lambda_k^*\right)} \\ &= \frac{1}{1+\eta_2\sum_{j\in\mathcal{N}_1} h_{j1}\sum_{q=1}^{M} h_{iq}\psi\left(\lambda_1^*\right)}, \forall k \in \mathcal{M}' - \{1\}. \end{aligned} \tag{32}$$

Thus, we get the relationship between λ_1^* and $\lambda_k^*, \forall k \in \mathcal{M}' - \{1\}$ as follows:

$$\psi\left(\lambda_k^*\right) = \frac{\sum_{j\in\mathcal{N}_1} h_{j1}\sum_{q=1}^{M} h_{jq}}{\sum_{i\in\mathcal{N}_k} h_{ik}\sum_{q=1}^{M} h_{iq}}\psi\left(\lambda_1^*\right) = H_k^{-1}\psi\left(\lambda_1^*\right). \tag{33}$$

Substitute $\psi\left(\lambda_k^*\right) = \left[-\left(W\left(-\frac{1}{\exp\left(1+\frac{\lambda_k^*}{\epsilon}\right)}\right)\right)^{-1} - 1\right]^{-1}$ into the above equation, we can get:

$$\lambda_k^* = \left(\ln\left(1+H_k\psi^{-1}(\lambda_1^*)\right) + \left(1+H_k\psi^{-1}(\lambda_1^*)\right)^{-1} - 1\right)\epsilon. \tag{34}$$

The proof is completed. □

Lemma 4. *Lagrange multiplier λ_1^* satisfies:*

$$\begin{aligned} &\sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M} h_{ij}\right)^{\frac{1}{3}}\frac{1}{3}a^{*-\frac{2}{3}} \\ &+\sum_{k\in\mathcal{M}'-\{1\}}\left[\sum_{j\in\mathcal{N}_k}\left(\epsilon\frac{\eta_2 h_{jk}\sum_{q=1}^{M} h_{jq}}{1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M} h_{jq}}{\tau_{jk}^*}a^*}\right)\right. \\ &\left.-\left(\ln\left(H_k\psi^{-1}(\lambda_1^*)+1\right) + \left(H_k\psi^{-1}(\lambda_1^*)+1\right)^{-1} - 1\right)\epsilon\right] \\ &+\sum_{j\in\mathcal{N}_1}\left(\epsilon\frac{\eta_2 h_{j1}\sum_{q=1}^{M} h_{jq}}{1+\frac{\eta_2 h_{j1}\sum_{q=1}^{M} h_{jq}}{\tau_{j1}^*}a^*}\right) - \lambda_1^* = 0, \end{aligned} \tag{35}$$

where $H_k \triangleq \frac{\sum_{i\in\mathcal{N}_k} h_{ik}\sum_{q=1}^{M} h_{iq}}{\sum_{j\in\mathcal{N}_1} h_{j1}\sum_{q=1}^{M} h_{jq}}$, $\forall k \in \mathcal{M}' - \{1\}$*, and a^* is a function of λ_1^* (see (32)).*

Proof. Substitute (31) into (20), we get:

$$\begin{aligned} \frac{\partial L}{\partial a} &= L_a(\lambda_1^*) = \sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M} h_{iq}\right)^{\frac{1}{3}}\frac{1}{3}a^{*-\frac{2}{3}} \\ &+\sum_{k\in\mathcal{M}'-\{1\}}\left[\sum_{j\in\mathcal{N}_k}\left(\epsilon\frac{\eta_2 h_{jk}\sum_{q=1}^{M} h_{jq}}{1+\frac{\eta_2 h_{jk}\sum_{q=1}^{M} h_{jq}}{\tau_{jk}^*}a^*}\right)\right. \\ &\left.-\left(\ln\left(1+H_k\psi^{-1}(\lambda_1^*)\right) + \left(1+H_k\psi^{-1}(\lambda_1^*)\right)^{-1} - 1\right)\epsilon\right] \\ &+\sum_{j\in\mathcal{N}_1}\left(\epsilon\frac{\eta_2 h_{j1}\sum_{q=1}^{M} h_{jq}}{1+\frac{\eta_2 h_{j1}\sum_{q=1}^{M} h_{jq}}{\tau_{j1}^*}a^*}\right) - \lambda_1^* = 0. \end{aligned} \tag{36}$$

The proof is completed. □

Lemma 3 gives an equation (35) consisting only one variable λ_1^* as a can be expressed the function of λ_1^* (refer to (32)). In order to further obtain the value of λ_1^*, we analyze the monotonicity of (35).

