Intelligent Compression Offloading and Adaptive Resource Allocation for Wireless Powered MEC

Xianlong Jiao, *Member, IEEE*, Yunhui Chen, Xiang Wu, Ruiyuan Li, Songtao Guo, *Senior Member, IEEE*, Yong Ma, and Jiannong Cao, *Fellow, IEEE*

***Abstract*—As a novel promising computational paradigm, wireless powered mobile edge computing (WPMEC) has been proposed to offer real-time energy and computing services for Internet of Things (IoT) devices. However, time-varying limited resources such as communication quality and residual energy pose great challenges in devising suitable real-time task offloading and resource allocation strategies to meet users' requirements for low latency and energy consumption. Existing studies either employ raw data offloading methods with significant communication overhead, or utilize ordinary data compression methods that result in poor compression effects. To cope with the challenges, this paper considers introducing the state-of-the-art lossless data compression technology into WPMEC and study the online joint optimization problem of task offloading decision, charging time allocation, and compression proportion allocation with the goal of optimizing task completion time. To tackle this problem, we propose an Intelligent Compression Offloading and adaptive resource Allocation algorithm called ICOA. We first put forward a well-devised framework based on deep reinforcement learning to generate a offloading decision vector set in real-time. Then we standardize the resource allocation problem as a linear programming problem and solve it using the simplex method. The experimental results on a real dataset show that, compared with the benchmark algorithms, the proposed algorithm can effectively reduce the task accomplishing time and energy consumption, and achieve the best approximate ratio. Moreover, ICOA requires low runtime, and can satisfy the real-time and effectiveness requirements very well.**

***Index Terms*—Offloading decision generation, lossless data compression, resource allocation, wireless powered mobile edge computing.**

I. Introduction

With the rapid development and popularization of the information technology, the ubiquitous interconnection of wireless devices and the intelligent perception of data enable Internet of Things (IoT) to significantly improve people's life and work efficiency [1]. Mobile edge computing (MEC) moves the computing and storage capacity to the edge of the IoT network to serve users more closely, thereby reducing task completion time and improving the response speed [2], [3]. However, the extremely limited battery lives of IoT devices have become one of the important factors affecting the quality of user experience. Traditional wired charging methods not only limit the usage scenarios of IoT devices, but also bring cumbersome operation steps to users [4], when these devices are located in complex or hazardous environments. Moreover, emerging computation-intensive application fields (such as smart home, augmented reality, smart healthcare, etc.) put forward higher requirements for the energy efficiency of IoT devices in MEC applications. Fortunately, the wireless charging technology has been utilized to reduce manual maintenance costs and ensure the sustainability guarantee of energy supply [5]. Customized wireless charging solutions for these emerging application areas can help promote the rapid development of related industries [6]. Therefore, wireless powered mobile edge computing (WPMEC) has received widespread attention in recent years [7], [8], [9].

The booming development of the IoT has led to the continuous expansion of latency-sensitive and computation-intensive applications [10]. One of the advantages of WPMEC is to achieve better application responsiveness by offloading tasks from IoT devices to nearby edge servers [11]. In a typical user behavior analysis application scenario, multiple mobile wireless devices equipped with high-precision sensors continuously collect real-time kinematic and user-related floating-point data, including acceleration, angular velocity, and positional coordinates, etc. These devices are wirelessly powered by dedicated edge servers, eliminating the need for frequent manual charging and enabling sustained operation in dynamic environments. The edge servers provide computation offloading and real-time data processing. They perform feature extraction tasks (such as trajectory prediction, motion anomaly detection, or sensor fusion) and return the processed results to the mobile devices. In the WPMEC network, reasonable allocation of computing, communication and charging resources can enhance the performance of task offloading. Worthy of note is that, to avoid signal interference, wireless charging and task offloading cannot be performed simultaneously. In real-time applications, if the charging time is long, the task offloading time will be short. If the charging time is short, the energy collected by the devices will be little, thereby affecting the sustainable operational capability of the devices. Hence, it is vital to devise task offloading and resource allocation

Received 29 June 2025; revised 8 October 2025; accepted 13 October 2025. Date of publication 22 October 2025; date of current version 11 December 2025. This work was supported in part by the Natural Science Foundation of Chongqing under Grant CSTB2024NSCQ-MSX0522, and in part by the National Natural Science Foundation of China under Grant 62072064, Grant 62272069, and Grant 62572086. *(Corresponding author: Xianlong Jiao.)*

Xianlong Jiao, Yunhui Chen, Xiang Wu, Ruiyuan Li, and Songtao Guo are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: xljiao@cqu.edu.cn; yhchen@stu.cqu.edu.cn; xwu@stu.cqu.edu.cn; ruiyuan.li@cqu.edu.cn; guosongtao@cqu.edu.cn).

Yong Ma is with Jiangxi Normal University, Nanchang 330022, China (e-mail: may@jxnu.edu.cn).

Jiannong Cao is with the Department of Computing, Hong Kong Polytechnic University, Kowloon Hong Kong (e-mail: jiannong.cao@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TSC.2025.3624068

methods for improving the network performance and resource utilization.

A few papers focus on optimizing the total computation rates [5] or the total task completion time [12] by proposing online task offloading and resource allocation methods. Nevertheless, these studies rely on raw task data transmission, and incur high communication overhead. The task data, such as sensor data from smart home devices and physiological monitoring data from smart medical devices, is usually floating point data. The task data is not only massive in quantity, but also requires extremely high precision. In particular, the lossless compression technology for floating point data can significantly reduce data storage and transmission overhead without sacrificing any data accuracy [13], [14], [15]. Therefore, it has very important research significance to apply the lossless compression technology into the online task offloading and resource allocation, which has not been well investigated by the existing research [5], [12]. Some studies [16], [17], [18], [19], [20] incorporate the lossless compression technology into the task offloading, but does not utilize the state-of-the-art lossless compression technology. Hence, it is difficult for these studies to effectively reduce the task offloading time and energy consumption.

This paper investigates the online task offloading and resource allocation problem for WPMEC with the support of the state-of-the-art lossless compression technology Elf [13]. Elf is a novel and highly efficient stream-based lossless compression algorithm for floating-point data. It demonstrates superior compression performance over existing lossless compression algorithms, and thus is adopted in this paper. In addition to the CPU and storage, charging time and compression proportions are critical system resources that significantly impact task completion time. These resources require adaptive allocation in response to dynamic system conditions, and thus are considered in our paper.

Specifically, we consider the joint optimization problem of task offloading decision, charging time allocation, and compression proportion allocation under both the residual energy and real-time constraints for WPMEC. We face the following two challenges on real-time and efficiency. *1) How to achieve the joint optimization decision solution of task offloading and resource allocation in real-time under dynamic changing scenarios?* Channel gains may change with the movement of IoT devices. The size of the task data that the device needs to complete is random. Affected by the energy harvesting and consuming, IoT devices possess time-varying residual energy. These dynamic changing factors in WPMEC make it highly challenging to achieve the joint optimization decision solution of task offloading and resource allocation in real-time. *2) How to make the devised algorithm converge to the optimal solution by learning the previous experience?* The decision of task offloading and resource allocation is made in each time-frame. It is vital to learn the previous decision experience to assist the current decision-making. Nevertheless, the environment varies across different time-frames, and it is challenging for the algorithm to converge to the optimal solution by learning the previous decision-making experience.

In this paper, to cope with the above challenges, we propose an Intelligent Compression Offloading and adaptive resource Allocation algorithm, called ICOA, which can make real-time decision for task offloading and resource allocation with near optimal task completion time. The proposed novel ICOA algorithm integrates a well-devised deep reinforcement learning (DRL) framework with a simplex-based resource allocation method. This hybrid strategy is specifically designed to meet stringent real-time and high-efficiency requirements in dynamic environments. The DRL framework learns to make intelligent offloading decisions, while the simplex method provides adaptive resource optimization. This combination enables our algorithm to rapidly converge to a near-optimal solution, significantly outperforming traditional methods in both the task completion time and resource utilization efficiency.

First, we decompose the joint optimization problem into two sub-problems: offloading decision generation, and allocation of charging time and compression proportion. Second, we propose a well-devised deep reinforcement learning (DRL) framework to generate an offloading decision vector set in real-time. We employ the DRL framework for its model-free capability to learn optimal policies directly from high-dimensional state spaces and complex environmental interactions. By leveraging deep neural networks, DRL generalizes effectively across states and learns more intelligent, robust, and self-improving solutions without requiring explicit system models or prior dynamics knowledge. Third, for each generated offloading decision vector, we standardize the second sub-problem as a linear programming problem, and address this sub-problem via the simplex method. Finally, we achieve the optimal solution in all the candidate solutions, and promote the algorithm convergence by leveraging the experience replay technology.

