Online Learning for Distributed Computation Offloading in Wireless Powered Mobile Edge Computing Networks

Xiaojie Wang, Zhaolong Ning, Lei Guo, Song Guo, *Fellow, IEEE*, Xinbo Gao, and Guoyin Wang, *Senior Member, IEEE*

Abstract—A novel paradigm named Wireless Powered Mobile Edge Computing (WP-MEC) emerges recently, which integrates Mobile Edge Computing (MEC) and Wireless Power Transfer (WPT) technologies. It enables mobile clients to both extend their computing capacities by task offloading, and charge from edge servers via energy transmission. Existing studies generally focus on the centralized design of task scheduling and energy charging in WP-MEC networks. To meet the decentralization requirement of the near-coming 6G network, we propose an online learning algorithm for computation offloading in WP-MEC networks with a distributed execution manner. Specifically, we first define the delay minimization problem by considering task deadline and energy constraints. Then, we transform it into a primal-dual optimization problem based on the Bellman equation. After that, we design a novel neural model that learns both offloading and time division decisions in each time slot to solve the formulated optimization problem. To train and execute the designed algorithm distributivity, we form multiple learning models decentralized on edge servers and they work coordinately to achieve parameter synchronization. At last, both theoretical and performance analyses show that the designed algorithm has significant advantages in comparison with other representative schemes.

Index Terms—Wireless powered mobile edge computing, delay minimization, distributed execution, online learning

1 Introduction

The development of the Sixth Generation of Mobile Communications (6G) and the Internet of Things (IoT) technology enables ultra-low latency and computation-intensive applications, such as virtual reality and three-dimensional videos, to become blooming. However, mobile clients are always limited on computing capabilities and battery lifetime due to the constraint of production costs and technologies [1]. Recently, Mobile Edge Computing (MEC) and Wireless Power Transfer (WPT) are regarded as two promising technologies to relieve the conflict between application requirements and limited terminal resources [2]. With MEC, the computing capabilities of mobile clients can be virtually enhanced by offloading computation-intensive tasks to edge servers. WPT allows mobile clients to charge their batteries over the air, where the Radio Frequency (RF) signals are transmitted from edge servers to mobile clients. After those signals are received, they can be converted into electricity for devices.

Currently, a new paradigm, named Wireless Powered Mobile Edge Computing (WP-MEC), is introduced by the integration of MEC and WPT technologies. Mobile clients in WP-MEC networks can not only offload their computation tasks to edge servers with the purpose of reducing task competition delay, but also charge from edge servers to sustain their energy supply [3]. Generally, due to the characteristic of half-duplex, offloading and charging processes for mobile clients can not be run simultaneously. Thus, an inevitable issue exists in WP-MEC networks: *how to make reasonable time division and resource allocation for both task offloading and energy charging to maximize the utilities of mobile clients*. Time division and resource allocation have mutual impacts on each other. On one hand, the time division can influence the amount of harvested energy of mobile clients, and task scheduling decisions are dependent on harvested energy. On the other hand, task scheduling results in the previous time slot inversely impact the time division of the subsequent time slot.

- *Xiaojie Wang, Zhaolong Ning, and Lei Guo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: {xiaojie.kara.wang, z.ning}@ieee.org, guolei@cqupt.edu.cn.*
- *Song Guo is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China. E-mail: song.guo@polyu.edu.hk.*
- *Xinbo Gao is with the Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: gaoxb@cqupt.edu.cn.*
- *Guoyin Wang is with the Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: wanggy@cqupt.edu.cn.*

Manuscript received 27 Aug. 2021; revised 27 Oct. 2021; accepted 14 Nov. 2021. Date of publication 22 Nov. 2021; date of current version 9 Dec. 2021.
This work was supported by the National Natural Science Foundation of China under Grants 61872310, 61971084, 62025105, and 62001073, the funding from Hong Kong RGC Research Impact Fund (RIF) with the Project No. R5060-19, General Research Fund (GRF) with the Project No. 152221/19E and 15220320/20E, Shenzhen Science and Technology Innovation Commission (R2020A045), the Natural Science Foundation of Chongqing under Grants cstc2019jcyj-cxttX0002, cstc2021ycjh-bgzxm0013, cstc2021ycjh-bgzxm0039 and cstc2021jcyj-msxmX0031, the Chongqing Municipal Education Commission (key cooperation Project No. HZ2021008 and CXQT21019), the Support Program for Overseas Students to Return to China for Entrepreneurship and Innovation under Grants cx2021003 and cx2021053.
(Corresponding authors: Zhaolong Ning and Song Guo.)
Recommended for acceptance by Y. Yang.
Digital Object Identifier no. 10.1109/TPDS.2021.3129618

Existing studies always formulate the computation offloading issue in WP-MEC networks by Mixed Integer Programming (MIP). Optimization methods, including convex optimization [4] and approximation method [5], are utilized to solve such issues. However, most of those algorithms are executed in a centralized manner, and require full network information to make scheduling decisions. According to the report in [6], near 100% geographical wireless coverage will be provided by 6G, where distributed management is needed to meet use cases with millisecond information update rates. Followed that, distributed computation scheduling algorithms in WP-MEC networks are called for to meet the coming 6G network. However, it is difficult to construct such a distributed system due to the following challenges:

First, there are many studies investigating the scheduling algorithm for computation offloading with single edge server in WP-MEC networks. It has been revealed that even the optimization problem of computation offloading with single edge server in a centralized manner is NP-hard [4], not to mention multiple edge servers with distributed scheduling. When there is one edge server, it merely needs to determine its own energy transmission and task offloading periods based on the system status. When there are several edge servers and a set of mobile clients in the system, those servers not only need to take the requirements of devices into consideration, but also their energy transmission time should be unified. This is because concurrent energy pulses can increase received energy levels of mobile clients in a short period [7]. However, it is difficult to reach a consensus on energy transmission time for all edge servers in the distributed environment.

Second, to schedule dynamic offloading requirements of mobile clients in an online manner for minimizing the long-term task completion delay, it is promising to design a Deep Reinforcement Learning (DRL) based scheduling algorithm with the advantage of its strong learning ability and fast convergence speed. Nevertheless, edge servers acting as learning agents need to make resource scheduling decisions independently, since they do not know policies of other agents beforehand. Meanwhile, their different decisions may result in multiple offloading choices for some clients while none for others, heavily impacting users' utilities. Thus, resource allocation policies of edge servers should be carefully designed.

To solve the above challenges, we propose an online learning algorithm for distributed computation offloading in wireless powered mobile edge computing networks, named OLD-COP, with the purpose of minimizing the long-term average task completion delay for mobile clients. To the best of our knowledge, this work is *the first to provide a distributed design of computation offloading with multiple edge servers in WP-MEC networks so far*. Our contributions can be summarized as follows:

- We first formulate the distributed computation offloading issue as a delay minimization problem by considering task deadline and energy constraints. To make it solvable, we transform it into a primal-dual optimization problem based on Bellman equation. Specifically, states, actions, state transition and rewards are formally defined based on the formulated Markov Decision Process (MDP).
- To solve the defined optimization problem, we design a novel neural model, including three learning components, i.e., policy, value and dual networks, which learns both time division and offloading decisions in each time slot. To train and execute the designed learning model distributively, we reform the whole learning model into multiple copies, which can be decentralized on edge servers and coordinated to achieve parameter synchronization.
- Comprehensive theoretical analyses are conducted both from slot-based and long-term perspectives. We prove that the upper bound of the achieved system performance exists, and the algorithm convergence can be guaranteed with certain conditions even in the distributed environment.
- Performance evaluation based on the city map of Manhattan is conducted to demonstrate the effectiveness of the designed algorithm in terms of average task completion delay, average task completion ratio and so on.

The rest of this paper is structured as follows: in Section 2, we review the related work; we present the system model and define the studied problem in Section 3; in Section 4, we formulate the task scheduling problem based on MDP; we design the online learning algorithm for distributed computation offloading in WP-MEC networks in Section 5, followed by performance evaluation in Section 6; finally, we conclude this work in Section 7.

2 Related Work

We review the state-of-the-art for computation offloading and energy harvesting in WP-MEC networks in this section. Currently, the studies on WP-MEC can be categorized into the following two kinds based on the number of involved edge servers.

The majority of researchers focus on the first category, i.e., computation scheduling with one edge server. The authors in [2] investigate binary offloading by jointly optimizing system time allocation and individual task offloading mode with the purpose of maximizing the sum computation rate of mobile clients. An online resource allocation algorithm based on sliding-window is proposed in [8], to minimize energy consumption in a system-wide area. The remote task execution, energy beamforming at the access point, task offloading and computing for mobile clients are jointly optimized. An online resource allocation strategy is proposed in [9], with the purpose of maximizing the fairness among different mobile users, where Lyapunov optimization technology is leveraged. The trade-off between response delay and energy efficiency in WP-MEC networks is investigated in [3]. Random task arrivals and channel conditions are considered, and a stochastic optimization problem is formulated. Similar with [9], Lyapunov optimization technology is also utilized.