Lemma 5. *$L_a(\lambda_1^*)$ is a monotonically decreasing function with respect to λ_1^*.*

Proof. Following (31), we first define:

$$F(x) \triangleq \left(\ln\left(1+H_k x\right) + \left(1+H_k x\right)^{-1} - 1\right)\epsilon. \tag{37}$$

Then the first derivative of $F(x)$ with respect to x is derived:

$$F'(x) = \frac{{H_k}^2 x}{\left(H_k x+1\right)^2}\epsilon. \tag{38}$$

From (38), we can know that $F'(x)$ is positive when x is positive and is negative when x is negative.

In addition, we have

$$
\begin{aligned}
L_a'\left(\lambda_1^*\right)= & \sum_{i\in\mathcal{N}_0}\eta_1\left(\sum_{q=1}^{M}h_{iq}\right)^{\frac{1}{3}}(-\frac{2}{9})a^{*-\frac{5}{3}}\frac{\partial a^*}{\partial\lambda_1^*} \\
& -\sum_{k\in\mathcal{M}'-\{1\}}\left[\sum_{j\in\mathcal{N}_k}\frac{\left[\left(\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}\right)^2\tau_{jk}^*\epsilon\right]}{\left(\tau_{jk}^*+\eta_2 h_{jk}\sum_{q=1}^{M}h_{jq}a^*\right)^2}\frac{\partial a^*}{\partial\lambda_1^*}\right] \\
& -\sum_{k\in\mathcal{M}'-\{1\}}\left[F'(\lambda_1^*)\left[\psi^{-1}(\lambda_1^*)\right]'\right]\epsilon \\
& +\sum_{j\in\mathcal{N}_1}\left[-\frac{\left[\left(\eta_2 h_{j1}\sum_{q=1}^{M}h_{jq}\right)^2\tau_{j1}^*\epsilon\right]}{\left(\tau_{j1}^*+\eta_2 h_{j1}\sum_{q=1}^{M}h_{jq}a^*\right)^2}\frac{\partial a^*}{\partial\lambda_1^*}\right]-1.
\end{aligned}
\tag{39}
$$

When λ_1^* is greater than 0, the value of $-\frac{1}{\exp\left(1+\frac{\lambda_1^*}{\epsilon}\right)}$ belongs to the interval $(-1/\mathrm{e},0)$, so the value of $W\left(-\frac{1}{\exp\left(1+\frac{\lambda_1^*}{\epsilon}\right)}\right)$ belongs to the interval $(-1,0)$. Then according to (24), we can get that the value of $\psi^{-1}(\lambda_1^*)$ is greater than 0, i.e., $\frac{\partial F(\psi^{-1}(\lambda_1^*))}{\partial\psi^{-1}(\lambda_1^*)}$ is greater than 0. On the other hand, $\psi^{-1}(\lambda_1^*)$ is a monotonically increasing function of λ_1^* when λ_1^* is greater than 0, and thus $\left[\psi^{-1}(\lambda_1^*)\right]'>0$.

From (23) we get that a^* is a monotonically decreasing function of $\psi(\lambda_1^*)$. Since $\psi(\lambda_1^*)$ is a monotonically decreasing function of λ_1^*, a^* is a monotonically increasing function of λ_1^* which means the value of $\frac{\partial a^*}{\partial\lambda_1^*}$ is greater than 0. Above all, since we can get that each term in the (39) is less than 0, $L_a\left(\lambda_1^*\right)$ is a monotonically decreasing function of λ_1^*.

The proof is completed. □

It should be further explained that the obtained $\boldsymbol{\tau}^*$ and a^* must satisfy the boundary constraint (17b) and (17c), respectively, which are not taken into account in the above analysis. Below we show that they are satisfied. When $\lambda_1^*=0$, $W\left(-\frac{1}{\exp\left(1+\frac{\lambda_1^*}{\epsilon}\right)}\right)=-1$. According to (24), $\psi^{-1}(\lambda_1^*)=0$. So when $\lambda_1^*=0$, substituting $\psi^{-1}(\lambda_1^*)=0$ into $L_a(\lambda_1^*)$, we can know that $L_a(\lambda_1^*)>0$. Considering the decreasing monotonicity of $L_a(\lambda_1^*)$, the solution found by bisection method must locates on $(0,+\infty)$, i.e., $\lambda_1^*>0$. According to (24), when $\lambda_1^*>0$, $\psi(\lambda_1^*)>0$. By using (23) in Lemma 1, we can find that a^* obtained by the proposed algorithm belongs to $(0,1)$. By using (31) in Lemma 2, we can find that λ_k^* obtained by the proposed algorithm belongs to $(0,+\infty)$. By using (22) in Lemma 1, the τ_{jk}^* derived by the proposed method must satisfy $\tau_{jk}^*>0$. Additionally, $\tau_{jk}^*<1$ holds. Otherwise, a^* obtained in (30) by using (21) definitely does not belong to $(0,1)$, which contradicts with the previous conclusion that a^* obtained by the proposed algorithm belongs to $(0,1)$. Hence, all the solutions derived by the proposed optimization algorithm, including a^*, $\boldsymbol{\lambda}^*$ and $\boldsymbol{\tau}^*$, can satisfy the constraints, which means that the proposed optimization method can obtain the optimal solutions for problem (P3).