The main contributions of this paper are listed as follows:

- We formulate the joint optimization problem of offloading decision generation, charging time allocation, and compression proportion allocation in WPMEC, whose goal is to minimize the total task completion time while ensuring energy feasibility and time feasibility. To our best knowledge, this paper is the first research to study this problem. We formally prove the non-convexity and NP-hardness of this problem.
- To reduce the difficulty of problem solving, we decompose the investigated problem into two sub-problems. We propose an algorithm ICOA based on a well-devised DRL framework and the simplex method to resolve these two sub-problems. We prove that the time complexity of ICOA is polynomial.
- We utilize a real dataset to comparatively evaluate the proposed ICOA algorithm. The experimental results demonstrate that, ICOA performs obtains lower task completion time and energy consumption than the benchmark algorithms. ICOA achieves the largest approximation ratio to the optimal algorithm, and requires low runtime.

The rest of this paper is organized as follows. Section II provides related work. Section III introduces the system model and formulates the investigated problem. Section IV proposes the algorithm ICOA. Section V analyzes the experimental results of the proposed algorithm, and Section VI concludes this paper.

II. Related Work

In recent years, the MEC has shown broad application prospects for development, and has attracted extensive attention from researchers [21], [22], [23]. In this section, we present the related work on task offloading and resource allocation in WPMEC, and data compression applied to MEC.

Task offloading and resource allocation in WPMEC: The problem of task offloading and resource allocation in WPMEC has recently been widely investigated. A multitude of methodologies have been put forth with varying objectives, such as energy efficiency, delay reduction, fairness or revenue maximization. Some research [24], [25], [26], [27], [28] focuses on efficiently allocate the resource to improve channel quality or energy transmission efficiency. For instance, Cao et al. [24] proposed a novel performance evaluation index to model the differential service transmission performance and designed a collaborative transmission mechanism and joint resource allocation scheme that combines intelligent reflective surface adaptive association to solve the coupling performance and resource management problems. Lu et al. [25] proposed the design of a wireless power supply MEC system assisted by intelligent reflectors, introduced the revenue of edge servers as an index to evaluate the system performance and proposed an iterative algorithm to maximize the performance revenue of the edge server. Zhu et al. [26] formulate a joint optimization problem of wireless charging and computation offloading in socially-aware D2D-assisted WP-MEC to maximize the utility, characterized by wireless devices' residual energy and the strength of social relationship. Dong et al. [27] proposed a wireless power transmission time optimization algorithm based on differential evolution, which designed a hybrid mutation operator and a disturbance-based binomial crossover operator, while introducing micro populations to improve optimization efficiency. Wen et al. [28] innovatively proposed a diffusion-based contract model designed to incentivize edge devices to contribute resources, thereby supporting high-quality content generation for users. However, these studies do not consider the collaborative optimization of task offloading and resource allocation, and also rely on raw data transmission modes. Some papers consider collaborative optimization of task offloading and resource allocation in WPMEC, focusing on dynamic allocation of computing, communication, and energy resources, and deeply coupling with offloading strategies. For example, Malik et al. [29] proposed a wireless power transfer integrated solution with computation offloading function, which jointly realizes data partitioning, time allocation, transmission power control, and satisfies the maximum feasible proportion of wireless charging requests by achieving optimal energy beamforming for wireless charging. Zhong et al. [30] developed a multi-dimensional contract model based on diffusion models to effectively motivate edge servers to contribute computational and bandwidth resources, thereby supporting efficient embodied Artificial Intelligence (AI) twin migration. Chen et al. [31] studied the wireless charging MEC system supports full duplex, which improves the efficiency of energy collection and task offloading by using full duplex technology on hybrid access points. Popska et al. [32] proposed an efficient online fairness-aware resource allocation method for WPMEC networks with TDMA and partial offloading. He et al. [33] considered minimizing the energy consumption of MEC networks and designed an Lyapunov-based online algorithm. However, the above studies do not consider data compression, and still incur high communication overhead.

Data compression applied to MEC: Some research focuses on optimizing the performance of MEC systems by leveraging the data compression technology, focusing on reducing energy consumption, reducing latency, or improving computing efficiency. For instance, Cheng et al. [17] studied the data compression scheme in the edge computing process of multi-user UAVs to reduce the energy consumption of wireless devices. Han et al. [18] considered MEC blockchain networks that support data compression and formulate a nonconvex problem to minimize the total energy consumption of nodes. Li et al. [19] optimized the system by jointly optimizing offloading decisions, compression decisions, transmission power, and computing resources. Tu et al. [20] studied the energy consumption minimization problem by jointly optimizing the user's data compression and offloading time, transmission power, and task compression and offloading ratio. Zhang et al. [34] proposed an energy-saving computing offload algorithm based on deep deterministic policy gradient to improve accuracy and reduce energy consumption in time-varying environments. Lai et al. [35] innovatively employed model compression techniques in resource-constrained mobile edge networks, thereby enabling efficient AI task offloading. Zheng et al. [36] studied the computing offload and semantic compression for intelligent computing tasks in MEC systems to jointly optimize the system utility. However, the above studies do not consider the energy benefits brought by wireless charging, which reduces the sustainability of wireless devices in specific situations. In addition, these studies either employed the lossy compression methods or did not incorporate the latest lossless compression technology, which can lead to the loss of accuracy or suboptimal compression efficiency. It remains open to explore more efficient lossless compression mechanisms to improve the performance of WPMEC.

III. Preliminary

In this section, we first present the models used in the WPMEC system and subsequently formulate the investigated problem.

A. System Model

As shown in Fig. 1, this paper studies the WPMEC system model where an edge server (ES) is responsible for providing wireless edge services to n wireless devices (WDs). The set of all WDs is denoted by $S_{WDs} = \{WD_1, WD_2, \ldots, WD_n\}$. The complete time of the system contains $\mathcal{T}$ consecutive time-frames, each of which has the same length of μ. We use t to represent the current time-frame. We suppose that the positions of the WDs remain fixed within each time-frame, but may change

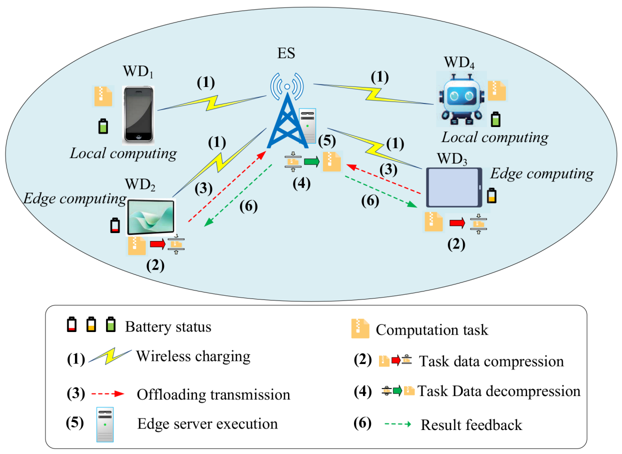


Fig. 1. WPMEC system model.

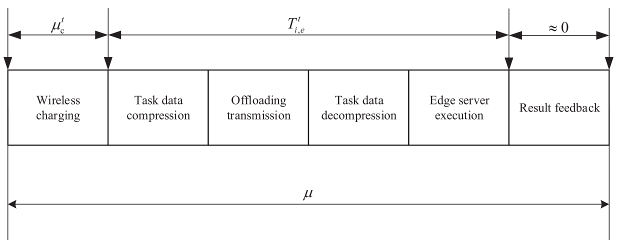


Fig. 2. The workflow within a time frame.

TABLE I
SYMBOLS

Symbols	Description
t	Current time-frame
$\mathcal{T}$	The maximum number of time-frames for running
WD_i	The i-th wireless device
μ	The time-frame length
h_i^t	Channel gain of WD_i in time-frame t
B	Bandwidth
P_i	Transmission power of WD_i
N_0	Noise power
r_i^t	Transmission speed of WD_i in time-frame t
D_i^t	Task data size of WD_i in time-frame t
c_i	Number of CPU cycles required for WD_i to process 1-bit of data
f_i^l	Computing rate of WD_i
$T_{i,l}^t$	Task complete time of WD_i in time-frame t
β_i^t	Data compression proportion of WD_i in time-frame t
J_i	Number of cycles of WD_i required to compress 1-bit of data
$T_{i,com}^t$	WD_i's compression time in time-frame t
α_i^t	Data compression rate of WD_i in time-frame t
$T_{i,off}^t$	WD_i's offloading time in time-frame t
J_{ES}	Number of CPU cycles required for the ES to decompress 1-bit of data
f_{ES}	CPU frequency of ES
$T_{i,dec}^t$	Decompression time of WD_i in time-frame t
η_i^t	Energy collection efficiency of WD_i in time-frame t
P_{ES}	Transmission power of ES
μ_c^t	Charging time in time-frame t
C_i^t	Energy collected by WD_i in time-frame t
$E_{i,com}^t$	Compression energy consumption of WD_i in time-frame t
$E_{i,off}^t$	Offloading energy consumption of WD_i in time-frame t

across distinct time-frames, thereby introducing different channel conditions.