With the purpose of maximizing the profit of the operator in multi-user WP-MEC networks, an iterative algorithm based on Lagrangian dual method is proposed in [10] by optimizing power allocation of mobile clients and data sizes of offloaded tasks. Both binary and partial computation offloading modes in WP-MEC networks are designed in [11],

TABLE 1
Main Notations

Notation	Description
ρ_i	The probability for mobile client i to generate a task in each time slot;
e_i^t	The task generated by mobile client i in time slot t;
ζ_i^t	The size of task e_i^t;
c_i^t	The required CPU cycle of task e_i^t;
$d_i^{t,max}$	The deadline of task e_i^t;
f_i^l	The CPU frequency of mobile client i;
f_j^s	The CPU frequency of edge server j;
τ^t	The duration of energy harvesting period in time slot t;
$\mathbb{T}$	The duration of each time slot;
$d_{ij}^{t,l}$	The transmission delay of task e_i^t from mobile client i to edge server j in time slot t;
Υ_{ij}^t	The transmission rate from mobile client i to edge server j in time slot t;
$d_i^{t,o}$	The computation delay of task e_i^t by local computing;
$d_{ij}^{t,o}$	The computation delay of task e_i^t with remote computing by edge server j;
$d_i^{t,w}$	The waiting delay of task e_i^t;
α_i^t	A binary value denoting whether task e_i^t is processed locally;
β_i^{tj}	A binary value denoting whether task e_i^t is offloaded to server j for computation;
d_i^t	The completion delay of task e_i^t;
$\mathbb{E}_i^{t,h}$	The harvested energy by client i in time slot t;
$\mathbb{E}_{ij}^{t,l}$	The consumed energy of task e_i^t transmitted from mobile client i to edge server j;
$\mathbb{E}_{ij}^{t,o}$	The consumed energy of task e_i^t by local computation;
$\mathbb{E}_i^{t,c}$	Total consumed energy of transmitting and processing task e_i^t;
s^t	The network state in time slot t;
a^t	The joint action of edge servers in time slot t;
a_j^t	The action of edge server j in time slot t;
r^t	The immediate reward received by servers after taking action a^t at state s^t;
$V^*(s^t)$	The optimal state value of the formulated MDP;
π^*	The optimal policy learned by edge servers;
Q_j^t	The processing sequence for all clients on server j in time slot t;
$H(\pi, s^t)$	The entropy regularizer for the value function;
$\omega_j, \phi_j, \theta_j$	Optimization parameters of the dual network, the value network and the policy network, respectively;
$\rho_j(s^t, a^t)$	The dual parameter for optimization problem P2.

with the purpose of maximizing the computation efficiency. Alternative optimization and iterative algorithms are designed by jointly optimizing offloading power and time, computing frequency as well as energy harvesting time. However, the above studies are based on one edge server along with multiple mobile clients, and merely centralized scheduling policies are designed.

In addition, many studies leverage DRL technology in WP-MEC networks, such as [12], [13], [14]. Aiming to minimize the long-term system cost, an online resource management scheme based on DRL is proposed in [12]. Both task offloading and server provision can be learned by a designed post-decision state based algorithm. In [13], the unmanned aerial vehicle is utilized as the server to charge mobile clients and harvest data from them. An on-board deep Q-network is designed for the minimization of data packet loss of mobile clients, where charging and data collection decisions are learned to derive the optimal solution. A DRL-based online framework is proposed to learn binary offloading decisions according to time-varying wireless channel conditions in [14]. Although the above studies are efficient in the environment with one edge server and multiple mobile clients, few ones focus on distributed computation offloading with multiple edge servers and mobile clients.

The other category is scheduling with more than one edge server. There are a limited number of studies related to this topic due to its complexity. The authors in [15] consider computation offloading scheduling with multiple edge servers in WP-MEC networks, aiming to maximize the computation completion ratio. The formulated optimization problem is proved to be NP-hard, and a centralized approximation algorithm is designed to improve the system performance. A joint resource allocation and task offloading algorithm is proposed in [16], where a Mixed Interger Nonlinear Problem (MINLP) is formulated to maximize task offloading gains of mobile clients. Although they focus on computation scheduling with multiple edge servers, their algorithms can not be conducted in a distributed manner.

To realize distributed computation offloading in WP-MEC networks with online scheduling, we design an algorithm based on DRL technology. A novel neural model is constructed, learning both offloading decisions and time division strategies in each time slot. Furthermore, the whole learning model can be reformed into single copies, which can be decentralized on edge servers and coordinated to achieve parameter synchronization. *To the best of our knowledge, we are the first to investigate distributed resource scheduling in WP-MEC networks with the coexisting of multiple servers and mobile clients.*

3 System Model and Problem Definition

In this section, we first present the system model. Then, we define the optimization problem. The main notations are listed in Table 1.

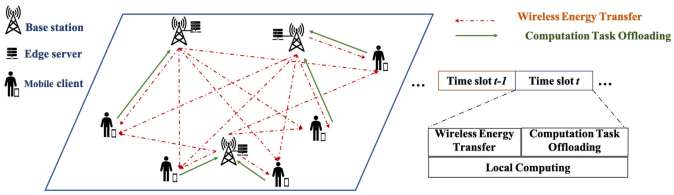


Fig. 1. The illustrative system model.

3.1 System Model

The objective of this paper is to efficiently allocate network resources to minimize the long-term expected completion delay of tasks generated by mobile clients. As shown in Fig. 1, we consider a 2D area, where M edge servers, denoted by $\mathbb{M} = \{1, ..., j, ..., M\}$, and N mobile clients, represented by $\mathbb{N} = \{1, ..., i, ..., N\}$ coexist. Each edge server is installed on one Base Station (BS), and BSs can communicate with mobile clients based on Orthogonal Frequency Division Multiple Access (OFDMA) technology. The subchannel matching algorithm in [17] can be applied in our system. Mobile clients randomly move around in the area, and can harvest energy through WPT from edge servers. They can also offload their tasks to edge servers for computation, and tasks are processed in sequence on each server. The time horizon is divided into a set of time slots with equal length $\mathbb{T}$. For simplicity and without loss of generality, we consider that mobile client i generates a task in each time slot with probability ρ_i, $i \in \mathbb{N}$. The task generated by mobile client i in time slot t can be represented by $e_i^t = \{\zeta_i^t, c_i^t, d_i^{t,max}\}$, where ζ_i^t is the task size; symbol c_i^t is the required CPU cycle for task computation, and can be computed by $c_i^t = \varsigma\zeta_i^t$, where ς represents the required CPU cycles of each bit. Symbol $d_i^{t,max}$ is the task deadline. We assume that the task is atomic, and can not be split into smaller ones. Thus, binary offloading decisions can be made.

Mobile client i is with CPU frequency f_i^l, and edge server j is with f_j^s. The mobile client leverages half-duplex transmission, thus WPT and task offloading can not be simultaneously conducted [18]. Similar to [19], we design a harvest-then-offload algorithm for task scheduling. That is, mobile clients first harvest energy from servers, and then offload their tasks to remote servers. We utilize τ^t ($\tau^t < \mathbb{T}$) to represent the duration of the energy harvesting period, and $\mathbb{T} - \tau^t$ denotes that of the task scheduling period in time slot t. In the following two subsections, we specify the detailed delay and energy computation models.

3.2 Task Completion Delay

For a task generated by a mobile client, it can either be offloaded to edge servers or processed locally. For the delay to complete the task, three components are included:

3.2.1 Transmission Delay

If task e_i^t is offloaded to edge server j, the transmission delay from mobile client i to edge server j in time slot t can be computed by $d_{ij}^{t,l} = \zeta_i^t / \Upsilon_{ij}^t$, where Υ_{ij}^t is the transmission rate between mobile client i and edge server j, and can be computed based on Shannon formula, i.e.,

$$\Upsilon_{ij}^t = \mathbb{B}\log_2\left(1 + \frac{P_{ij}h_{ij}^t}{\sigma_j^2}\right). \tag{1}$$

Symbol $\mathbb{B}$ is the channel bandwidth, and σ_j^2 is the noise power of server j. Variable P_{ij} is the transmission power from mobile client i to edge server j, and h_{ij}^t is the wireless channel gain. If task e_i^t is processed locally, there is no transmission delay.

3.2.2 Computation Delay

If task e_i^t is processed locally, the computation delay can be obtained by $d_i^{t,o} = c_i^t / f_i^l$. If task e_i^t is offloaded to edge server j, the computation delay can be obtained by $d_{ij}^{t,o} = c_i^t / f_j^s$.

3.2.3 Waiting Delay

Since tasks are processed sequentially on edge servers, waiting delay can be caused even after energy harvesting. Herein, waiting delay refers to the duration for a task from the time it is generated until that it starts to be processed locally or by the edge server, i.e.,

$$d_i^{t,w} = \tau^{t'} + \Delta t' + (t' - t)\mathbb{T}, \quad t' \geq t, \tag{2}$$

where t' represents the time slot in which task e_i^t is processed. Variable $\Delta t'$ is the waiting delay in the processing queue of the targeted edge server in time slot t' after energy harvesting.

We utilize binary value α_i^t to denote whether task e_i^t is processed locally. When it is processed on client i, variable $\alpha_i^t = 1$, and vice versa. Binary value β_{ij}^t denotes whether task e_i^t is offloaded to server j in time slot t. Consequently, we can obtain the total completion delay for task e_i^t by:

$$d_i^t = \alpha_i^t d_i^{t,o} + (1 - \alpha_i^t) \sum_{j=1}^{M} \beta_{ij}^t (d_{ij}^{t,l} + d_{ij}^{t,o}) + d_i^{t,w}. \tag{3}$$

Since the sizes of computation results are generally small, the delay for returning them back to mobile clients can be neglected [20].

3.3 Energy Harvesting and Consumption

In the scenario of energy harvesting with a set of transmitters and multiple receivers, all those energy transmitters on edge servers send energy pulse simultaneously to mobile clients [7]. These energy waves can be combined either destructively or constructively. In the destructive mode, the harvested energy from the combined energy wave is lower than that of the individual energy wave. Conversely, the harvested energy from the combined energy wave is higher than that of the individual energy wave in the constructively mode. Since transmission frequencies of edge servers are fixed in our system, edge servers can be regarded as a virtual transmitter similar to [5] and [21]. Then, all edge servers transmit RF power to mobile clients during the first τ^t duration of each time slot.

In this situation, the harvested energy by each mobile client is related to the transmission frequency of those transmitters and the distance between the transmitter and the receiver. Since our focus is to reasonably make time division

and task offloading decisions, similar with [5] and [11], we adopt a linear energy harvesting model in our system for simplicity and without loss of generally. Thus, the harvested energy by client i in time slot t can be expressed by $\mathbb{E}_i^{t,h} = \mu \mathcal{P}_i \hat{h}_i^t \tau^t$, where μ is a ratio between 0 and 1, referring to the energy conversion efficiency for mobile clients, and $\hat{h}_i^t$ is the downlink channel gain from the virtual transmitter to client i. The received transmission power from the virtual transmitter by client i can be regarded by $\mathcal{P}_i$.

After energy harvesting, the generated tasks of mobile clients can be scheduled, and the consumed energy for task scheduling includes the following two aspects:

3.3.1 Transmission Energy

The consumed energy for task e_i^t transmitted from mobile client i to edge server j can be computed by $\mathbb{E}_{ij}^{t,l} = P_{ij} d_{ij}^{t,l}$.