Algorithm 2 The Lagrangian duality based optimization algorithm for solving problem (P3)

```
Require: channel state $\boldsymbol{h}$, offloading decision $\boldsymbol{x}$;
Ensure: The offloading duration for every ED $\tau$, WPT duration $a$;
1:  Initialize error tolerance $\gamma = 0.0001$
2:  Initialize $\lambda_1^{max} = 100000000$, $\lambda_1^{min} = 0$, $\lambda_1 = \left(\lambda_1^{max}+\lambda_1^{min}\right)/2$
3:  Calculate $L_a(\lambda_1)$ by substituting $\lambda_1$ into (35);
4:  while $\lambda_1^{max} - \lambda_1^{min} > \gamma$ do
5:      if $L_a(\lambda_1) < 0$ then
6:          $\lambda_1^{max} = \lambda_1$ and $\lambda_1 = \left(\lambda_1^{max}+\lambda_1^{min}\right)/2$
7:      end if
8:      if $L_a(\lambda_1) > 0$ then
9:          $\lambda_1^{min} = \lambda_1$ and $\lambda_1 = \left(\lambda_1^{max}+\lambda_1^{min}\right)/2$
10:     end if
11: end while
12: By substituting $\lambda_1$ into (31), obtain the value of $\lambda_k^*, \forall k \in \mathcal{M}$;
13: By substituting $\lambda_1^*$ into (23), obtain the value of optimal WPT duration $a^*$;
14: By substituting $\lambda_k^*$ and $a^*$ into (22), obtain the optimal value of $\boldsymbol{\tau}^*$;
```

Finally, our algorithm for sub-problem (P3) is as follows (refer to Algorithm 2). According to Lemma 3 and Lemma 4, we can use the bisection search method to find the value of λ_1^* according (35). By using (31), the value of $\lambda_k^*, \forall k\in\mathcal{M}$, can be derived. According to (22) and (23) in Lemma 1, the optimal $\boldsymbol{\tau}^*$ and a^* can be obtained.

D. Computation Complexity Analysis

In algorithm 1, offloading decision is output by DNN and its followed normalization and sampling, which totally have the complexity of $O(NM)$. In algorithm 2, the complexity of bisection method is $O(\log_2(\frac{\lambda_1^{max}-\lambda_1^{min}}{\gamma}))$ which is a constant smaller than 100. In lines 12-14, the complexity for computing $\lambda_k(k\in\mathcal{M}'-\{1\})$, a and $\tau_{jk}(j\in\mathcal{N}_k, k\in\mathcal{M}')$ is $O(M)$, $O(NM)$ and $O(NM)$, respectively. Thus, the overall complexity is $O(NM)$.

V. Numerical Results

This section evaluates the performance of our proposed algorithm. The parameters are set as follows: P = 3 Watts, μ = 0.8, B = 2 MHz, N_0 = 10^{-10} Watts, v_u = 1.1, k = 10^{-26} and ϕ = 100. The locations of all WDs and HAPs are randomly within an area of 12m*12m. For WD_i and HAP_j with the distance d_{ij}, the large-scale fading component constant $\alpha_{ij}=A_d\left(\frac{3\cdot 10^8}{4\pi f_c d_{ij}}\right)^{d_e}$, where the antenna gain A_d = 4.11, the carrier frequency f_c = 915 MHz, and the path loss exponent d_e = 2.3. The small-scale Rayleigh fading follows an exponential distribution with unit mean. The used neural network has 5 fully connected layers with the number of neurons 350, 450,

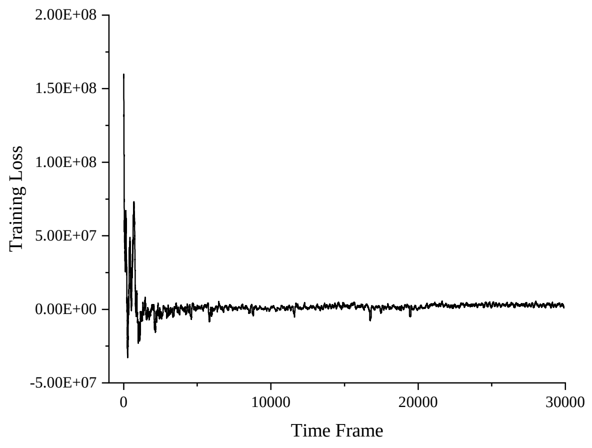


Fig. 4. The training loss for algorithm when N = 5, M = 3 and β=0.00095.