In addition, each WD has two working modes: local computing and edge computing. In the local computing mode, WD directly utilizes wireless charging energy received from edge servers to complete the computation and processing of task data locally. In the edge computing mode, WD first uses the received wireless charging energy to power itself, and then compresses the task data. The compressed data is transmitted to the ES through a dynamically changing wireless channel. After receiving the data, ES will decompress it, perform computational processing, and finally send the resulting data back to the corresponding target WD. In the edge computing mode, the workflow within a time frame is illustrated as shown in Fig. 2.

Main symbols and their description used in this paper are summarized in Table I

B. Communication Model

We assume that data communication adopts the half duplex mode, and task data offloading and result data returning are transmitted through non-orthogonal multiple access channels. We use $h_i^t (1 \leq i \leq n)$ to represent the wireless channel gain between ES and WD_i and assume that it is fixed within a time-frame and may vary across different time-frames. According to [5], we define h_i^t as follows:

$$h_i^t = h_a \left(\frac{3 \cdot 10^8}{4\pi\varphi d_{i,s}} \right)^{\gamma} \mathcal{B}_i^t \tag{1}$$

where h_a, φ, and γ are the antenna gain, the carrier frequency, and the path loss index, respectively. $d_{i,s}$ is the distance between WD_i and ES. $\mathcal{B}_i^t$ is the independent random channel fading factor. We achieve the transmission speed as follows [37]:

$$r_i^t = B \log \left(1 + \frac{P_i h_i^t}{N_0} \right) \tag{2}$$

where B is the bandwidth, P_i is the transmission power of WD_i and N_0 is the noise power.

C. Compression Model

Numerous edge computing applications rely fundamentally on floating-point arithmetic due to its essential role in processing high-resolution, dynamic signals under noisy and resource-constrained edge environments. In the user behavior analysis application, the task data includes acceleration, angular velocity, and positional coordinates, etc. Hence, we assume that all the task data is composed of floating-point numbers. The task data is compressed using the Elf method proposed in [13] to achieve lossless compression before offloading. The basic principle of compression is to increase the number of suffixes 0 by tail erasing, and then use the XOR method for compression. For each time-frame, we can obtain the task data compression rate α_i^t for each WD_i, and use J_i to represent the amount of CPU cycles needed to compress one bit of data. As compressing data requires local CPU resources, this will lead to an increase in WDs' energy consumption. Hence, we need to adjust the data compression proportion β_i^t reasonably to meet energy feasibility constraint.

D. Time Model

In the local computing mode, the task complete time of WD_i can be formulated as:

$$T_{i,l}^t = \frac{D_i^t c_i}{f_i^l} \tag{3}$$

where D_i is the data size and f_i^l is the local computing rate, c_i represents the amount of CPU cycles needed for WD_i to process 1-bit of data.

In edge computing mode, WDs offload their task data to ES for execution. The process includes five steps: task data compression, offloading transmission, task data decompression, edge server execution, and result feedback.

The time of task data compression can be expressed as:

$$T_{i,com}^t = \frac{\beta_i^t D_i^t J_i}{f_i^l} \tag{4}$$

The time of offloading transmission can be expressed as:

$$T_{i,off}^t = \frac{(1-\beta_i^t)D_i^t + \alpha_i^t \beta_i^t D_i^t}{r_i^t} \tag{5}$$

The time of task data decompression is expressed as follow:

$$T_{i,dec}^t = \frac{\beta_i^t D_i^t J_{ES}}{f_{ES}} \tag{6}$$

where f_{ES} is the cycle frequency of the server CPU and J_{ES} is the number of cycles required to decompress one bit of data.

The time of edge server execution can be achieved as:

$$T_{i,exec}^t = \frac{D_i^t c_{ES}}{f_{ES}} \tag{7}$$

where c_{ES} is the amount of cycles needed for the ES to handle one-bit data.

Due to the very small amount of data in the executed results, which can be ignored, we reasonably assume that the time for returning the results is also negligible [19].

The total time in the edge computing mode, obtained by adding them up, is represented by $T_{i,e}^t$:

$$T_{i,e}^t = T_{i,com}^t + T_{i,off}^t + T_{i,dec}^t + T_{i,exec}^t \tag{8}$$

E. Energy Model

This paper assumes that each WD can convert the wireless energy signals received from the ES into electrical energy, which is then stored in a rechargeable battery. We assume that wireless charging time is μ_c^t $(0 \le \mu_c^t \le \mu)$. According to [5], the harvested energy in each time-frame is represented as:

$$C_i^t = \eta_i^t P_{ES}^t h_i^t \mu_c^t \tag{9}$$

where η_i^t and P_{ES}^t respectively represent the energy collection efficiency and ES's transmission power within the time-frame μ. η_i^t is within the range of (0,1). We use L_i^t to represent the remaining energy of WD_i. Specifically, due to the discreteness of time-frames, L_i^t is defined as:

$$L_i^t = L_i^{t-1} + C_i^{(t-1)} - E_i^{t-1} \tag{10}$$

where E_i^{t-1} represents the energy consumed by WD_i in time-frame $t-1$. Specifically, based on the characteristics of battery power supply, L_i^t satisfies $L_{\min} \le L_i^t \le L_{\max}$, where $L_{\min}$ is the minimum remaining energy required to support the normal operation of WDs, and $L_{\max}$ is the battery capacity. When $L_i^t < L_{\min}$, WD_i will stop working until it obtains enough energy through wireless charging.

Then, we achieve the energy consumption in the local computing mode as follows:

$$E_{i,l}^t = a_i D_i^t c_i \tag{11}$$

where a_i is the energy consumed per CPU cycle, usually expressed as: $a_i = b(f_i^l)^2$, where b is a constant. For each WD, if it chooses the edge computing mode, its task execution energy consumption consists of task data compression energy consumption and offloading transmission energy consumption. That is,

$$E_{i,e}^t = E_{i,com}^t + E_{i,off}^t \tag{12}$$

where $E_{i,com}^t = a_i \beta_i^t D_i^t J_i$, and $E_{i,off}^t = P_i T_{i,off}^t$.

F. Problem Formulation

We define $\mathbf{z}^t = [z_1^t, z_2^t, \ldots, z_n^t]$ as the decision vector for all WDs to determine the computing mode. If the WD works in the local computing mode, then $z_i^t = 0$, else $z_i^t = 1$. Let Q represent the total time it takes for all WDs in the system to complete their tasks within a time-frame. According to (3) and (8), we can obtain:

$$Q(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t) = \sum_{i=1}^{n}((1-z_i^t)T_{i,l}^t) + z_i^t T_{i,e}^t \tag{13}$$

where the vectors $\mathbf{L}^t = [L_1^t, L_2^t, \ldots, L_n^t]$, $\mathbf{D}^t = [D_1^t, D_2^t, \ldots, D_n^t]$, and $\mathbf{h}^t = [h_1^t, h_2^t, \ldots, h_n^t]$ represent the remaining energy, data sizes, and channel gains of all devices in the system, respectively.

Unlike existing studies, we can adjust the compression proportion to regulate the compression energy consumption. Therefore, the key challenge lies in how to adaptively adjust both the charging time and the compression proportion under energy feasibility constraint to optimize the task completion time. We require to providing decision support for the wireless charging time μ_c^t of ES, the offloading decision vector $\mathbf{z}^t = [z_1^t, z_2^t, \ldots, z_n^t]$ of WDs, and the data compression proportion $\boldsymbol{\beta}^t = [\beta_1^t, \beta_2^t, \ldots, \beta_n^t]$ at the initiation of time-frame t. The goal is to minimize the sum of task completion time Q, as shown below:

$$\mathcal{P}: Q^* = \min_{\mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t} Q(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t) \tag{14}$$