3.3.2 Local Computation Energy

According to [15], if the task is processed locally on the client, its consumed energy for computation can be obtained by $\mathbb{E}_i^{t,o} = \kappa_i (f_i^l)^3 d_i^{t,o}$, where κ_i is the coefficient related to energy efficiency on mobile client i. As a result, total consumed energy $\mathbb{E}_i^{t,c}$ can be expressed by:

$$\mathbb{E}_i^{t,c} = \alpha_i^t \mathbb{E}_i^{t,o} + (1 - \alpha_i^t) \mathbb{E}_{ij}^{t,l}. \tag{4}$$

3.4 Problem Definition

In time slot t, there are mainly two periods, i.e., one is for energy harvesting, and the other is for task scheduling and processing. In the period of energy harvesting, all edge servers transmit RF power to mobile clients, and all clients charge for energy. After that, the generated tasks are scheduled for computation. Mobile clients decide whether and where to offload their tasks based on the network condition. An offloading request generated by the mobile client can be broadcast to reachable edge servers. After receiving the offloading requests from mobile clients, edge servers learn to decide the scheduling sequences for those offloaded tasks and send the process sequences to mobile clients. Then, tasks can be scheduled by mobile clients based on the received results. To satisfy the users' computation requirements, we define the considered problem as a delay-minimization problem, presented as follows:

Objective:

Our objective is to minimize the average task completion delay over all time slots, i.e.,

$$\text{P1:} \quad \min \lim_{t \to \infty} \frac{1}{Nt} \sum_{v=0}^{t} \sum_{i=1}^{N} d_i^v, \tag{5}$$

$$\text{s.t.} \quad d_i^t \le d_i^{t,max}, \tag{5a}$$

$$\mathbb{E}_i^{t,c} \le \mathbb{E}_i^{t,h} + \mathbb{E}_i^{t-1,r} - \mathbb{E}_i^t. \tag{5.b}$$

Input:

Edge server set $\mathbb{M}$, mobile client set $\mathbb{N}$, task set $\{e_i^t\}_{i \in \mathbb{N}}$, CPU frequency sets $\{f_i^l\}_{i \in \mathbb{N}}$ and $\{f_j^s\}_{j \in \mathbb{M}}$, channel bandwidth $\mathbb{B}$, wireless channel gain $\{h_{ij}^t\}_{i \in \mathbb{N}, j \in \mathbb{M}}$, noise power $\{\sigma_j^2\}_{j \in \mathbb{M}}$, transmission power $\{P_{ij}\}_{i \in \mathbb{N}, j \in \mathbb{M}}$, energy conversion efficiency μ, received power $\{\mathcal{P}_i\}_{i \in \mathbb{N}}$, and energy efficiency coefficient κ_i.

Output:

1) Length τ^t of energy harvesting period in time slot t;

2) Offloading decisions α_i^t and β_{ij}^t for task e_i^t, $i \in \mathbb{N}$, $j \in \mathbb{M}$ and $t \in \{1, 2, 3....\}$.

Constraints:

For client j, its generated tasks should be scheduled and processed before deadline $d_i^{t,max}$, i.e., Equation (5a). If equation (5a) can not be satisfied, the task is discarded. Then, the consumed energy for completing task e_i^t should satisfy Equation (5.b). Symbol $\mathbb{E}_i^{t-1,r}$ is the remaining energy of mobile client i in time slot $t-1$, and $\mathbb{E}_i^t$ is the consumed energy before processing task e_i^t in time slot t. That is to say, the consumed energy of client i in time slot t can not exceed the remaining energy in time slot t. If Equation (5.b) can not be satisfied, the task can not be scheduled in time slot t.

It is rather challenging to achieve the above objective even in the centralized network environment. This is because not only task scheduling decisions but also time division strategies for energy harvesting and task scheduling affect the system performance, making the delay minimization problem be NP-hard [15], not to mention solving the optimization problem in a distributed manner. On one hand, edge servers should reach a consensus on the energy harvesting period based on the decentralized environment. On the other hand, the task offloading requirement of mobile clients should be scheduled to the most suitable edge servers for delay minimization, especially based on local observations of edge servers. Thus, we intend to find policy π that can be executed in a distributed manner to achieve the objective, while satisfying constraints of task deadline and device residual energy.

4 MDP Optimization Problem Formulation

In this section, we model the defined delay minimization problem as an MDP, which can be represented by tuple $(S, A, R, \mathbb{P}, \gamma)$. Symbol S is the state space, and A is the action space. Symbol R represents the reward, and $\mathbb{P}$ is the state transition probability. Symbol $\gamma \in (0, 1)$ is the discount factor. Policy π is utilized to select actions based on each state. In the following, we provide the detailed introduction for the formation of elements in the considered MDP.

4.1 States

The state of an MDP can be represented by $S \triangleq \{s^t = (S_m, S_c)\}$, $t \in \{0, 1, 2, ...\}$, where two components are included in each state:

1) S_m denotes the states of mobile clients, where $S_m = \{(x_i^t, y_i^t, \mathbb{Z}_i^t, C_i^t, \Delta d_i^{t,min}, \Delta d_i^{t,max})\}_{i \in \mathbb{N}}$. Variables x_i^t and y_i^t are the coordinates of mobile client i in time slot t; symbol $\mathbb{Z}_i^t$ is the total data amount of local tasks for client i in time slot t, and C_i^t is the total required CPU cycles for computing tasks of client i in time slot t; symbol Δd_i^t is the remaining life cycles of tasks for client i in time slot t. Correspondingly, symbols $\Delta d_i^{t,min}$ and $\Delta d_i^{t,max}$ are the minimum and maximum left life cycles of tasks for client i in time slot t,

respectively. Thus, we can obtain total data amount $\mathbb{Z}_i^t = \sum_{v=0}^{t} (\xi_i^v)_{\Delta d_i^v > 0}$, and total required CPU cycle $C_i^t = \sum_{v=0}^{t} (c_i^v)_{\Delta d_i^v > 0}$, where Δd_i^v is the left life cycle of task e_i^v. We define function $\Gamma(\mathbb{X}) = (\mathbb{X})_{\Delta d_i^v > 0}$. When $\Delta d_i^v > 0$ holds, $\Gamma(\mathbb{X}) = \mathbb{X}$. Otherwise, $\Gamma(\mathbb{X}) = 0$.

2) The information related to communication is included in S_c, where $S_c = \{(P_{ij}, \Upsilon_{ij}^t)\}_{i \in \mathbb{N}, j \in \mathbb{M}}$.

4.2 Actions

The action space can be represented by $A \triangleq \{a^t = (\tau^t, \alpha_i^t, \beta_{ij}^t)\}_{i \in \mathbb{N}, j \in \mathbb{M}}$. That is, not only the energy harvesting period, but also the task scheduling result should be learned.

4.3 State Transition

The state transition probability distribution is denoted by $\mathbb{P} : S \times A \times S \rightarrow [0, 1]$, and $\rho_0 : S \rightarrow [0, 1]$ is the distribution of initial state s^0. State s^t can be transferred into s^{t+1} by taking action a^t based on probability $\mathbb{P}(s^{t+1}|s^t, a^t)$.

4.4 Rewards

The immediate reward received by servers after taking action a^t at state s^t can be represented by $r^t : S \times A \rightarrow \mathbb{R}$. Since our objective is to minimize the average task completion delay, we define $r^t = -\sum_{v=0}^{t} \sum_{i=1}^{N} \rho_i \left(d_i^v\right)_{\Delta d_i^v > 0}$.

4.5 Optimization Problem Formulation

We need to find policy π that can minimize the average task completion delay from a long-term perspective. In other words, we prepare to maximize the accumulative reward defined in the last subsection. Then, our optimization problem can be formulated by:

$$\text{P2:} \quad V^*(s^t) = \max_{a^t} [r^t + \gamma E_{s^{t+1}|s^t, a^t}[V(s^{t+1})]], \tag{6}$$

$$\text{s.t.} \quad \text{Constraints (5a) and (5b)},$$

where $V^*(s^t)$ is the optimal state value of the formulated MDP. Thus, optimal policy π^* can be founded by $\pi^* = \arg\max_{a^t} [r^t + \gamma E_{s^{t+1}|s^t, a^t}[V^*(s^{t+1})]]$. However, the above formulated optimization problem can not be solved directly by applying DRL-based approaches, such as traditional gradient based policies. The reasons are as follows:

First, although we merely need to determine the action for each state in the formulated MDP, the elements in the action contain integer variables while the output of the neural network is possibilities related to actions, making the actual action hard to determine. In addition, the elements in the defined actions are coupled with each other, and the change of one element has heavy impacts on others. For example, duration τ^t of energy harvesting period affects task scheduling decisions α_i^t and β_{ij}^t. Actions can not be directly derived from a neural network model. Second, the corresponding constraints make the optimization problem rather complex. The delay and energy conditions should be both satisfied, and need to be transformed into a solvable form. Third, in our considered decentralized environment, rewards are affected by actions of multiple servers. However, there is no centralized management to train policies, and servers need to train their own policies independently.

5 Online Learning for Distributed Computation Offloading

To solve the formulated optimization problem P2, we propose an online learning algorithm which can be trained and executed in a distributed manner. We first transfer problems P2 into P3 in Section 5.1, which can be solved by each edge server independently with their trained policies. After that, we present the whole online algorithm in Section 5.2, where three neural networks are specified on initialization, action execution, batch data collection and network training. At last, we provide a comprehensive algorithm analysis in Section 5.3 s.

5.1 Problem Transformation

In this subsection, we first present action, reward and value function reformation, and then process the constraints in the original optimization problem. At last, we formulate the solvable optimization problem for each server.

5.1.1 Action Reformation

In our considered multi-server environment, it is difficult to resolve problem P2 directly, since each edge server chooses actions independently in a decentralized manner. Thus, we first redefine the action of each server, which can formulate their joint actions for the formulated MDP.