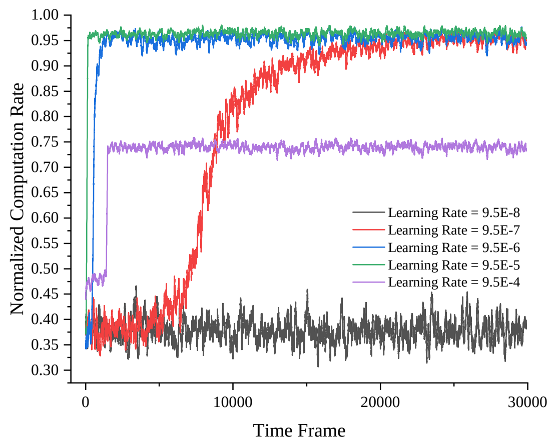


Fig. 5. Performance of proposed algorithm under different learning rates.

600, 700 and 800, respectively. The activation function is Tanh. The simulations were conducted on a laptop with a 3.2 GHz AMD Processor and 16 GB RAM.

In order to effectively evaluate the proposed DRL-based offloading algorithm, we take the exhaustive search (ES) algorithm to obtain the optimal offloading decision. Define the normalized computation rate $\overline{Q}(\boldsymbol{h},\boldsymbol{x})$ as:

$$\overline{Q}(\boldsymbol{h},\boldsymbol{x}) = \frac{Q^*(\boldsymbol{h},\boldsymbol{x})}{Q^*_{exhaustive}(\boldsymbol{h},\boldsymbol{x'})}. \tag{40}$$

Here, $Q^*(\boldsymbol{h},\boldsymbol{x})$ is the computation rate obtained by the proposed algorithm and $Q^*_{exhaustive}(\boldsymbol{h},\boldsymbol{x'})$ is the computation rate obtained by the ES algorithm. The ES algorithm refers to traversing all the possible offloading decisions and outputting the maximum computation rate that the network can achieve. By comparing with this maximum computation rate, we can evaluate the performance of the proposed algorithm intuitively. We consider the scenario with 3 HAPs and 5 WDs for the sake of the rather long running time of the ES algorithm.

First, we investigate the proposed algorithm's convergence speed. Fig. 4 shows that in the early stage of neural network training, the value of loss is large. With the training of the neural network, the loss value gradually becomes smaller, which means that the network gradually converges. When the training of the neural network reaches the later stage, that is, after the time slot reaches 8000, we find that the loss value is close to 0.

In Fig. 5, we compare the performance of four different learning rates of the algorithm. As we can see, when the learning rate is large, say $9.5*10^{-4}$, the algorithm can only converge to the normalized computation rate of about 0.75 and fluctuate around this value. This is because a higher learning rate will make the update of the neural network parameters too aggressive, and then fall into a sub-optimal solution. In contrast to this, low Learning rate, say $9.5*10^{-8}$, may cause the algorithm's performance not been improved significantly with observable time slots. This is because when the learning rate is too small, the update of the neural network parameters

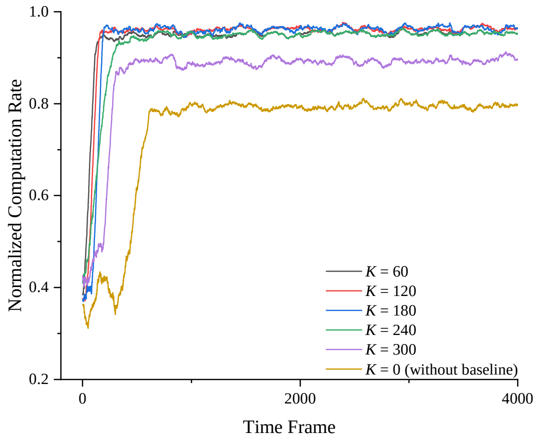


Fig. 6. Comparison of algorithm performance with different baselines

may be too conservative to jump out of the local optimum. When setting the learning rate as $9.5*10^{-7}$, although the neural network can converge to near optimum, the convergence speed is slow because of the small step for every training. Therefore, we take $9.5*10^{-5}$ as the learning rate for our algorithm. We can see that under this learning rate, the algorithm can not only converge faster, but also perform well, which can reach about 96% of the maximal computation rate.