$$\text{s.t.} \quad \forall i \in \{1, \ldots, n\},$$

$$L_i^t + C_i^t - ((1-z_i^t)E_{i,l}^t + z_i^t E_{i,e}^t) \ge L_{\min} \tag{15}$$

$$\mu_c^t + \sum_{i=1}^{n} z_i^t T_{i,e}^t \le \mu \tag{16}$$

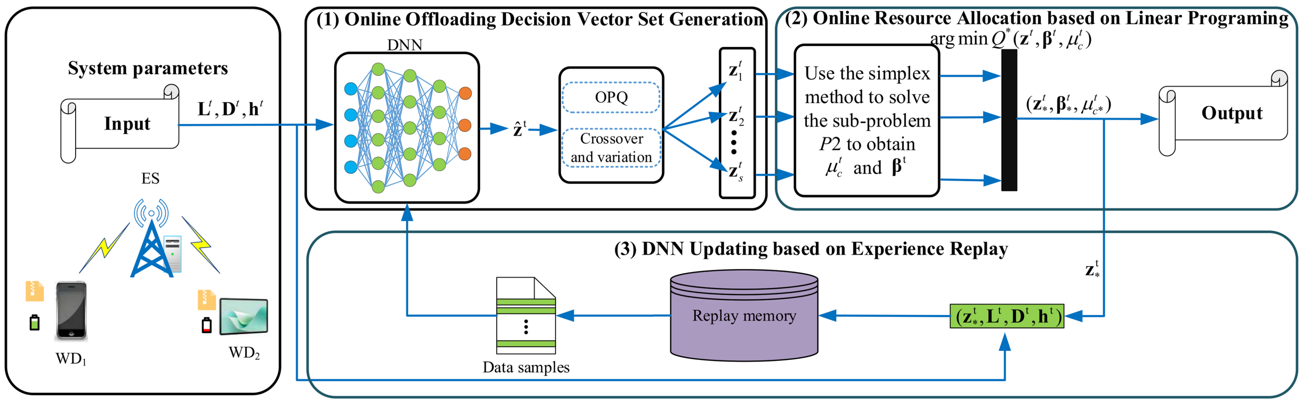

System parameters
Input
$\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t$
ES
WD1
WD2
(1) Online Offloading Decision Vector Set Generation
DNN
$\hat{\mathbf{z}}^t$
OPQ
Crossover and variation
$\mathbf{z}_1^t$
$\mathbf{z}_2^t$
$\mathbf{z}_s^t$
(2) Online Resource Allocation based on Linear Programing
$\arg\min Q^*(\mathbf{z}^t, \boldsymbol{\beta}^t, \mu_c^t)$
Use the simplex method to solve the sub-problem P2 to obtain μ_c^t and $\boldsymbol{\beta}^t$
$(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t)$
Output
(3) DNN Updating based on Experience Replay
$\mathbf{z}_*^t$
Replay memory
$(\mathbf{z}_*^t, \mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t)$
Data samples

Fig. 3. ICOA algorithm overview.

$$0 \le \mu_c^t \le \mu \tag{17}$$

$$\forall i \in \{1, \ldots, n\}, z_i^t \in \{0, 1\} \tag{18}$$

$$\forall i \in \{1, \ldots, n\}, 0 \le \beta_i^t \le 1 \tag{19}$$

where (15) represents the constraint for residual energy feasibility, (16) represents the constraint for real-time feasibility, and (17)–(19) are the range of value for variables μ_c^t, z_i^t, and β_i^t.

Theorem 1: The problem $\mathcal{P}$ is a non-convex and NP-hard problem.

Proof: Due to the discreteness of the variable z_i^t in the objective function (14) and constraints (15), (16), (18), the problem $\mathcal{P}$ is a non-convex problem. Besides, assuming that we allocate a fixed amount of resources to each WD, and we construct a knapsack for every WD, appropriately setting the weight and value of the knapsack based on the task completion time associated with that WD, we can effectively reduce the offloading decision problem to a classic 0-1 knapsack problem. Since 0-1 knapsack problem is an NP-hard problem, $\mathcal{P}$ is also NP-hard.

IV. Algorithm Design

According to Theorem 1, we know that directly solving this non-convex NP-hard problem is computationally infeasible due to exponential growth in computational complexity. To reduce the difficulty of problem solving, we decompose the problem into two sub-problems:

1) *Sub-problem $\mathcal{P}1$ (Online Offloading Decision Vector Set Generation)*: At the start of every time-frame, we use appropriate strategies to generate a set of candidate offloading decision vectors.
2) *Sub-problem $\mathcal{P}2$: (Online Resource Allocation)* At the start of every time-frame, we allocate charging time μ_c^t and compression proportion $\boldsymbol{\beta}^t$ for every candidate decision vector.

Through analysis, we can achieve that the original problem $\mathcal{P}$ has a solution $(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t)$ only when both subproblems $\mathcal{P}1$ and $\mathcal{P}2$ have solutions $\mathbf{z}_*^t$ and $(\boldsymbol{\beta}_*^t, \mu_{c*}^t)$, respectively. Any subproblem without a solution will result in the original problem without a solution. Hence, the decoupled subproblems are equivalent to the original problem. The decision space for the sub-problem $\mathcal{P}1$ is 2^n. In order to reduce the search space while achieving high efficiency in offloading decision making, we adopt a carefully designed DNN to generate a set of candidate decision vectors. For sub-problem $\mathcal{P}2$, after obtaining several candidate decision vectors, we will solve for their corresponding resource allocation schemes. We standardize the sub-problem into a standard linear programming problem and solve it using the simplex method to obtain the values of μ_c^t and $\boldsymbol{\beta}^t$. After the above two steps, we obtain several solutions. We select the optimal one as the final solution, and use it to update the DNN. Fig. 3 shows the ICOA algorithm overview. Our ICOA algorithm consists of three parts corresponding to the following three subsections.

A. Online Offloading Decision Vector Set Generation

We will solve the sub-problem $\mathcal{P}1$ through the following three steps:

(1) Relaxed decision vector generation: We carefully design a three-layer DNN. The DNN adopts the remaining energy $\mathbf{L}^t$, the task data sizes $\mathbf{D}^t$, and the channel gains $\mathbf{h}^t$ as inputs at the beginning of time-frame t and outputs a relaxed decision vector $\hat{\mathbf{z}}^t$, where $\hat{\mathbf{z}}^t \in [0, 1]^n$. This relaxed decision vector can be expressed as:

$$\hat{\mathbf{z}}^{\mathbf{t}} = f_{\omega^t}(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t) \tag{20}$$

where ω^t is the weight of the DNN.

(2) Order-preserving quantization: This paper uses the order-preserving quantization (OPQ) mechanism to generate k offloading decision vectors by quantifying $\hat{\mathbf{z}}^t$ as follows:

$$f_{OPQ} : \hat{\mathbf{z}}^{\mathbf{t}} \mapsto S_{OPQ}^t = \{\mathbf{z}_j^t | \mathbf{z}_j^t \in \{0, 1\}^n, j = 1, 2, \ldots, k\} \tag{21}$$

where S_{OPQ}^t is the set of generated offloading decision vectors. A key feature of the OPQ method is that it maintains the generation order of vectors during quantization. Specifically, if $\hat{z}_{i1}^t \ge \hat{z}_{i2}^t$ then $z_{j,i1}^t \ge z_{j,i2}^t$. By setting the appropriate value for k, we can attain both satisfactory performance and reduced complexity. Following thorough analysis and validation, k is set to n in this paper.

(3) Crossover and variation: We use the crossover and variation mechanisms to obtain more offloading decision vectors. We choose n pairs of vectors from S^t_{OPQ} and set a random intersection point ψ for each selected decision vector pair $(\mathbf{z}^t_i, \mathbf{z}^t_j)$. Then, a new decision vector $\breve{\mathbf{z}}^t_l$ is generated using the following equation:

$$\breve{\mathbf{z}}^t_l = \begin{cases} \mathbf{z}^t_{i,l} & 1 \le l \le \psi \\ \mathbf{z}^t_{j,l} & \psi \le l \le n \end{cases} \tag{22}$$

When the probability of random generation is not greater than a variation rate ϕ, we randomly choose a decision variable $\breve{z}^t_i$ from $\breve{\mathbf{z}}^t_l$ and update $\breve{z}^t_i$ to $\breve{z}^t_i - 1$. Finally, a decision vector set S_{CV} can be implemented that contains up to n new decision vectors. Through the above steps, we can obtain a set $S^t = S^t_{OPQ} \cup S^t_{CV}$, which contains no more than $2n$ candidate decision vectors. Hence, we compress the search space from 2^n to at most $2n$, and significantly reduce the decision search time.

B. Online Resource Allocation Based on Linear Programming

To solve the sub-problem $\mathcal{P}2$, for each $\breve{\mathbf{z}}^t$ in the set S^t, we can divide all WDs into two sets based on it: the set $\mathcal{M}$ of WDs in the edge computing mode and the set $\mathcal{N}$ of WDs in the local computing mode. We substitute $\breve{\mathbf{z}}^t$ into the (13). Combining (3)-(8), the (13) can be expanded into the following equation:

$$\begin{aligned} Q = &\sum_{i\in\mathcal{M}} D^t_i \left(\frac{J_i}{f^l_i} + \frac{\alpha^t_i - 1}{r^t_i} + \frac{J_{ES}}{f_{ES}}\right)\beta^t_i \\ &+ \sum_{i\in\mathcal{M}} \left(\frac{D^t_i}{r^t_i} + \frac{D^t_i c_{ES}}{f_{ES}}\right) \\ &+ \sum_{i\in\mathcal{N}} T^t_{i,l} \end{aligned} \tag{23}$$

The constraints (15) and (16) can be expanded as:

$$\begin{aligned} &\eta^t_i P^t_{ES} h^t_i \mu^t_c - D^t_i \left[a_i J_i + \frac{P_i(\alpha^t_i - 1)}{r^t_i}\right]\beta^t_i \\ &+ L^t_i - \frac{P_i D^t_i}{r^t_i} - L_{\min} \ge 0 \end{aligned} \tag{24}$$

$$\begin{aligned} &\mu^t_c + \sum_{i\in\mathcal{M}} D^t_i \left(\frac{J_i}{f^l_i} + \frac{\alpha^t_i - 1}{r^t_i}\right)\beta^t_i \\ &+ \sum_{i\in\mathcal{M}} \left(\frac{D^t_i}{r^t_i} + \frac{D^t_i c_{ES}}{f_{ES}}\right) - \mu = 0 \end{aligned} \tag{25}$$