From Section 4.2, we can observe that the defined original action includes three elements: energy harvesting period τ^t, task offloading decision α_i^t and task scheduling decision β_{ij}^t. For the first element, all servers should reach a consensus on it, due to the half-duplex character of mobile clients. For the remaining elements, servers make their own decisions without the awareness of others' decisions. Because the three elements are coupled together, we can not solve them simultaneously. In addition, it is difficult for the learning agents on edge servers to make suitable offloading decisions for all clients. Herein, we consider that all clients send offloading requests for all their tasks, based on which learning agents learn their policies. Then, we provide the following definition:

Definition 1 (Estimated task processing sequence). *It is defined by the processing sequence of all offloaded tasks of mobile clients, when they are all processed by edge server j in time slot t. It can be represented by $\mathbb{Q}_j^t = \{\mathbb{Q}_{ji}^t\}_{i=1}^N$, where $\mathbb{Q}_{ji}^t$ is the processing sequence for offloaded tasks of mobile client i, and can be computed by $\mathbb{Q}_{ji}^t = \{q_{ij}^{t'}, e_i^{t'}, \iota_{ij}^{t'}\}_{t'=0}^t$. Symbol $q_{ij}^{t'}$ is the processing sequence number of task $e_i^{t'}$, and $\iota_{ij}^{t'}$ is the estimated completion time of task $e_i^{t'}$.*

Following that, we utilize a_j^t to represent the action for edge server j in time slot t, and define $a_j^t = (\tau_j^t, \mathbb{Q}_j^t)$, $j \in \mathbb{M}$, where τ_j^t is the duration of energy harvesting period learned by edge server j. The reasons for utilizing sequence $\mathbb{Q}_{ji}$ instead of task scheduling decision β_{ij}^t are: 1) task scheduling decisions should be made based on the full network state, whereas servers are not aware of others' policy, resulting in partial observation; 2) their unawareness in the decentralized environment may cause multiple offloading selections for some clients but no choice for others. Thus, to balance the offloading choices among different clients, we

consider that each server broadcasts the processing sequence for all clients in each time slot. Based on all received sequences, mobile clients can always make proper offloading decision α_i^t and choose optimal edge server β_{ij}^t, depending on their own energy consumption and delay constraints.

5.1.2 Constraint Relaxation and Reward Reformation

Since edge server j chooses action a_j^t independently, its received reward is different from the whole reward. To make the reward of each server be consistent with the whole reward, we define the reward of edge server j by:

$$r_j^t = -\sum_{v=0}^{t} \sum_{i\in\mathbb{N},\alpha_i^t=0,\beta_{ij}^t=1} \rho_i\big(d_i^v\big)_{\Delta d_i^v>0} - \frac{1}{M}\sum_{v=0}^{t} \sum_{i\in\mathbb{N},\alpha_i^t=1} \rho_i\big(d_i^v\big)_{\Delta d_i^v>0}. \tag{7}$$

Thus, we can obtain $r^t = \sum_{j=1}^{M} r_j^t$. In this way, each server can train their own policies locally, without the need of knowing policies of others.

For the constraints in problem P2, they are difficult to meet during the execution process of the neural network. Therefore, we add penalty l_j^t to reward r_j^t when edge server j has expired tasks. Then, the updated reward becomes $\hat{r}_j^t = r_j^t - l_j^t$, where penalty l_j^t can be computed by $l_j^t = \lambda_j^t n^t$. Symbol λ_j^t is an adjustment parameter, and n^t is the total number of expired tasks for all mobile clients in time slot t. Then, reward r^t can be represented by $r^t \triangleq \sum_{j=1}^{M} \hat{r}_j^t$, and the prove can be found in the following theorem:

Theorem 1 (Consistency of the reward function). *Reward function $\hat{r}_j^t$ of edge server j is consistent with the whole system reward r^t.*

The proof can be found in Appendix A of Supplementary File, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2021.3129618.

5.1.3 Solvable Optimization Problem Formulation

For DRL-based algorithms, the balance between exploration and exploitation is significant to help the agent to learn good policies. With the purpose of encouraging exploration and avoiding early convergence to a suboptimal result, entropy regularization is utilized to form the soft bellman optimality by augmenting the standard value function with a discounted entropy regularizer [22]. Thus, the value function in our work can be represented by:

$$V_\xi(s^t) = \max_{\pi(s^t,\cdot)} \big(E_{a^t\sim\pi(s^t,\cdot)}\big(r^t + \gamma E_{s^{t+1}|s^t,a^t} V_\xi(s^{t+1})\big) + \xi H(\pi, s^t)\big), \tag{8}$$

where parameter ξ is a control variable that scales the degree of regularization and entropy $H(\pi, s^t) = -\sum_{a^t\in A}\pi(s^t, a^t)\log\pi(s^t, a^t)$. When $\xi = 0$, the above equation degrades to the standard value function as shown in Equation (6). According to Proposition 3 in [23], when optimal pair (V_ξ^*, π_ξ^*) is the unique pair of (V, π) for Equation (8), the following equation is satisfied:

$$V(s^t) = r^t + \gamma E_{s^{t+1}|s^t,a^t}[V(s^{t+1})] - \xi\log\pi(s^t, a^t). \tag{9}$$

Inspired by Equation (9), a natural way to find the optimal pair of (V^*, π^*) is to resolve the following problem:

$$\min_{V,\pi} E_{s^t,a^t}(r^t + \gamma E_{s^{t+1}|s^t,a^t}[V(s^{t+1})] - \xi\log\pi(s^t, a^t) - V(s^t))^2. \tag{10}$$

Although the above optimization problem encourages exploration for the training process, it is still hard to solve. This is because its inner conditional expectation requires two independent sampling of s^{t+1}, which is difficult to obtain in practice. Consequently, a primal-dual form of optimization problem in Equation (10) has been introduced in [23], which can be expressed by:

$$\min_{V,\pi}\max_{\rho} E_{s^t,a^t,s^{t+1}}\Big[\big(\delta\big(s^t, a^t, s^{t+1}\big) - V(s^t)\big)^2\Big] - \eta E_{s^t,a^t,s^{t+1}}\Big[\big(\delta\big(s^t, a^t, s^{t+1}\big) - \rho\big(s^t, a^t\big)\big)^2\Big], \tag{11}$$

where parameter $\eta \in [0, 1]$ is a control variable to balance the trade-off between the two parts of Equation (11), and $\rho(s^t, a^t)$ is a global dual variable. Function $\delta(s^t, a^t, s^{t+1})$ can be computed by:

$$\delta\big(s^t, a^t, s^{t+1}\big) = r^t + \gamma V(s^{t+1}) - \xi\log\pi(s^t, a^t). \tag{12}$$

In our considered system, edge servers make decisions independently, thus policy $\pi(s^t, a^t)$ can be factorized by:

$$\pi(s^t, a^t) = \prod_{j=1}^{M} \pi_j(s^t, a_j^t). \tag{13}$$

In addition, we make $r^t \triangleq \sum_{j=1}^{M} \hat{r}_j^t$ to relax the constraints in problem P2. Then, function $\delta(s^t, a^t, s^{t+1})$ can be reformed by:

$$\begin{aligned}\delta\big(s^t, a^t, s^{t+1}\big) &= \sum_{j=1}^{M} \hat{r}_j^t + \gamma V(s^{t+1}) - \xi\log\prod_{j=1}^{M}\pi_j(s^t, a_j^t) \\ &= \sum_{j=1}^{M}\Big(\hat{r}_j^t + \frac{\gamma}{M}V(s^{t+1}) - \xi\log\pi_j(s^t, a_j^t)\Big).\end{aligned} \tag{14}$$

We define $\delta_j\big(s^t, a_j^t, s^{t+1}\big) = \hat{r}_j^t + \frac{\gamma}{M}V(s^{t+1}) - \xi\log\pi_j(s^t, a_j^t)$, and $\delta(s^t, a^t, s^{t+1}) = \sum_{j=1}^{M}\delta_j\big(s^t, a_j^t, s^{t+1}\big)$. Following that, the optimization objective in problem P2 can be transformed into:

$$\min_{V,\{\pi_j\}_{j=1}^{j=M}}\max_{\rho} E_{s^t,\left\{a_j^t\right\}_{j=1}^{M},s^{t+1}}\left[\left(\sum_{j=1}^{M}\delta_j\Big(s^t, a_j^t, s^{t+1}\Big) - V(s^t)\right)^2\right] - \eta E_{s^t,\left\{a_j^t\right\}_{j=1}^{M},s^{t+1}}\left[\left(\sum_{j=1}^{M}\delta_j\Big(s^t, a_j^t, s^{t+1}\Big) - \rho\big(s^t, a^t\big)\right)^2\right]. \tag{15}$$

We can observe that parameters π and a^t have been transformed into parametric forms. To make the optimization problem in Equation (15) solvable in a fully distributed way, parameters V and ρ also need to be parameterized into M

elements corresponding to M edge servers. As a result, we consider that edge server j keeps a local copy of global parameters $V(s^t)$ and $\rho(s^t, a^t)$ by $V_j(s^t)$ and $\rho_j\left(s^t, a_j^t\right)$, respectively. That is, there is a consensus update process for edge servers to make their local copies the same with that in [24], i.e., $V_1(s^t) = ... = V_j(s^t) = V_{j+1}(s^t) = ... = V_M(s^t)$. Then, edge server j intends to independently solve the following optimization problem:

$$\begin{aligned}\text{P3:}\quad & \min_{V_j,\pi_j}\max_{\rho_j} E_{s^t,a_j^t,s^{t+1}}\left[\left(\delta_j\left(s^t,a_j^t,s^{t+1}\right)-\frac{1}{M}V_j(s^t)\right)^2\right]\\ & 0-\eta E_{s^t,a_j^t,s^{t+1}}\left[\left(\delta_j\left(s^t,a_j^t,s^{t+1}\right)-\frac{1}{M}\rho_j\left(s^t,a_j^t\right)\right)^2\right],\\ \text{s.t.}\quad & V_1(s^t) = ... = V_j(s^t) = V_{j+1}(s^t) = ... = V_M(s^t). \end{aligned} \tag{16}$$

5.2 The Whole Algorithm

In our considered system, mobile clients send computation offloading requests to edge servers. After receiving those requests, edge servers can schedule them by solving problem P3. That is, each edge server acts as a learning agent, and makes scheduling decisions for mobile clients without knowing the policies of others. Then, edge servers broadcast the scheduling results (task processing sequences) to mobile clients, and those clients choose the best scheduling strategies among the results of those servers to minimize their task completion delays. To solve problem P3, each edge server maintains and trains its constructed neural networks locally. The training process of edge server j can be specified as follows:

5.2.1 Neural Network Initialization

For edge server j, it needs to construct three neural networks corresponding to three optimization parameters ρ_j, V_j and π_j in problem P3, i.e., dual network with optimization parameter ω_j, value network with ϕ_j and policy network with θ_j. The dual network is utilized to train dual parameter $\rho_j(s^t, a_j^t)$, and the policy network trains agent policy $\pi_j(s^t, a_j^t)$. The value network is utilized to predict state value $V_j(s^t)$. Thus, policy $\pi_j(s^t, a_j^t)$ is equivalent to $\pi_{\theta_j}(s^t, a_j^t)$, and state value $V_j(s^t)$ is equivalent to $V_{\phi_j}(s^t)$, similar with dual parameters $\rho_j(s^t, a^t)$ and $\rho_{\omega_j}(s^t, a^t)$. Then, problem P3 is also equivalent to the following problem:

$$\begin{aligned}\text{P4:}\quad & \min_{\phi_j,\theta_j}\max_{\omega_j} L(\phi_j,\theta_j,\omega_j)\\ & = E_{s^t,a_j^t,s^{t+1}}\left[\left(\delta_j\left(s^t,a_j^t,s^{t+1}\right)-\frac{1}{M}V_j(s^t)\right)^2\right]\\ & \quad -\eta E_{s^t,a_j^t,s^{t+1}}\left[\left(\delta_j\left(s^t,a_j^t,s^{t+1}\right)-\frac{1}{M}\rho_j\left(s^t,a_j^t\right)\right)^2\right],\\ \text{s.t.}\quad & \phi_1 = ... = \phi_j = \phi_{j+1} = ... = \phi_M,\\ & \omega_1 = ... = \omega_j = \omega_{j+1} = ... = \omega_M. \end{aligned} \tag{17}$$

5.2.2 Action Execution

In time slot t, edge server j chooses action a_j^t based on its policy π_j. First, edge server j inputs system state s^t into the policy network, and its output is $\pi_j(s^t, a_j^t)$. Based on the output possibilities, edge server j converts them into processing sequence $\mathbb{Q}_j^t$ of mobile clients. Then, the mobile client makes the proper offloading decision and selects the proper edge server for computation offloading. That is, for task e_i^t, client i compares the estimated delay for completing e_i^t based on processing sequence $\{\mathbb{Q}_{ji}^t\}_{j=1}^M$, and estimated local processing delay $\hat{d}_i^{t,o}$. The above process can be formed as an optimization problem for client i by:

$$\begin{aligned}\text{P5:}\quad & \hat{j} = \arg\min\left\{\iota_{ij}^t\right\}_{j=1}^M,\\ \text{s.t.}\quad & \text{inequation (5b)}, \end{aligned} \tag{18}$$

and

$$\begin{aligned}\text{P6:}\quad & D_i = \arg\min\left\{\iota_{i\hat{j}}^t, \hat{d}_i^{t,o}\right\},\\ \text{s.t.}\quad & \text{inequation (5b)}. \end{aligned} \tag{19}$$

The above two problems can be easily solved by the method of exhaustion. If $D_i = \iota_{i\hat{j}}^t$, then $\alpha_i = 0$ and $\beta_{ij} = \beta_{i\hat{j}}$, and server $\hat{j}$ can be selected for processing task e_i^t. If $D_i = \hat{d}_i^{t,o}$, then $\alpha_i = 1$ and local computing is selected.

5.2.3 Batch Data Collection

In order to learn good scheduling results by training policy, value and dual networks, we collect decision trajectories of those edge servers in mini-batches. The system state, predicted state value and outputs of dual and policy networks are recorded together in batches for each edge server. Then, edge servers train their local neural networks by minimizing their losses based on the collected batch data.

5.2.4 Model Training

Solving problem P4 is not easy by DRL, since both dual and primal parameters need to be approximated based on neural network training. Thus, three steps are utilized here to train the three neural networks. First, we solve the inner dual problem, i.e.,

$$\begin{aligned}\text{P7:}\quad & \max_{\omega_j} L(\phi_j,\theta_j,\omega_j),\\ \text{s.t.}\quad & \omega_1 = ... = \omega_j = \omega_{j+1} = ... = \omega_M. \end{aligned} \tag{20}$$

To get the learned value of ω_j, we need to train the dual network. Since the first part in the expression of $L(\phi_j, \theta_j, \omega_j)$ can be regarded as a constant in the training process of the dual network, we set the corresponding loss function by:

$$\begin{aligned} L(\omega_j) = & -\eta E_{s^t,a_j^t,s^{t+1}}[(\delta_j(s^t,a_j^t,s^{t+1})\\ & -\frac{1}{M}\rho_{\omega_j}(s^t,a^t))^2]. \end{aligned} \tag{21}$$

We utilize the gradient descent method with momentum [25] to train the dual network, since it can accelerate the gradient descent process. The involved moment is a velocity vector accumulated in the direction of persistent reduction for $L(\omega_j)$ across a batch of time slots. To get the approximated value of ω_j, we collect state-action pairs of edge server j with batch size B, and train the dual network based

on the momentum optimizer. Similar to [24], we conduct a consensus update for ω_j by:

$$\omega_j^t = \sum_{i=1}^{M} W_{ji}\omega_i^t - \frac{\chi^t m_{\omega_j}^t}{\sqrt{z_{\omega_j}^t}}, \tag{22}$$

where W_{ji} is a mixing matrix for edge server j and others, and χ^t is the momentum coefficient. Variables $m_{\omega_j}^t$ and $z_{\omega_j}^t$ are moment vectors.

Algorithm 1. Pseudo-Code of Training Processes for Edge Servers

Input: Batch size B; initial policies with policy parameters $\{\theta_j\}_{j=1}^M$, dual parameters $\{\omega_j\}_{j=1}^M$ and value parameters $\{\phi_j\}_{j=1}^M$

Output: Learned policy $\{\pi_{\theta_j}\}_{j=1}^M$.

for round $k = 1, 2, ..$ **do**
  Sample a mini-batch of state transition $(s^t, \{a_j^t\}_{j=1}^M, a^{t+1}, \{r_j^t\}_{j=1}^M)$ with size B.
  for edge server $j = 1, 2, ..., M$ **do**
    Solve Problem P7 by the following steps:
    Compute gradient $g(\omega_j)$ of optimization parameters ω_j in equation (21).
    Update moment vectors $m_{\omega_j}^t$ and $z_{\omega_j}^t$ related to the dual network by:
    $m_{\omega_j} = o_1 m_{\omega_j}^{t-1} + (1-o_1)\big(-g(\omega_j)\big)$
    $z_{\omega_j} = o_2 z_{\omega_j}^{t-1} + (1-o_2) g(\omega_j) \odot g(\omega_j)$
    Update parameter ω_j by equation (22).
  end for
  Sample a mini-batch of state transition $(s^t, \{a_j^t\}_{j=1}^M, a^{t+1}, \{r_j^t\}_{j=1}^M)$ with size B.
  for edge server $j = 1, 2, ..., M$ **do**
    Solve Problem P8 by the following steps:
    Compute gradients $g(\theta_j)$ and $g(\phi_j)$ of parameter θ_j and ϕ_j in equation (16).
    Update moment vectors $m_{\phi_j}^t$ and $z_{\phi_j}^t$ related to the value network by:
    $m_{\phi_j} = o_1 m_{\phi_j}^{t-1} + (1-o_1)\big(-g(\phi_j)\big)$
    $z_{\phi_j} = o_2 z_{\phi_j}^{t-1} + (1-o_2) g(\phi_j) \odot g(\phi_j)$
    Update parameter ϕ_j by:
    $\phi_j^t = \sum_{i=1}^M W_{ji}\phi_i^t - \frac{\chi^t m_{\phi_j}^t}{\sqrt{z_{\phi_j}^t}}$
    Update parameter θ_j by the Adam optimizer.
  end for
end for

After that, we train the value and policy networks to solve the following problem:

$$\begin{aligned} \text{P8:} \quad & \min_{\phi_j, \theta_j} L(\phi_j, \theta_j, \omega_j), \\ \text{s.t.} \quad & \phi_1 = ... = \phi_j = \phi_{j+1} = ... = \phi_M. \end{aligned} \tag{23}$$

Similar to the dual network, we also utilize the momentum optimizer to train the value network and the Adam optimizer for the policy network [26]. Compared with the momentum optimizer, Adam is based on adaptive estimates of lower-order moments for gradients. Then, gradients of the optimization parameters for Problem P8 can be derived by the partial derivative of ϕ_j and θ_j. After that, a consensus update process is conducted for parameter ϕ_j. For parameter θ_j, it does not need to be consistently updated with others, since each edge server trains its policy independently. For the output of the policy network, the first element is regarded as the action related to energy harvesting period, and others are related to the action for server selection. The training process for edge servers and the whole process of OLD-COP algorithm can be found in Algorithms 1 and 2, respectively.

5.3 Algorithm Analysis

In this subsection, we provide comprehensive theoretical analysis for the designed OLD-COP algorithm. We regard edge servers as learning agents, to solve the formulated optimization problem in a decentralized manner, with the output of the learned energy harvesting period and the scheduling queue for task processing. First, we provide the following theories for task scheduling:

Theorem 2 (Upper bound of task completion delay). *When τ^t is fixed, the actual completion time of task e_i^t is no more than the best scheduled result broadcast by edge servers in time slot t, when the remaining energy of mobile client i is sufficient for the scheduling of task e_i^t, i.e., $\iota_{ij}^{t,acl} \le \iota_{i\hat{j}}^t$, where:*

$$\iota_{i\hat{j}}^t = \begin{cases} \tau^t + d_{i\hat{j}}^{t,l} + d_{i\hat{j}}^{t,o}, & q_{i\hat{j}}^t = 1; \\ \tau^t + \max\Big\{\iota_{i'\hat{j}}^t, d_{i\hat{j}}^{t,l}\Big\} + d_{i\hat{j}}^{t,o}, & 1 < q_{i\hat{j}}^t \le |\mathbb{Q}_j^t|, \\ & q_{i'\hat{j}}^t = q_{i\hat{j}}^t - 1. \end{cases} \tag{24}$$

Symbol $\hat{j}$ is the optimal server for task processing of mobile client i, and $q_{i\hat{j}}^t$ is the estimated processing sequence of task e_i^t by edge server $\hat{j}$. Variable $|\mathbb{Q}_j^t|$ is the total number of tasks in $\mathbb{Q}_j^t$.

The proof can be found in Appendix B of Supplementary File, available in the online supplemental material.