In Fig. 6, we investigate the influence of baseline value K. Note that we set the average computation rate of K time slots as the baseline. When $K = 0$, i.e., there is no baseline, the normalized computation rate is about 0.8, which is much lower than the other curves. This is because in the absence of a baseline to adjust the reward value, the environment will give a positive reward value for all offloading decisions, so that the parameters of the policy network cannot be updated in the direction of the optimal policy, i.e., the direction with the largest reward. From Fig. 6, We can conclude that the baseline applied in our algorithm is effective from the

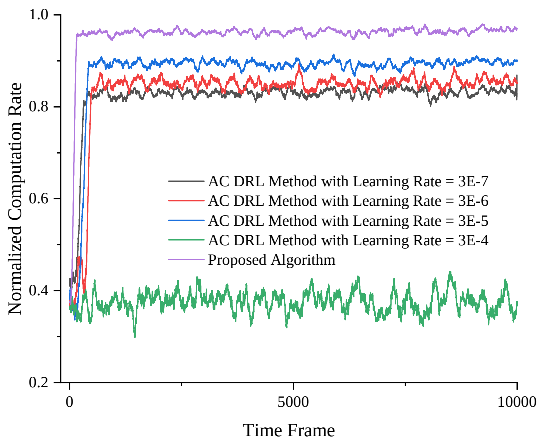


Fig. 7. Comparison of the proposed algorithm and AC algorithm with different learning rates

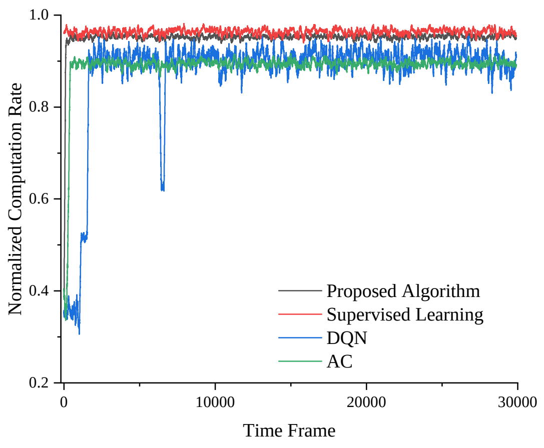


Fig. 8. Comparison of the proposed algorithm with different algorithms.

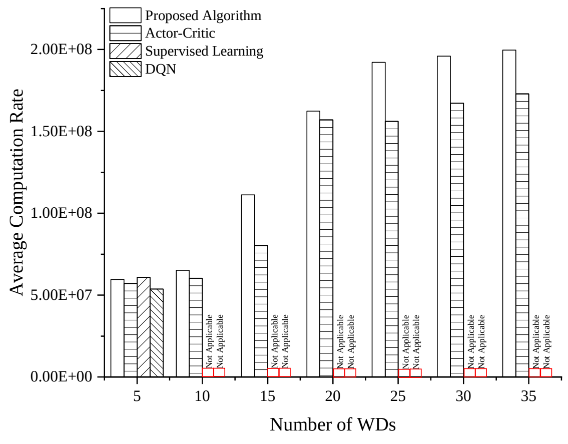


Fig. 9. Comparison of different offloading algorithms with 3 HAPs and increasing number of WDs.

perspective of normalized computation rate and convergence speed. According to the simulation results, we set $K = 120$.

To measure the performance of our algorithm, we mainly compare it with the following three algorithms:

- Supervised Learning [22]: The neural network is trained offline by 10000 training samples with the optimal samples. These samples are consisted of channel states and corresponding optimal actions which are obtained by the ES method.
- DQN [21]: The DQN algorithm outputs the state-action value corresponding to each offloading decision combination based on the neural network, and selects the offloading decision combination with the highest state-action value for execution.
- AC algorithm: The AC algorithm evaluates the action output by the actor network through the critic network, so that the action output by the actor network can maximize the output value of the criticism network in the current state.

Fig. 7 shows the normalized computation rate of the proposed algorithm and the most common policy-based DRL algorithm, AC algorithm with the same learning rate and structure of policy network as ours. As shown in the figure, although we set different learning rates of critic network, the performance of AC is always lower that the proposed algorithm. This is because the evaluation of the action-state value by the critic network is not accurate enough compared with the reward value given by the environment. Therefore, there is a certain gap between AC and our proposed algorithm. Additionally, from the perspective of algorithm convergence speed, our algorithm can converge faster than AC because the latter needs to maintain two networks while our algorithm maintains single policy network, and the AC algorithm can only converge when both networks converge. Fig. 7 indicates that the proposed algorithm is efficient for solving the multi-HAPs offloading decision problem in WPT empowered MEC.