It is not difficult to see that the (23) and constraints (24), (25), (17)–(19) are all linear combinations of variables μ^t_c and $\boldsymbol{\beta}^t$. For the convenience of description, we use some symbols to represent some sub terms in (23)–(25):

$$\begin{cases} G_i = D^t_i(\frac{J_i}{f^l_i} + \frac{\alpha^t_i - 1}{r^t_i} + \frac{J_{ES}}{f_{ES}}) \\ H_i = -D^t_i[a_i J_i + \frac{P_i(\alpha^t_i - 1)}{r^t_i}] \\ K_i = \eta^t_i P^t_{ES} h^t_i \\ I_i = D^t_i \left(\frac{1}{r^t_i} + \frac{c_{ES}}{r^t_i}\right) \\ O_i = L^t_i - \frac{P_i D^t_i}{r^t_i} - L_{\min} \end{cases} \tag{26}$$

Then, we will obtain some coefficient matrices:

$$\mathbf{A}' = \begin{bmatrix} K_1 & H_1 & 0 & 0 & \cdot & \cdot & 0 \\ K_2 & 0 & H_2 & 0 & & & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & & & \cdot & \cdot & 0 \\ K_{|\mathcal{M}|} & 0 & \cdot & \cdot & \cdot & 0 & H_{|\mathcal{M}|} \end{bmatrix} \tag{27}$$

where $\mathbf{A}'$ is a $(|\mathcal{M}| \times (|\mathcal{M}| + 1))$ matrix. The other matrix is as follows:

$$\mathbf{c} = [0, G_1, G_2, \ldots, G_{|\mathcal{M}|}]^T \tag{28}$$

$$\mathbf{b}_1 = [O_1, O_2, ..., O_{|\mathcal{M}|}]^T \tag{29}$$

$$\mathbf{q} = [1, G_1, G_2, \ldots, G_{|\mathcal{M}|}] \tag{30}$$

$$\mathbf{r} = [\mu, 1, 1, \ldots, 1]^T \tag{31}$$

$$\theta = \sum_{i\in\mathcal{M}} I_i - \mu \tag{32}$$

After that, we can express the sub-problem $\mathcal{P}2$ as:

$$\min_{\mathbf{X_1}} \quad \mathbf{c}^T\mathbf{X_1} + \sum_{i\in\mathcal{M}} \left(\frac{D^t_i}{r^t_i} + \frac{D^t_i c_{ES}}{f_{ES}}\right) + \sum_{i\in\mathcal{N}} T^t_{i,l} \tag{33}$$

$$\text{s.t.} \quad \mathbf{A}'\mathbf{X_1} + \mathbf{b}_1 \ge 0 \tag{34}$$

$$\mathbf{q}\mathbf{X_1} + \theta = 0 \tag{35}$$

$$\mathbf{0} \le \mathbf{X_1} \le \mathbf{r} \tag{36}$$

where $\mathbf{X}_1 = [\mu^t_c, \beta^t_{(1)}, \beta^t_{(2)}, \ldots, \beta^t_{(|\mathcal{M}|)}]^T$ is a vector consisting of multiple optimization variables for sub-problem $\mathcal{P}2$, and $\beta^t_{(i)}$ is the compression proportion corresponding to each 1in $\breve{\mathbf{z}}^t$.

In order to solve the problem by using linear programming methods, we must standardize it. We denote $\mathbf{X} = [\mu^t_c, \beta^t_{(1)}, \beta^t_{(2)}, \ldots, \beta^t_{(|\mathcal{M}|)}, x_1, x_2, \ldots, x^t_{2|\mathcal{M}|+1}]^T$ as optimization variable, and x_i is the introduced relaxed variable. After omitting last two constant term in (33) and let $\mathbf{c} = [0, G_1, G_2, \ldots, G_{|\mathcal{M}|}, 0, \ldots, 0]^T$, the original problem can be transformed as follows:

$$\min_{\mathbf{X}} \quad \mathbf{c}^T\mathbf{X} \tag{37}$$

$$\text{s.t.} \quad \mathbf{AX} + \mathbf{b} = 0 \tag{38}$$

$$\mathbf{X} \ge 0 \tag{39}$$

where $\mathbf{A}$ is a coefficient matrix and $\mathbf{b}$ is a constant vector, which can be represented as:

$$\mathbf{A} = \begin{bmatrix} -\mathbf{A}' & E_1 & 0 \\ \mathbf{q} & 0 & 0 \\ E_2 & 0 & E_2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -\mathbf{b}_1 \\ \theta \\ -\mathbf{r} \end{bmatrix} \tag{40}$$

where E_1 and E_2 are identity matrices with dimensions of $(|\mathcal{M}| \times |\mathcal{M}|)$ and $((|\mathcal{M}| + 1) \times (|\mathcal{M}| + 1))$, we can know that $\mathbf{A}$ is a matrix with dimensions $((2|\mathcal{M}| + 2) \times (3|\mathcal{M}| + 2))$.

Then, the problem becomes a standard linear programming problem. In short, it can be solved by the simplex method.

Through the above steps, we could obtain several solutions:

$$S_f^t = \{(\mathbf{z}_1^t, \boldsymbol{\beta}_1^t, \mu_{c1}^t), (\mathbf{z}_2^t, \boldsymbol{\beta}_2^t, \mu_{c2}^t), \ldots, (\mathbf{z}_s^t, \boldsymbol{\beta}_s^t, \mu_{cs}^t)\} \quad (41)$$

Then we select the optimal solution $(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t)$ to the original problem. Since we aim to optimize the overall time, we conduct the chosen operation according to the Q. Hence, the solution $(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t)$ with the optimal value Q^* will be chosen as the optimal one:

$$(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t) = \underset{(\mathbf{z}_j^t, \boldsymbol{\beta}_j^t, \mu_{cj}^t) \in S_f^t}{\arg\min} \; Q^*(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t) \quad (42)$$

C. DNN Updating Based on Experience Replay

The last part of the algorithm is the updating of DNN. We will combine the obtained optimal decision vector $\mathbf{z}_*^t$ with the input data of the algorithm at the beginning of the time-frame to form a data sample vector $(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mathbf{z}_*^t)$ and replay it into a replay memory of size $|Y|$. We denote the replay memory as S_{TF}. In this way, a data sample will be placed when the replay memory is full at each time-frame. In this paper, the DNN training utilizes the random replay technique. When the system time experiences an interval ϑ, randomly select some data sample vectors from the replay memory, and the selected set of data sample vectors is represented as $S_{DS}^t = \{(\mathbf{L}_{ti}, \mathbf{D}^{ti}, \mathbf{h}^{ti}, \mathbf{z}_*^{ti}) | ti \in S_{TF}\}$. This paper uses the Adam [38] algorithm to train and update DNN parameters. We use ξ to denote the learning rate, and the cross entropy loss function is:

$$Loss = -\frac{1}{|S_{DS}^t|} \sum_{t \in S_{TF}} [(\mathbf{z}_*^t)^T \log f_{\omega^t}(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t) + (1 - \mathbf{z}_*^t)^T \log(1 - f_{\omega^t}(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t))] \quad (43)$$

This process will continue until convergence.

The pseudocode of the algorithm including the above steps is shown in Algorithm 1. In this algorithm, lines 4-7 correspond to the first module in Fig. 3, lines 8-11 correspond to the second module, and lines 12-15 correspond to the third module.

D. Time Complexity Analysis

Theorem 2: The ICOA algorithm has the time complexity of at most $O((m_1(n + m_2)) + n^3 + |Y|)$ in each time-frame, where m_1 and m_2 represent the number of neurons in the two hidden layers of the DNN, and $|Y|$ is the size of replay memory.

Proof: The main complexity of the ICOA algorithm in each time-frame is analyzed as follows. The line 4 is the generation of the relaxed offloading action vector based on the DNN. We can analyze that this step takes at most $O(m_1(n + m_2))$ to run. The runtime of the OPQ mechanism is at most $O(n^2)$. The crossover and variation in line 6 requires at most $O(n)$ time. The time complexity of traversing the set and using linear programming method in lines 8-10 is $O(n^3)$. Line 11 demands at most $O(n)$ time for execution. The lines 12-15 requires the maximum runtime of $O(|Y|)$. In summary, we can obtain that this theorem holds.