Theorem 3 (Upper bound of the average task completion delay). *When mobile devices always have sufficient energy to schedule offloaded tasks, the upper bound of the average task completion delay achieved by the designed algorithm can be obtained by:*

$$d^{max} = \lim_{T\to\infty} \frac{1}{TN} \sum_{t=0}^{T} \sum_{i=1}^{N} \big[\alpha_i^t d_i^{t,o} + (1-\alpha_i^t)\big(\iota_{i\hat{j}}^t - \tau^t\big) + d_i^{t,w}\big]. \tag{25}$$

The proof can be found in Appendix C of Supplementary File, available in the online supplemental material.

In addition, we can prove that the designed online learning algorithm can converge to a stationary solution. First, we provide the following definition:

Definition 2 (Lipschitz continuous gradient). *If function $f(z)$ is lower bounded, differentiable and has a Lipschitz continuous gradient, the following inequation is founded for $L > 0$, i.e., $\| \bigtriangledown f(z_1) - \bigtriangledown f(z_2) \| \le L \| z_1 - z_2 \|$, when parameters z_1 and z_2 satisfy $z_1, z_2 \in R^n$, and R^n is n-dimensional Euclidean space.*

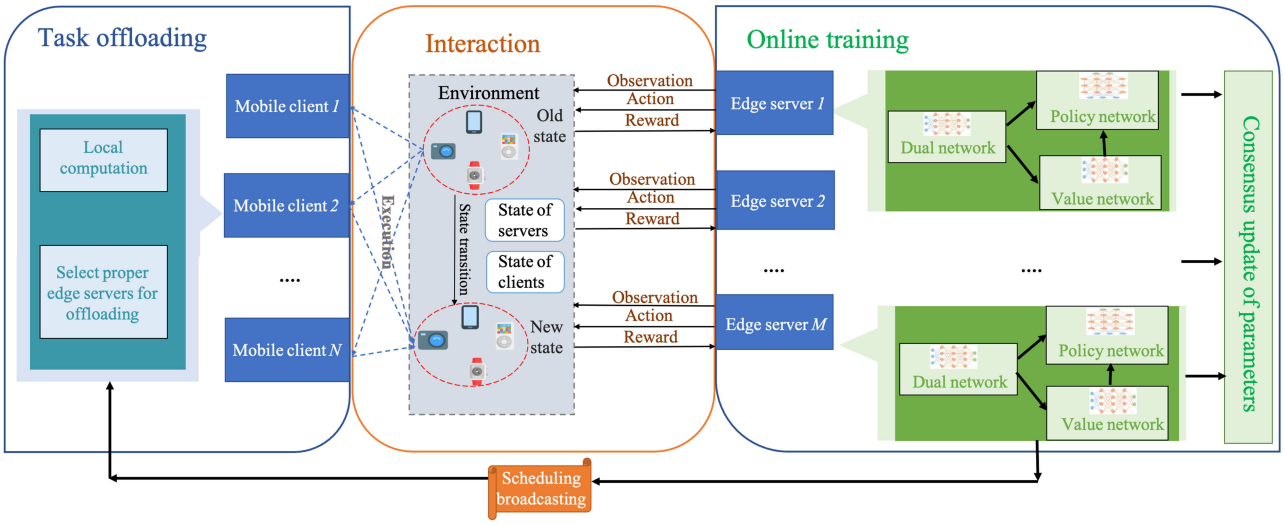


Fig. 2. The structure of the designed algorithm.

The original definition and detailed information of Lipschitz continuous gradient can be found in [27].

Algorithm 2. Pseudo-Code of the OLD-COP Algorithm

Input: State of mobile clients S_m and state of communication-related information S_3c.
Output: The average task completion delay $E(d_i^t)$.

for time slot $t = 0, 1, 2, ..$ **do**
  Get state s^t.
  for mobile client $i = 1, 2, ..., N$ **do**
    Broadcast offloading requests for all tasks.
  end for
  for server $j = 1, 2, ..., M$ **do**
    Get action a_j^t by the learning model based on Algorithm 1.
    Compute energy harvesting period τ^t.
    Compute task scheduling period $\mathbb{T} - \tau^t$.
    Broadcast estimated processing sequence $\mathbb{Q}_j^t$ to mobile clients.
  end for
  for mobile client $i = 1, 2, ..., N$ **do**
    Solve problems P4 and P5.
    Offload tasks that need to be processed remotely to corresponding edge server $\hat{j} \in \mathbb{M}$.
    Process tasks that need to be processed locally.
    Compute average task completion delay $E(d_i^t)$.
  end for
end for

For problems P5 and P6, they are defined for each edge server. From the systematic perspective, optimization problems are:

$$\text{P9:} \quad \max_{\omega} L(\phi, \theta, \omega),$$
$$\text{s.t.} \quad \omega_1 = ... = \omega_j = \omega_{j+1} = ... = \omega_M, \tag{26}$$

and

$$\text{P10:} \quad \min_{\phi,\theta} L(\phi, \theta, \omega),$$
$$\text{s.t.} \quad \phi_1 = ... = \phi_j = \phi_{j+1} = ... = \phi_M, \tag{27}$$

where $\omega = \{\omega_1, ..., \omega_M\}$, $\theta = \{\theta_1, ..., \theta_M\}$, and $\phi = \{\phi_1, ..., \phi_M\}$. In the following, we analyze the algorithm convergence based on problems P6 and P7.

Theorem 4 (Algorithm convergence). *Consider that three neural networks, i.e., dual, policy and value networks, are trained for totally B^2 time slots, and the size of mini-batches for training is B. The designed online learning algorithm converges to the stationary solution of problem P7 with rate $O(\frac{1}{B^2})$ based on the following conditions:*

i) The function approximators, utilized in the neural network training for $v_j(s^t)$, $\pi_j(s^t, a_j^t)$ and $\rho_j(s^t, a^t)$, have Lipschitz continuous gradients, i.e., $\| \Delta_\phi L(\phi, \theta, \tilde{\omega}) - \Delta_\phi L(\phi, \theta, \omega^*) \| \leq c_1/B$ and $\| \Delta_\theta L(\phi, \theta, \tilde{\omega}) - \Delta_\theta L(\phi, \theta, \omega^*) \| \leq c_2/B$, where $\tilde{\omega}$ is the obtained result by solving problem P6;

ii) The stochastic gradients of $g(\phi)$, $g(\rho)$ and $g(\omega)$ estimated by each step are bounded by ϱ^2.

The proof can be found in Appendix D of Supplementary File, available in the online supplemental material.

6 Performance Evaluation

6.1 Simulation Setup

To demonstrate the feasibility of OLD-COP, our experiments are conducted based on Tensorflow 2.1 and Python 3.6. As illustrated in Fig. 3, we utilize the city map of Manhattan for performance validation. We select an area with 1 km $\times$ 1 km squared by red lines, where servers are uniformly distributed in the area without movements. Actually, walkers or riders holding mobile terminals can be viewed as mobile clients [28]. They move along roads in the considered area based on Manhattan mobility model [29]. The simulated time horizon is time-slot-based because of the high repeatability of individuals' activities. The settings of simulated parameters are based on [15] and [19], and can be found in Table 2.

In addition, we consider the Rayleigh fading channel model, where the wireless channel gain between edge server j and mobile client i can be computed by $h_{ij}^t = \Phi_0 \mathbb{X}_{ij}^{-\varpi} h_{ij}^F h_{ij}^S$. Generally, ϖ is between 2 and 4, and we define

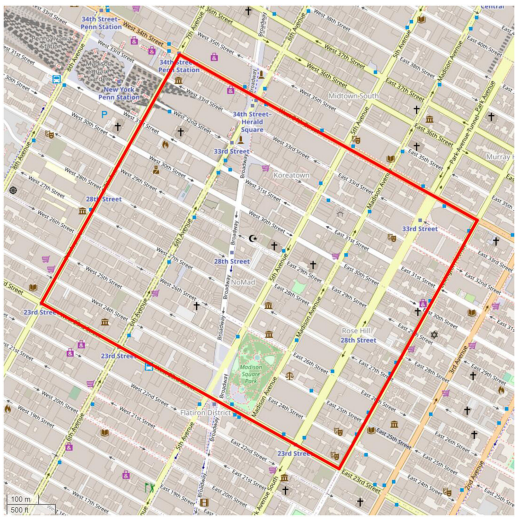


Fig. 3. The city map of Manhattan.

it by 3. Herein, $\Phi_0 = -20dB$, and $\mathbb{X}_{ij}$ is the distance between edge server j and mobile client i. Symbols h_{ij}^F and h_{ij}^S are channel gains caused by small-time-scale channel fading and large-time-scale shadowing, respectively, and are i.i.d vectors of $CN(0,1)$. We leverage multi-layer perceptions for the distributed online learning, where four fully connected layers are constructed for policy, dual and value networks, respectively. Softmax activation function [30], Momentum Optimizer [25] and Adam Optimizer [26] functions are utilized to train neural networks. Since there is no existing study considering distributed computation offloading among multiple edge servers and mobile clients in WP-MEC networks, we compare the designed OLD-COP algorithm with the following four schemes:

1) CoCoRaM [15]: It is a centralized task scheduling algorithm with a set of edge servers and mobile clients. An approximation algorithm is designed for reasonable time allocation and task scheduling in WP-MEC networks.
2) Energy-Aware Scheduling (EAS) [31]: This algorithm is leveraged in our system. Thus, the time division is the same with our algorithm. Whereas, for task scheduling process, mobile clients always select the edge server that can minimize their consumed energy.
3) Random Computation Offloading (RCO): Time division is also set the same with our algorithm, while mobile clients randomly select edge servers for task offloading.
4) Local Computing (LC): All tasks are processed on mobile clients without offloading, and the duration of energy harvesting is set by the length of each time slot.