In Fig. 8, we compare the proposed algorithm with different available algorithms. It can be seen in the figure that the neural network trained by the supervised learning method can achieve the highest normalized computation rate. However, since its training process requires the samples obtained by the ES method, and the time consumption of label acquisition will increase exponentially with the number of WDs and HAPs, it is not applicable to large scale network scenarios. The average performance of DQN algorithm is close to that of AC algorithm, but the curve fluctuates wildly. For the DQN, the action space is $(M+1)^N$, which increases exponentially with the number of WDs. Hence, DQN is not applicable for the large scale network scenarios. For the AC algorithm here, the actor network adopts the same independent sampling with the proposed algorithm in this paper, so it is able to avoid exponentially increasing action space. The proposed method can achieve almost the same performance as supervised learning without need of providing the optimal samples.

In Fig. 9 and Fig. 10, we evaluate the performance of different algorithms in terms of the average SCR over 500 time slots. Considering that the action space increases exponentially

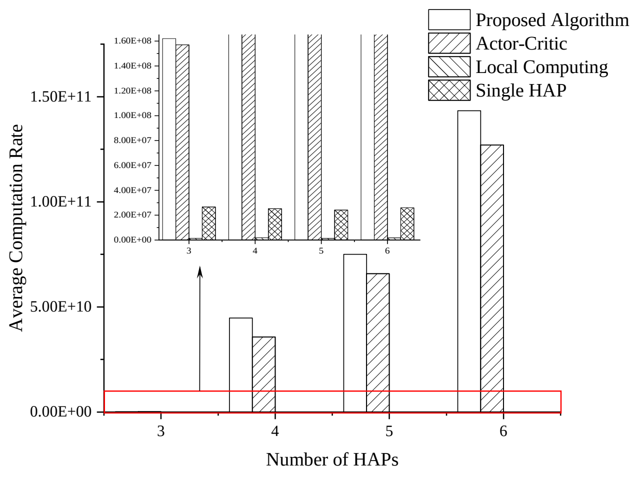


Fig. 10. Comparison of different offloading algorithms with 20 WDs and increasing number of HAPs.

with the number of WDs, supervised learning and DQN are not applicable for the scenario with 10 or more WDs, and thus we only apply them to the scenario of 3 HAPs and 5 WDs.

In Fig. 9, it can be seen that when all WDs choose to conduct the computation by themselves, the sum of their computation rates is the lowest. This is because in our scenario, the computation ability of HAP is much greater than that of WDs. When all WDs are allowed to offload data to only one of the three HAPs, the achieved average SCR is much smaller than that of our algorithm. This is because when all WDs choose to offload data to the same HAP, due to the limited time resource, all the WDs need to share the time slot of selected HAP. As a result, each WD cannot be allocated enough time to offload data to achieve a large enough data offloading rate. This can also be verified from the phenomenon that the average SCR almost does not increase when the number of WDs increases if all WDs are allowed to offload data to only one of three HAPs. Obviously, providing multiple HAPs is an efficient way to improve the computation rate for WPT empowered MEC. We further compare the performance of our algorithm with AC. It's obvious that although the computation rate of the two algorithms is significantly improved as WDs' number increases, there is still a certain gap between our proposed algorithms and AC. This verifies that our algorithm still has advantages over traditional DRL algorithms such as AC when the number of WDs increases. In addition, we found that if the number of WDs increases to a certain number, the computation rate of the network will no longer have a clear growth with the increase of WDs. The reason is that the total system time resource of HAPs is fixed and when the number of WDs increases, less time is allocated to each WD for transmission.

In Fig. 10, we evaluate the average computation rates under different numbers of HAPs with 20 WDs. It can be seen that when the number of HAPs increases, the average computing rate of the network increases significantly. This is because the total system time resource of HAPs increases linearly with the number of HAPs. In addition, we can see that compared to the AC algorithm, the performance enhancement obtained by our proposed algorithm increases as the number of HAPs increases.

TABLE III
COMPARISON OF CPU RUNNING TIME BETWEEN THE ES ALGORITHM AND OURS UNDER THE 3-HAPS CASES.

Number of WDs	ES algorithm	Proposed algorithm
5	3.1359s	0.0143s
6	14.2609s	0.0167s
7	64.6999s	0.0189s
8	289.7720s	0.0204s

TABLE IV
COMPARISON OF CPU RUNNING TIME BETWEEN THE AC ALGORITHM AND OURS UNDER THE 3-HAPS CASES.