Algorithm 1: ICOA Algorithm.

```
Input: $S_T$,$\mathbf{L}^t$,$\mathbf{D}^t$,$\mathbf{h}^t$,$P_{ES}^t$,$\mathcal{T}$
Output: $\mathbf{z}_*^t$ and $\mu_c^t$ and $\boldsymbol{\beta}^t$
1  Initialize a DNN with the embedded parameters $\omega^t$ and a replay memory;
2  $t \leftarrow 0$;
3  while $t < \mathcal{T}$ do
4      $\hat{\mathbf{z}}^t \leftarrow f_{\omega^t}(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t)$;
5      Invoke the OPQ method to quantize $\hat{\mathbf{z}}^t$ to achieve a decision vector set $S_{OPQ}^t$;
6      Conduct crossover and variation on $S_{OPQ}^t \bigcup \{\vec{z}^t\}$ to achieve a set $S_{CV}^t$ of decision vectors;
7      $S^t \leftarrow S_{OPQ}^t \bigcup S_{CV}^t$;
8      for z in $S^t$ do
9          Standardize the sub-problem $\mathcal{P}2$ to a linear programming problem using Eqs. (23)-(39);
10         Get $\mu_c^t$ and $\boldsymbol{\beta}^t$ by using the simplex method to solve the linear programming problem;
11     $(\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t) \leftarrow \arg\min_{\mathbf{z}_j^t \in S_f^t} Q^*(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mathbf{z}_j^t, \mu_c^t, \boldsymbol{\beta}^t)$;
12     Put $(\mathbf{L}^t$,$\mathbf{D}^t$,$\mathbf{h}^t$,$\mathbf{z}_*^t)$ into the replay memory;
13     if $t \mod \vartheta = 0$ then
14         $S_{DS}^t \leftarrow \{(\mathbf{L}^{t_i}, \mathbf{D}^{t_i}, \mathbf{h}^{t_i}, \mathbf{z}_*^{t_i}) | t_i \in S_{TF}\}$;
15         Train the DNN with $S_{DS}^t$ and renew $\omega^t$ via the Adam algorithm;
16     $t \leftarrow t + 1$;
17     output($\mathbf{z}_*^t, \boldsymbol{\beta}_*^t, \mu_{c*}^t$);
18 return ;
```

V. Performance Evaluation

A. Simulation Setting

The experiments are evaluated based on a publicly available real dataset, referred to EUA [39]. The dataset comprises the geographical coordinates of mobile subscribers and base stations in Melbourne, Australia. We choose the location of a base station as the location of ES, and n user locations around the base station as the locations of WDs. Furthermore, we select three benchmark algorithms for comparison:

- DROO [5]: This algorithm utilizes the DRL and OPQ to generate candidate decision vector.
- EAOO [12]: This algorithm utilizes the OPQ, crossover and variation mechanisms to generate candidate decision vectors. However, it does not utilize data compression.
- LOCAL: This algorithm relies entirely on local computing.

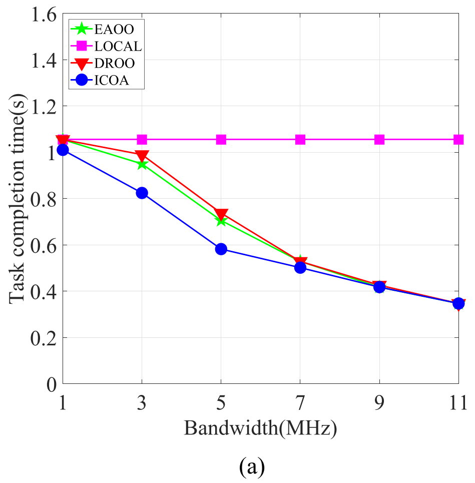
Table II lists the range of values of the evaluation parameters used in these experiments.

B. Impact of Bandwidth

Fig. 4(a) shows the average task completion time of the four algorithms when the bandwidth is increasing. We can observe that the average task completion time of the DROO, EAOO, and ICOA algorithms is decreasing, and the LOCAL task completion time remains unchanged. This is because as the bandwidth increases, the data transmission speed also increases, which

TABLE II
EVALUATION PARAMETER SETTINGS

Parameter	Value
n	10∼30
λ	2 s
η_i^t	0.6∼0.8
L_{min}	10∼20 J
h_i^t	$10^{-3} \sim 10^{-1}$
P_{ES}^t	50 W
f_i^l	150∼200 MHz
c_i	2∼3
D_i^t	100∼150 Mb
B	10∼30 MHz
P_i	30∼40 mW
N_0	$10^{-10} \sim 10^{-9}$ W
ξ	0.0001-0.1
$\lvert Y \rvert$	64-512
ϑ	5-20



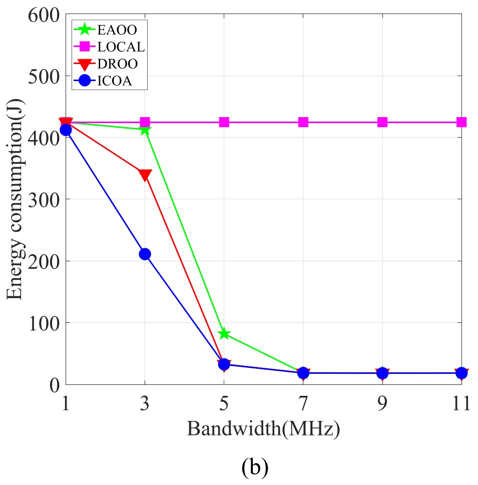
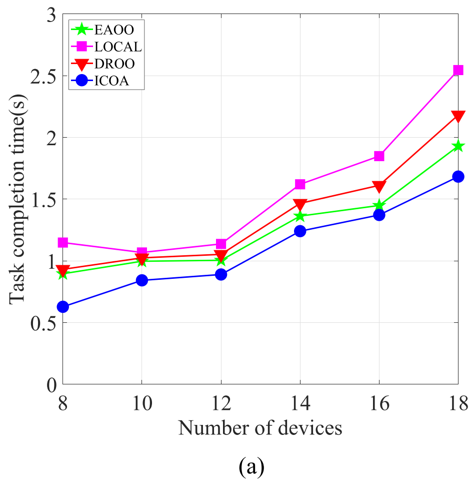


Fig. 4. Task completion time and energy consumption under different channel bandwidths



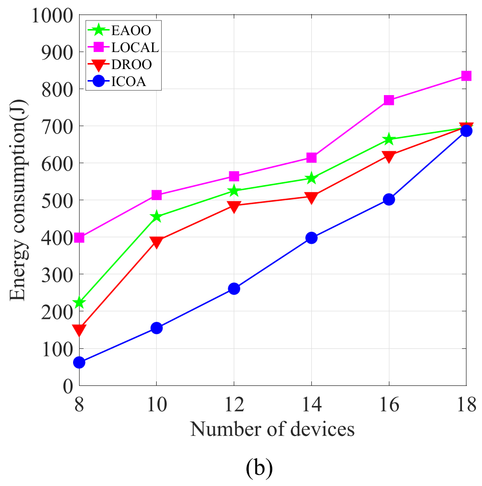


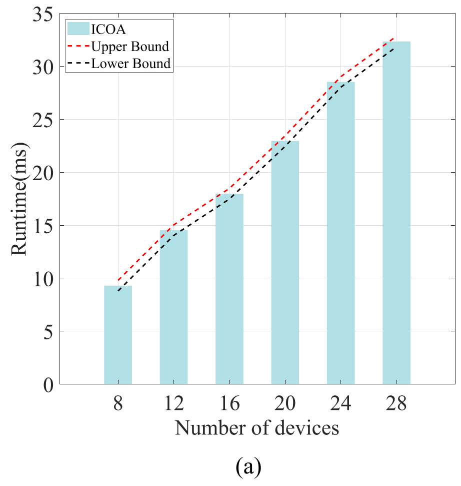
Fig. 5. Task completion time and energy consumption under different number of devices.

reduces the offloading transmission time and, consequently, the total task completion time. The LOCAL algorithm remains unchanged because it executes the task entirely locally. Therefore, it is not affected by the increase in bandwidth. Our ICOA algorithm is the most effective due to the introduction of data compression which reduces the amount of data transmitted. When the bandwidth reaches a certain value 5 MHz, ICOA, DROO and EAOO have the same effect, this is because as the bandwidth increases, the decrease in data size is not enough to compensate for the time consumed by data compression itself, which encourages ICOA to choose not to compress at higher bandwidth.

As the bandwidth continues to increase, the average energy consumption of the four algorithms is shown in Fig. 4(b). We can see that when the bandwidth is less than 5 MHz, the energy consumption of ICOA is lower than that of other algorithms. This is because transmission energy consumption makes up the majority of total energy consumption, and data compression can effectively reduce it.