TABLE 2
Simulation Parameters

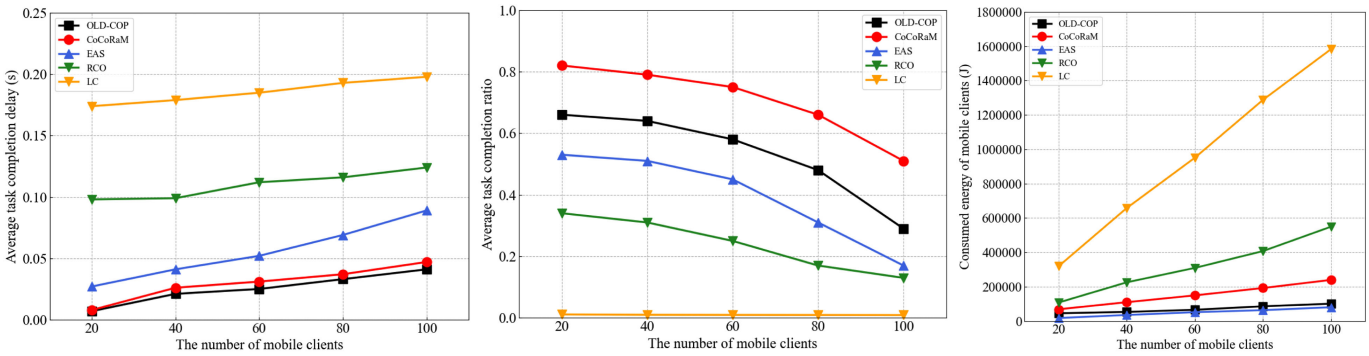
Parameter description	Value
The number of edge servers	$\{3,4,5,6,7,8\}$
The number of mobile clients	$\{10,20,30,40,50,60,70,80\}$
The length of each time slot	$0.3s$
Task generation probability	$\{0.4,0.5,0.6,0.7,0.8\}$
Task size	$[50,100]$ kb
Task required CPU	50 cycles/bit
CPU frequency of each mobile client	$[300,500]$ MHz
CPU frequency of each edge server	2.5 GHz
Transmission power of edge servers	3 W
Transmission power of mobile clients	0.1 W
Energy harvesting efficiency	$\{0.5,0.6,0.7,0.8,0.9\}$
The maximum moving speed of mobile clients	20 km/h
The bandwidth	1 MHz
The learning rate for training models	0.01
The batch size	50

6.2 Simulation Results

6.2.1 Performance based on Different Numbers of Mobile Clients

Fig. 4a illustrates the performance of average task completion delay for CoCoRaM, EAS, RCO, LC and the designed algorithm, OLD-COP, with different numbers of mobile clients. We notice that the average task completion delay of OLD-COP is lower than that of the other four algorithms. This is because our algorithm aims at minimizing the average task completion delay by designing the online scheduling algorithm. CoCoRaM algorithm is a centralized scheduling algorithm in WP-MEC networks, and solves a generalized assignment problem to schedule tasks by fixing the energy harvesting period. Then, it utilizes an approximation algorithm to derive the energy harvesting period. However, merely the minimum energy for local computation of each client is considered and the long-term performance is not considered during the optimization process. Thus, its corresponding performance is worse than that of the designed algorithm. Although EAS has the same time division method with ours, it attempts to minimize the consumed energy of mobile clients first. RCO randomly selects edge servers for offloading without reasonable scheduling, and RC algorithm merely depends on the computation capability of mobile clients, which can cause a large delay. When the number of mobile clients increases, average task completion delays of the five algorithms all become large. This is because more mobile clients lead to more offloaded tasks in the system. Consequently, more tasks consume more computation resources and lead to larger delays.

The performance of average task completion ratio based on different numbers of mobile clients is illustrated in Fig. 4b. Herein, task completion ratio refers to the ratio of tasks that can be successfully processed compared to those generated in each time slot. We can find that the performance on average task completion ratio of OLD-COP is higher than those of EAS, RCO, LC algorithms, while lower than that of CoCoRaM algorithm. This is because CoCoRaM algorithm maximizes the task completion ratio by solving a defined optimization problem in a centralized manner. Whereas, our designed algorithm aims to minimize task completion delay by a designed learning algorithm in a distributed manner. Thus, for task scheduling in our designed algorithm, mobile



(a) Average task completion delay (b) Average task completion ratio (c) Consumed energy of mobile clients

Fig. 4. Performance with different numbers of mobile clients.

clients always select edge servers with the minimum delays for their tasks. The performance of consumed energy of mobile clients with different numbers of mobile clients is shown in Fig. 4c. It is obvious that EAS algorithm has the minimum energy consumption. This is because mobile clients scheduled by EAS algorithm always select edge servers that can minimize their energy consumption for task offloading. To maximize the task completion ratio, more energy is consumed for task transmission and processing in CoCoRaM algorithm. Thus, the consumed energy of mobile clients for OLD-COP is lower than that of CoCoRaM algorithm, while higher than that of EAS algorithm.

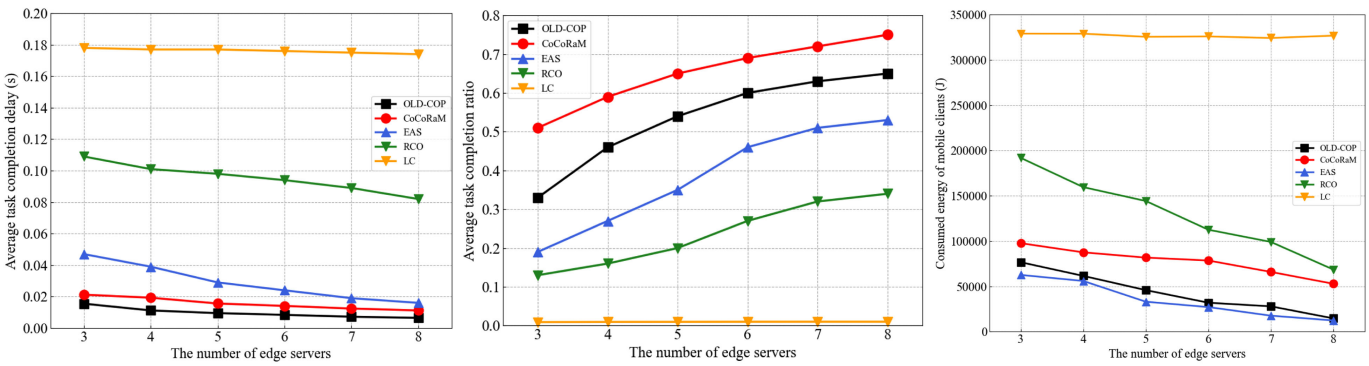
6.2.2 Performance based on Different Numbers of Edge Servers

The results of average task completion delay based on different numbers of edge servers are presented in Fig. 5a. It is obvious that the performance of the five algorithms is similar with that of Fig. 5a, i.e., average task completion delay of OLD-COP is the minimum, CoCoRaM algorithm second, and those of EAS, RCO and LC are higher. When the number of edge servers becomes big, the performance of OLD-COP, CoCoRaM, EAS, RCO and LC algorithms also increases. This is because when there are more edge servers, more computation resources can be utilized for task processing. Then, average task completion delay can be reduced with more computation resources. However, the performance of LC algorithm has almost no change when the number of edge servers becomes big. This is because it processes tasks locally without offloading, so that the computation resources of mobile clients have no change.

Fig. 5b illustrates the performance of average task completion ratio with different numbers of edge servers. We can find that when the number of edge servers becomes big, average task completion ratios of OLD-COP, CoCoRaM, EAS, RCO and LC algorithms also increase. The reason is that, on one hand, a bigger number of edge servers result in more energy sources for mobile clients, and they can harvest more energy. On the other hand, more computation resources can be utilized, speeding up the process of task processing. The performance of consumed energy of mobile clients for the five algorithms is illustrated in Fig. 5c. When the number of edge servers becomes big, the consumed energy of mobile clients decreases. This is because when there are more edge servers, mobile clients have higher chances to offload their tasks to nearer edge servers, which reduces the consumed transmission energy of tasks.

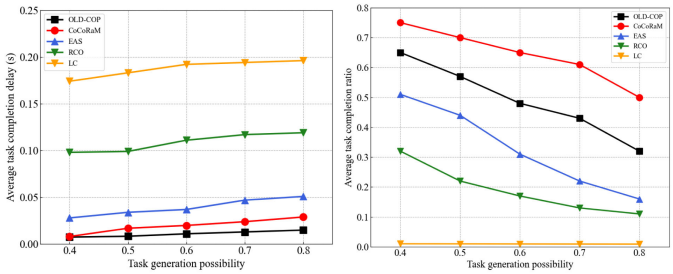
6.2.3 The Impact of Task Generation Possibilities

Fig. 6 illustrates the impact of task generation possibilities on the system performance. The trend of average task completion delay is illustrated in Fig. 6a. It shows that average task completion delay increases when the task generation probability becomes large. For example, when the task generation probability is 0.5, average task completion delays of OLD-COP, CoCoRaM, EAS, RCO and LC algorithms are 0.0084s, 0.017s, 0.034s, 0.099s and 0.183s, respectively. When the task generation probability is 0.7, average task completion delays



(a) Average task completion delay (b) Average task completion ratio (c) Consumed energy of mobile clients

Fig. 5. Performance with different numbers of edge servers.



(a) Average task completion delay (b) Average task completion ratio

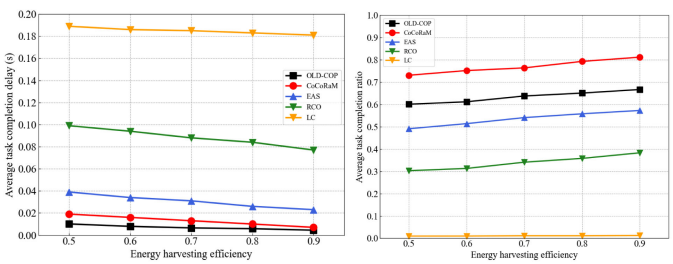
Fig. 6. Performance with different task generation possibilities.

of the five algorithms are 0.013s, 0.024s, 0.047s, 0.117s and 0.194s, respectively. The reason is that, more tasks can be generated when the task generation probability becomes large. Following that, there are more tasks need to be scheduled in the system. Thus, when the number of edge servers is fixed, average task completion delay becomes large.

Fig. 6b is the trend of average task completion ratio with different task generation possibilities. We can observe that when the task generation probability becomes large, the trend of average task completion ratio drops. For instance, when the task generation probability is 0.5, average task completion ratios of OLD-COP, CoCoRaM, EAS, RCO and LC algorithms are 0.57, 0.70, 0.44, 0.22 and 0.0098, respectively. When the task generation probability increases to 0.7, average task completion ratios of the five algorithms are 0.43, 0.61, 0.22, 0.13 and 0.0092, respectively. This is because when more tasks exist in the system, more computation and communication resources are consumed to process those tasks. However, some tasks can not be processed in time, resulting in lower average task completion ratios. In addition, the trend of average task completion ratio is similar with that in Fig. 4b, which has similar reasons for the five algorithms.