Number of WDs	AC algorithm	Proposed algorithm
5	0.0199s	0.0143s
10	0.0373s	0.0248s
15	0.0507s	0.0352s
20	0.0665s	0.0455s
25	0.0870s	0.0527s
30	0.1056s	0.0681s
35	0.1176s	0.0826s

TABLE V
COMPARISON OF CPU RUNNING TIME BETWEEN THE AC ALGORITHM AND OURS WITH INCREASE OF HAPS UNDER THE 20-WDS CASES.

Number of HAPs	AC algorithm	Proposed algorithm
3	0.0665s	0.0455s
4	0.0889s	0.0580s
5	0.1065s	0.0759s
6	0.1209s	0.0833s

Table III shows the average CPU running time of the proposed algorithm and the ES algorithm under different numbers of WDs. We can see that as the number of WDs increases, the time consumed by the ES algorithm will greatly increase. This is because the time complexity of ES algorithm is exponential, and increasing the number of WDs causes the time of executing the algorithm to grow exponentially. For our algorithm, the operations inside neural network and Algorithm 2 have the computation complexity roughly linear with the number of WDs.

In Table IV and Table V, we investigate the average CPU running time of our algorithm and the widely used AC algorithm under the increasing number of WDs and the increasing number of HAPs. The average CPU running time required by the AC algorithm is about 1.4 times longer than our algorithm. The reason is that there are two networks in the AC algorithm. Compared with our algorithm, the AC network needs to spend more time to compute the value of the state-value function by using its critic network.

VI. Conclusion

This paper considered the computation rate maximization problem in a WPT empowered MEC with multiple WDs and multiple HAPs. The binary offloading mode was considered. We proposed an online DRL-based algorithm to obtain the near-optimal offloading decision based on the channel gains and designed a Lagrangian duality based algorithm to optimally allocate time resource under the given offloading decision. The proposed DRL-based algorithm converges fast and can achieve about 96 percent of the maximal computation rate with low computational complexity. Compared with the common actor-critic algorithm, it has advantages in computation rate, convergence speed and running time. This work can be further explored from two aspects: on one hand, the DRL algorithm can be improved to achieve better performance. On the other hand, more complex scenarios with consideration of feedback, edge processing delay and long-term optimization will bring new challenges which are worth being studied.

References

[1] A. S. Kumar, L. Zhao, and X. Fernando, "Multi-agent deep reinforcement learning-empowered channel allocation in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1726-1736, 2022.

[2] H. Guo, X. Zhou, J. Liu, and Y. Zhang, "Vehicular intelligence in 6g: Networking, communications, and computing," *Vehicular Communications*, vol. 33, pp. 100399, 2022.

[3] S. Zhang, S. Kong, K. Chi, and L. Huang, "Energy management for secure transmission in wireless powered communication networks," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1171-1181, 2022.

[4] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, "A survey on space-air-ground-sea integrated network security in 6G," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 53-87, 2022.

[5] Z. Chen, K. Chi, K. Zheng, Y. Li, X. Liu, "Common throughput maximization in wireless powered communication networks with non-orthogonal multiple access," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7692-7706, 2020.

[6] K. Zheng, X. Liu, B. Wang, H. Zheng, K. Chi, Y. Yao, "Throughput maximization of wireless-powered communication networks: An energy threshold approach," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1292-1306, 2021.

[7] S. Gong, C. Xing, S. Wang, L. Zhao, and J. An, "Throughput maximization for intelligent reflecting surface aided MIMO WPCNs with different DL/UL reflection patterns," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2706-2724, 2021.

[8] Y. Zeng, B. Clerckx, and R. Zhang, "Communications and signals design for wireless power transmission," *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 2264-2290, 2017.

[9] A. Biason and M. Zorzi, "Battery-powered devices in WPCNs," *IEEE Transactions on Communications*, vol. 65, no. 1, pp. 216-229, 2017.

[10] J. Moon, H. Lee, C. Song, and I. Lee, "Secrecy performance optimization for wireless powered communication networks with an energy harvesting jammer," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 764-774, 2017.

[11] B. Lyu, P. Ramezani, D. T. Hoang, S. Gong, Z. Yang, and A. Jamalipour, "Optimized energy and information relaying in self-sustainable IRS-empowered WPCN," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 619-633, 2021.

[12] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519-7537, 2021.

[13] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN enhanced vehicular edge computing networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 217-228, 2020.