C. Impact of the Number of Devices

We compare the performance of four algorithms with the increasing of the number of devices. We increase the number of devices from 8 to 18 and observe their average task completion time and energy consumption, as shown in Fig. 5(a) and (b), respectively. We can see that as the number of devices increases, the average task completion time and energy consumption of the four algorithms also increase. This is because the greater the number of devices, the more tasks need to be completed, which leads to increased time and energy consumption. We can see that



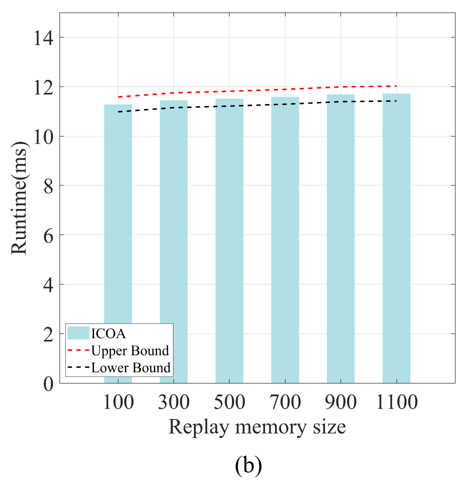
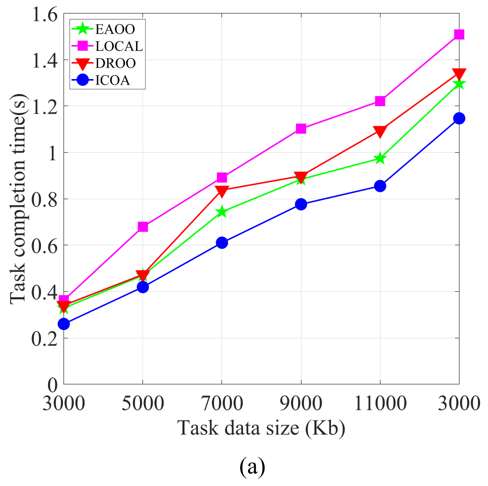


Fig. 6. The runtime of ICOA algorithm.



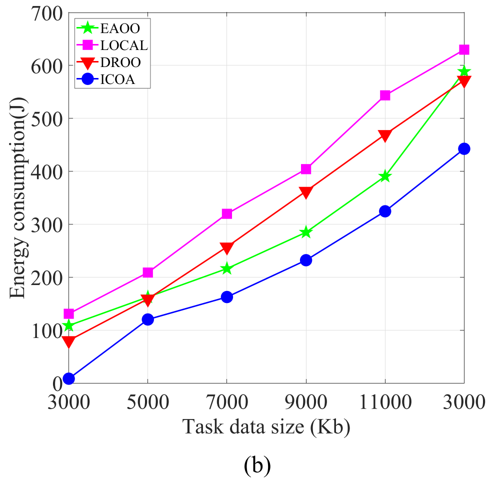


Fig. 7. Comparison of task completion time and energy consumption in different data sizes.

our algorithm outperforms the other three benchmark algorithms in terms of task completion time and energy consumption. This is because we have introduced data compression, which not only reduces data transmission time but also greatly reduces energy consumption during task offloading transmission.

D. Running Time

In this section, we examine the impact of the number of devices and the size of the replay memory on the algorithm's running time at each time-frame. From Theorem 2 we know the time complexity of ICOA algorithm is $O((m_1(n+m_2))+n^3+|Y|)$. Fig. 6(a) shows the variation of the running time of the algorithm as the number of devices continues to increase. We can see that as the number of devices increases from 8 to 28, the maximum running time of the algorithm does not exceed 35 milliseconds. Fig. 6(b) shows the variation of the running time with the size of the replay memory. We can see that its impact on the running time is relatively small. Therefore, we can observe that our ICOA algorithm meets the real-time requirement.

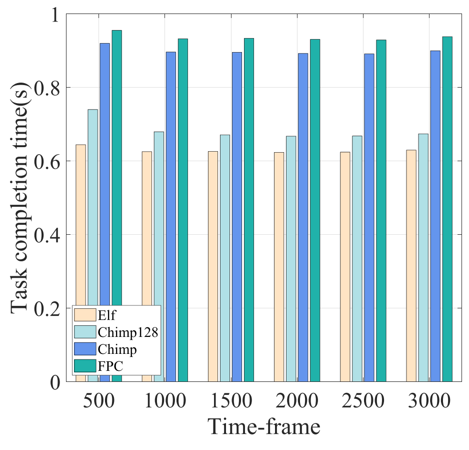
E. Impact of the Size of Data

Fig. 7 clearly indicates that, as the data size of the task increases, all algorithms generate more task completion time and energy consumption. This is because the larger the task size, the more computation is required for each task, resulting in more energy consumption and more task completion time. Among all four algorithms, we can observe that our ICOA algorithm has the least task completion time and energy consumption compared to other competing algorithms.

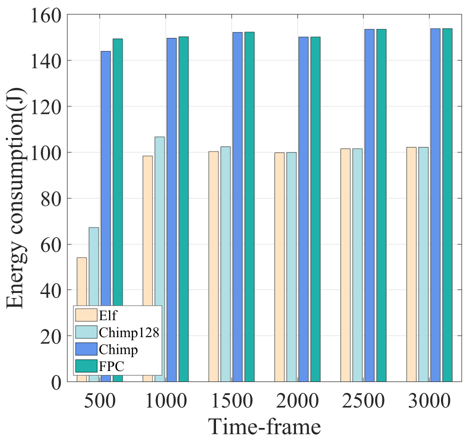
F. Impact of Different Compression Algorithms

In this section, we test the impact of using different compression algorithms on task completion time and energy consumption. The four compression algorithms are:

- *Elf [13]:* A lightweight floating-point compression algorithm based on delta encoding and variable-length integer compression, which dynamically adjusts window sizes to achieve low-overhead data reduction.
- *Chimp128 [14]:* An enhanced 128-bit streaming compression algorithm that detects data patterns through XOR operations and combines leading-zero counting with dynamic bitmasking for high compression ratios.
- *Chimp [14]:* A real-time floating-point compression method using XOR-based pattern matching, which compares neighboring values via a sliding window and

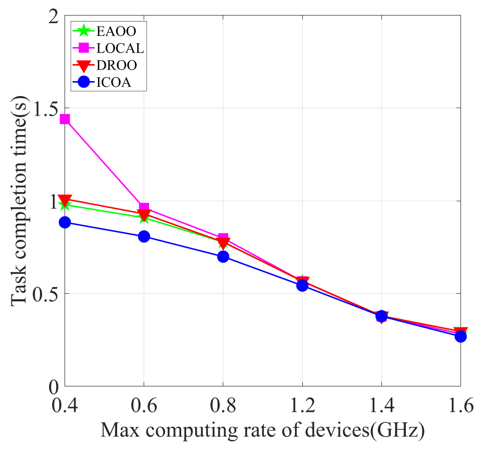


(a)

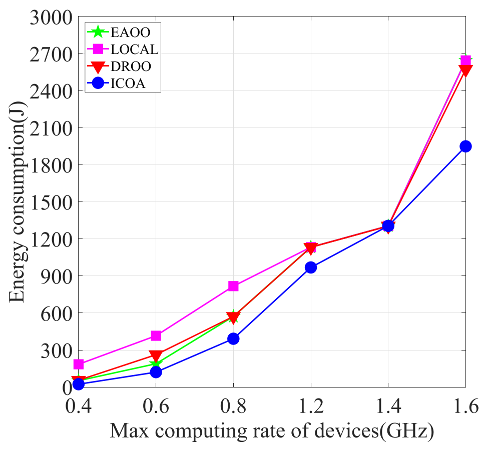


(b)

Fig. 8. Comparison of task completion time and energy consumption in different compression algorithms.



(a)



(b)

Fig. 9. Comparison of task completion time and energy consumption in different max culculate rate of devices.

employs leading/trailing zero prediction with variable-length encoding for fast lossless compression.

- *FPC [15]:* A hardware-optimized floating-point compressor utilizing a three-stage pipeline (scalar quantization, bit-plane splitting, and entropy coding) to enable high-speed lossless compression for scientific datasets.

We test 3000 time-frames and take the average results of every 500 time-frames as the final results, which are shown in Fig. 8. We can observe that, when Elf is used, the algorithm has the lowest task completion time and energy consumption. This is because Elf has the best compression effects among all the compression algorithms.

G. Impact of Max Computing Rate

As illustrated in Fig. 9, an investigation was conducted to assess the impact of the max computing rate variation of the device on the performance of four distinct algorithms. It is evident that, as the max computing rate of the device increases, the total task completion time of four algorithms exhibits a notable decline, whereas the energy consumption rises considerably. This phenomenon can be attributed to the fact that as the computing rate rises, the time required for tasks to be computed locally is significantly reduced. However, the computing energy consumption is positively correlated with the computing rate, and thus an enhancement in the computing rate also entails an increase in energy consumption.