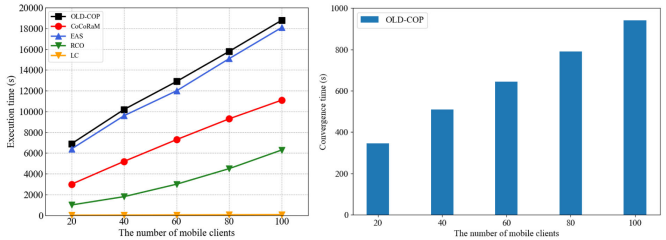
6.2.4 The Impact of Energy Harvesting Efficiency

System performance with different energy harvesting efficiencies is illustrated in Fig. 7. We can observe that, for average task completion delay shown in Fig. 7a, the designed algorithm, OLD-COP, has the best performance compared with the other four algorithms, since our algorithm intends to minimize average task completion delay by designing reasonable online scheduling algorithms. However, CoCoRaM algorithm does not consider the long-term performance and merely tries to approach to the optimal scheduling in each time slot. Although it is a centralized approach, the performance of average task completion delay is still worse than that of our algorithm. When the



(a) Average task completion delay (b) Average task completion ratio

Fig. 7. Performance with different energy harvesting efficiencies.



(a) Execution time (b) Convergence time

Fig. 8. Performance of execution and convergence time.

energy harvesting efficiency becomes large, average task completion ratios of the five algorithms gradually become small due to the sufficient harvested energy.

Fig. 7b shows the performance trend of average task completion ratios for the five algorithms. When the energy harvesting efficiency increases, average task completion ratios of the five algorithms become large. For example, when the energy harvesting efficiency is 0.6, average task completion ratios for OLD-COP, CoCoRaM, EAS, RCO and LC algorithms are 0.612, 0.752, 0.514, 0.313 and 0.0098, respectively. When the energy harvesting efficiency increases to 0.8, average task completion ratios of the five algorithms are 0.651, 0.793, 0.558, 0.358 and 0.011, respectively. The reason is that more harvested energy can support more tasks for both local computing and remote transmission.

6.2.5 The Performance of Execution and Convergence Time

Fig. 8a illustrates the execution time of the five algorithms. It is observed that the execution time of the designed algorithm, OLD-COP, is the maximum, and EAS tightly follows it. The execution time of LC algorithm is the minimum. This is because OLD-COP algorithm includes three neural networks, and periodically trains them based on collected state-action trajectories, consuming much time. Since EAS algorithm has the same method to learn time division with our method by training neural networks, its execution time is similar with ours. However, CoCoRaM, RCO, and LC algorithms do not have the training process. LC algorithm totally depends on local computing and merely needs to put tasks into local computing queues. RCO randomly selects available edge servers, while CoCoRaM algorithm produces approximate optimal computation schedules with given WPT time for WP-MEC networks, thus its time complexity is higher than that of RCO and LC algorithms. When the number of mobile clients becomes big, the execution time of the five algorithms also becomes long. This is because more clients can generate more tasks, and more tasks consume more time for scheduling.

Fig. 8b is the convergence time of the designed algorithm. Since our algorithm and EAS depend on the neural network training, and EAS utilizes the same training method with ours, we merely evaluate the convergence time of ours. We can observe that its convergence time is acceptable, compared with its total execution time. When the number of mobile clients becomes big, the convergence time of the designed algorithm also increases. This is because more clients can generate more tasks, and the algorithm execution time becomes longer, correspondingly.

7 Conclusion

In this paper, we proposed an online learning algorithm for computation offloading in WP-MEC networks with distributed execution. Our objective is to minimize the average task completion delay of mobile clients in a long-term perspective. First, we defined the delay minimization problem by considering task delay and energy constraints. Then, we transformed it into a primal-dual optimization problem based on the Bellman equation. To solve the formulated optimization problem in an efficient way, we designed a novel neural model, which can learn time division and task offloading decisions simultaneously. To further make the learning model trained and executed in a distributed manner, we formed multiple learning models that can be decentralized on edge servers and coordinated to achieve parameter synchronization. At last, both theoretical analyses and performance results showed that our designed algorithm has a significant advantage in terms of average task completion delay and algorithm convergence speed.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

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

[3] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1855–1867, Mar. 2020.

[4] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[5] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[6] X. You *et al.*, "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Sci. China Inf. Sci.*, vol. 64, no. 1, pp. 1–74, 2021.

[7] M. Y. Naderi, P. Nintanavongsa, and K. R. Chowdhury, "RF-MAC: A medium access control protocol for re-chargeable sensor networks powered by wireless energy harvesting," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3926–3937, Jul. 2014.

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

[9] H. Mirghasemi, L. Vandendorpe, and M. Ashraf, "Optimal online resource allocation for SWIPT-based mobile edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–8.

[10] 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 Commun. Lett.*, vol. 8, no. 5, pp. 1320–1323, Oct. 2019.

[11] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3170–3184, May 2020.

[12] J. Xu and S. Ren, "Online learning for offloading and autoscaling in renewable-powered mobile edge computing," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.

[13] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12215–12226, Dec. 2019.

[14] L. Huang, S. Bi, and Y. J. 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.

[15] T. Zhu, J. Li, Z. Cai, Y. Li, and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 596–605.

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

[17] Z. Ning *et al.*, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.3025116, 2020.

[18] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.

[19] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[20] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.3012509.

[21] L. Ji and S. Guo, "Energy-efficient cooperative resource allocation in wireless powered mobile edge computing," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4744–4754, Jun. 2019.

[22] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 2775–2785.

[23] B. Dai *et al.*, "SBEED: Convergent reinforcement learning with nonlinear function approximation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1125–1134.

[24] C. Qu, S. Mannor, H. Xu, Y. Qi, L. Song, and J. Xiong, "Value propagation for decentralized networked deep multi-agent reinforcement learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2019, pp. 1184–1193.

[25] S. Ilya, M. James, D. George, and H. Geoffrey, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–11.

[27] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, 1966.

[28] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 411–425, Feb. 2021.

[29] G. Walunjkar and A. K. Rao, "Simulation and evaluation of different mobility models in disaster scenarios," in *Proc. 4th Int. Conf. Recent Trends Electron. Inf. Commun. Technol.*, 2019, pp. 464–469.

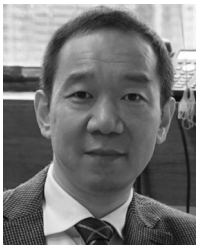
[30] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 243–252.

[31] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 87–96, Jan. 2017.

Xiaojie Wang received the PhD degree from the Dalian University of Technology, Dalian, China, in 2019. After that, she was a postdoctoral with the Hong Kong Polytechnic University. She is currently a distinguished professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interests include wireless networks, mobile edge computing and machine learning. She has published more than 40 scientific papers in international journals and conferences, such as *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Communications Surveys and Tutorials*.

Zhaolong Ning received the PhD degree from Northeastern University, China in 2014. He was a research fellow with Kyushu University from 2013 to 2014, Japan. Currently, he is a full professor with the College of Communication and Information Engineering, the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include Internet of Things, mobile edge computing, deep learning, and resource management. He has published more than 120 scientific papers in international journals and conferences. He serves as an associate editor or guest editor of several journals, such as *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Social Computational Systems*, *The Computer Journal and so on*.

Lei Guo received the PhD degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a full professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an editor for several international journals. His research interests include communication networks, optical communications, and wireless communications.



Song Guo (Fellow, IEEE) is currently a full professor with the Department of Computing, The Hong Kong Polytechnic University. He also holds a Changjiang chair professorship awarded by the Ministry of Education of China. He is a fellow of the Canadian Academy of Engineering and a fellow of the IEEE (Computer Society). His research interests include big data, edge AI, mobile computing, and distributed systems. He published many papers in top venues with wide impact in these areas and was recognized as a Highly Cited Researcher (Clarivate Web of Science). He is the recipient of over a dozen best paper awards from IEEE/ACM conferences, journals, and technical committees. He is the editor-in-chief of IEEE Open Journal of the Computer Society and the chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He was an IEEE ComSoc distinguished lecturer and a member of IEEE ComSoc Board of Governors. He has served for IEEE Computer Society on Fellow Evaluation Committee, and been named on editorial board of a number of prestigious international journals like *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Emerging Topics in Computing*, etc. He has also served as chairs of organizing and technical committees of many international conferences.

Xinbo Gao received the BEng, MSc, and PhD degrees in electronic engineering, signal and information processing from Xidian University, Xi'an, China, in 1994, 1997, and 1999, respectively. From 1997 to 1998, he was a research fellow with the Department of Computer Science, Shizuoka University, Shizuoka, Japan. From 2000 to 2001, he was a postdoctoral research fellow with the Department of Information Engineering, Chinese University of Hong Kong, Hong Kong. Since 2001, he has been with the School of Electronic Engineering, Xidian University, where he is currently a Cheung Kong professor of Ministry of Education, and a professor of Pattern Recognition and Intelligent System, and a professor of Computer Science and Technology with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has published six books and around 300 technical articles in refereed journals and proceedings. His research interests include Image processing, computer vision, multimedia analysis, machine learning, and pattern recognition. He is on the editorial boards of several journals, including *Signal Processing* (Elsevier) and *Neurocomputing* (Elsevier). He served as the general chair/co-chair, Program Committee chair/co-chair, or a PC Member for around 30 major international conferences. He is a fellow of the Institute of Engineering and Technology and the Chinese Institute of Electronics.

Guoyin Wang (Senior Member, IEEE) received the BS, MS, and PhD degrees in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 1992, 1994, and 1996, respectively. During 1998-1999, he was a visiting scholar with the University of North Texas, Denton, TX, and the University of Regina, Regina, SK, Canada. Since 1996, he has been with the Chongqing University of Posts and Telecommunications, Chongqing, China, where he is currently a professor, a vice-president of the university, and the director with the Chongqing Key Laboratory of Computational Intelligence. He was appointed as the director of the Institute of Electronic Information Technology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, in 2011. He is the author of ten books and has more than 300 reviewed research publications. His current research interests include rough set, granular computing, knowledge technology, data mining, neural network, and cognitive computing. He is the editor of dozens of proceedings of international and national conferences. He was the President of International Rough Set Society (IRSS) from 2014 to 2017. He is currently a vice-president of the Chinese Association for Artificial Intelligence (CAAI) and a Council Member of the China Computer Federation (CCF).