[14] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of NOMA-based multiaccess edge computing," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3186-3203, 2022.

[15] A. S. Kumar, L. Zhao, and X. Fernando, "Mobility aware channel allocation for 5G vehicular networks using multi-agent reinforcement learning," in *Proc. IEEE ICC*, 2021, pp. 1-6.

[16] H. A. Shah, L. Zhao, and I.-M. Kim, "Joint network control and resource allocation for space-terrestrial integrated network through hierarchal deep actor-critic reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4943-4954, 2021.

[17] W. Chen, G. Shen, K. Chi, S. Zhang, and X. Chen, "DRL based partial offloading for maximizing sum computation rate of FDMA-based wireless powered mobile edge computing," *Computer Networks*, vol. 214, pp. 109158, 2022.

[18] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581-2593, 2020.

[19] Y. Yu, Y. Yan, S. Li, Z. Li, and D. Wu. "Task delay minimization in wireless powered mobile edge computing networks: A deep reinforcement learning approach," in *Proc. WCSP*, 2021, pp. 1-6.

[20] S. Zhang, H. Gu, K. Chi, L. Huang, K. Yu, and S. Mumtaz, "DRL-based partial offloading for maximizing sum computation rate of wireless powered mobile edge computing network," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10934-10948, 2022.

[21] Q. Chen, Z. Kuang, and L. Zhao,"Multiuser computation offloading and resource allocation for cloud–edge heterogeneous network,"*IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3799-3811, 2022.

[22] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: a multi-task learning approach,"*IEEE Transactions on Mobile Computing*, vol. 20, no. 9, pp. 2745-2762, 2021.

[23] B. He, S. Bi, H. Xing, and X. Lin, "Collaborative computation offloading in wireless powered mobile-edge computing systems," in *Proc. IEEE GLOBECOM Workshops*, 2019, pp. 1-7.

[24] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927-1941, 2018.

[25] X. Zhou, L. Huang, T. Ye, and W. Sun, "Computation bits maximization in UAV-assisted MEC networks with fairness constraint," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20997-21009, 2022.

[26] L. Shi, Y. Ye, X. Chu, and G. Lu, "Computation energy efficiency maximization for a NOMA-based WPT-MEC network," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10731-10744, 2021.

[27] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784-1797, 2018.

[28] F. Wang, H. Xing, and J. Xu, "Real-time resource allocation for wireless powered multiuser mobile edge computing with energy and task causality," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 7140-7155, 2020.

[29] R. Malik and M. Vu, "On-request wireless charging and partial computation offloading in multi-access edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6665-6679, 2021.

[30] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177-4190, 2018.

[31] M. Zeng, R. Du, V. Fodor, and C. Fischione, "Computation rate maximization for wireless powered mobile edge computing with NOMA," in *Proc. IEEE WoWMoM*, 2019, pp. 1-9.

[32] P. X. Nguyen, D.-H. Tran, O. Onireti, P. T. Tin, S. Q. Nguyen, S. Chatzinotas, and H. Vincent Poor, "Backscatter-assisted data offloading in OFDMA-based wireless-powered mobile edge computing for IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9233-9243, 2021.

[33] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3170-3184, 2020.

[34] C. Wang, W. Lu, S. Peng, Y. Qu, G. Wang, and S. Yu, "Modeling on energy-efficiency computation offloading using probabilistic action generating," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20681-20692, 2022.

[35] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1320-1323, 2019.

[36] J. Liu, K. Xiong, D. W. K. Ng, P. Fan, Z. Zhong, and K. B. Letaief, "Max-min energy balance in wireless-powered hierarchical fog-cloud computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7064-7080, 2020.

[37] K. Zheng, G. Jiang, X. Liu, K. Chi, X. Yao, J. Liu, "DRL-based offloading for computation delay minimization in wireless-powered multi-access edge computing," *IEEE Transactions on Communications*, vol. 71, no. 3, pp. 1755-1770, 2023.

[38] N. H. Motlagh, E.Lagerspetz, P. Nurmi, X. Li, S. Varjonen, J. Mineraud, M. Siekkinen, A. Rebeiro-Hargrave, T. Hussein, T. Petaja, M. Kulmala, and S. Tarkoma, "Toward massive scale air quality monitoring,"*IEEE Communications Magazine*, vol. 58, no. 2, pp. 54-59, 2020.

[39] S. Lee and D. Choi,"Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources,"*IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 488-497, 2022.

[40] S. Boyd and L. Vandenberghe,"Convex Optimization," *Cambridge, U.K.: Cambridge Univ. Press*, 2004.