H. Approximate Ratio

In this section, in order to clarify the relationship between our method and the theoretical optimal solution, we evaluate the approximation ratio σ of our algorithm ICOA at different time-frames and compare it with three benchmark algorithms. We define the approximation ratio as follows:

$$\sigma = \frac{Q^*_{enum}(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t)}{Q^*(\mathbf{L}^t, \mathbf{D}^t, \mathbf{h}^t, \mu_c^t, \mathbf{z}^t, \boldsymbol{\beta}^t)} \tag{44}$$

where Q^*_{enum} is the theoretical optimal value obtained by the enumeration method. This indicates that the closer the approximation ratio is to 1, the better the algorithm performs. From the Fig. 10, We can see that our ICOA algorithm significantly outperforms the benchmark algorithms, and achieves an approximate ratio above 0.8. Moreover, our ICOA algorithm can

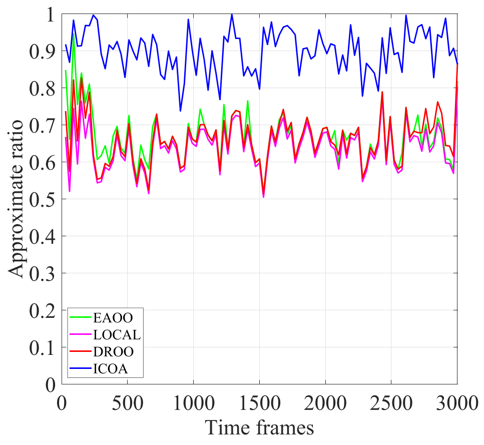
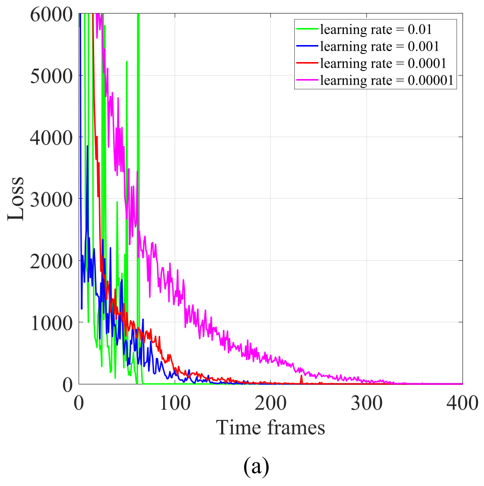


Fig. 10. Approximate ratio comparison of four algorithms.



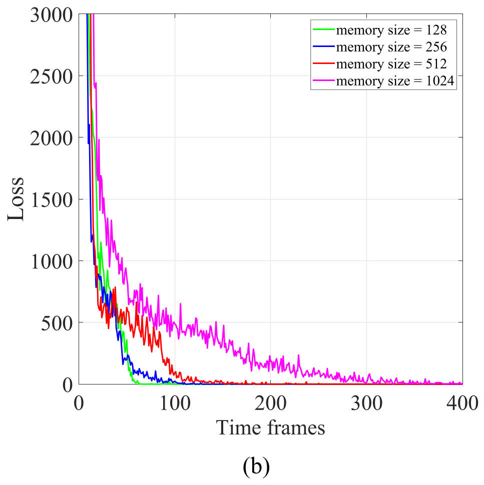


Fig. 11. Training loss of DNN.

sometimes achieve an approximate ratio of 1, which indicates that its performance is near optimal.

I. Training Loss

We analyze the training loss of ICOA with varying parameters. This paper presents a measurement of the training loss at varying learning rates, as illustrated in Fig. 11(a). From this figure, we can observe that the maximum learning rates of 0.01 and 0.0001 result in a significant deviation in the performance curve, whereas a learning rate of 0.001 produces the most stable performance. All of the aforementioned curves ultimately converge to a value of approximately 0.2. Subsequently, the size of the relay memory is modified, and the resulting experimental outcomes are illustrated in Fig. 11(b). ICOA exhibits disparate convergence curves contingent on the relay memory size. It is evident that a memory size of 256 culminates in the most stable convergence curve, whereas other replay memory sizes can engender considerable performance jitter. This phenomenon can be attributed to the fact that smaller relay memory sizes can precipitate frequent updates of data samples, whereas larger relay memory sizes can give rise to delayed updates, both of which are inimical to convergence performance.

VI. Conclusion

This paper proposes an adaptive intelligent compression offloading algorithm ICOA for WPMEC. The ICOA algorithm fully utilizes the lossless data compression technique to effectively improve the efficiency of the task offloading process. The DRL technique and the simplex method are adopted to well resolve the joint optimization problem of offloading decision generation, charging time allocation and compression ratio allocation. The simulation results on a real dataset show that, the proposed ICOA algorithm has significant advantages in reducing task completion time and energy consumption. Furthermore, ICOA algorithm can achieve near optimal performance with low runtime, providing strong technical support for satisfying the real-time and efficiency requirements of WPMEC. In future, we will consider optimizing the other performance metrics (such as throughput and energy consumption) by introducing adaptive power control algorithms under adjustable power conditions.

References

[1] H.-H. Chen, S. Fortes, I. Haque, A. R. Hussein, S. J. Johnson, and T. Maksymyuk, “Series editorial: Internet of Things,” *IEEE Commun. Mag.*, vol. 62, no. 7, pp. 42–43, Jul. 2024.

[2] B. Lin, J. Weng, X. Chen, Y. Ma, and C.-H. Hsu, “A game-based computation offloading with imperfect information in multi-edge environments,” *IEEE Trans. Services Comput.*, vol. 18, no. 1, pp. 1–14, Jan./Feb. 2025.

[3] C. Wang, X. Chai, S. Peng, Y. Yuan, and G. Li, “Deep reinforcement learning with entropy and attention mechanism for D2D-assisted task offloading in edge computing,” *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3317–3329, Nov./Dec. 2024.

[4] J. Du, M. Xu, S. S. Gill, and H. Wu, “Computation energy efficiency maximization for intelligent reflective surface-aided wireless powered mobile edge computing,” *IEEE Trans. Sustain. Comput.*, vol. 9, no. 3, pp. 371–385, May/Jun. 2024.

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

[6] M. Bolourian and H. Shah-Mansouri, “Energy-efficient task offloading for three-tier wireless-powered mobile-edge computing,” *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10400–10412, Jun. 2023.

[7] X. Wang et al., “Wireless powered mobile edge computing networks: A survey,” *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–37, 2023.

[8] S. Pang, L. Wang, H. Gui, S. Qiao, X. He, and Z. Zhao, “UAV-IRS-assisted energy harvesting for edge computing based on deep reinforcement learning,” *Future Gener. Comput. Syst.*, vol. 163, 2025, Art. no. 107527.

[9] Y. Deng et al., “Semi-asynchronous federated learning with trajectory prediction for vehicular edge computing,” in *Proc. IEEE/ACM 32nd Int. Symp. Qual. Service*, 2024, pp. 1–10.

[10] G. Wu, X. Chen, Z. Gao, H. Zhang, S. Yu, and S. Shen, “Privacy-preserving offloading scheme in multi-access mobile edge computing based on madrl,” *J. Parallel Distrib. Comput.*, vol. 183, 2024, Art. no. 104775.

[11] Y. Chen, J. Zhao, J. Hu, S. Wan, and J. Huang, “Distributed task offloading and resource purchasing in NOMA-enabled mobile edge computing: Hierarchical game theoretical approaches,” *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 1, pp. 1–28, 2024.

[12] X. Jiao et al., “Deep reinforcement learning empowers wireless powered mobile edge computing: Towards energy-aware online offloading,” *IEEE Trans. Commun.*, vol. 71, no. 9, pp. 5214–5227, Sep. 2023.

[13] R. Li, Z. Li, Y. Wu, C. Chen, and Y. Zheng, “Elf: Erasing-based lossless floating-point compression,” *Proc. VLDB Endowment*, vol. 16(7), pp. 1763–1776, 2023.

[14] P. Liakos, K. Papakonstantinopoulou, and Y. Kotidis, “Chimp: Efficient lossless floating point compression for time series databases,” *Proc. VLDB Endowment*, vol. 15, no. 11, pp. 3058–3070, 2022.

[15] M. Burtscher and P. Ratanaworabhan, “High throughput compression of double-precision floating-point data,” in *Proc. 2007 Data Compression Conf.*, 2007, pp. 293–302.

[16] W. Xiao, Y. Hao, J. Liang, L. Hu, S. A. Alqahtani, and M. Chen, “Adaptive compression offloading and resource allocation for edge vision computing,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 6, pp. 2357–2369, Dec. 2024.

[17] K. Cheng, X. Fang, and X. Wang, “Energy efficient edge computing and data compression collaboration scheme for uav-assisted network,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16395–16408, Dec. 2023.

[18] B. Han, Y. Ye, L. Shi, Y. Xu, and G. Lu, “Energy-efficient computation offloading for MEC-enabled blockchain by data compression,” in *Proc. 2024 IEEE/CIC Int. Conf. Commun. China*, 2024, pp. 1970–1975.

[19] N. Li, A. Iosifidis, and Q. Zhang, “Attention-based feature compression for cnn inference offloading in edge computing,” in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 967–972.

[20] W. Tu and X. Liu, “Energy consumption minimization for a data compression based NOMA-MEC system,” in *Proc. 5th Inf. Commun. Technol. Conf.*, 2024, pp. 337–343.

[21] Y. Chen, J. Xu, Y. Wu, J. Gao, and L. Zhao, “Dynamic task offloading and resource allocation for noma-aided mobile edge computing: An energy efficient design,” *IEEE Trans. Services Comput.*, vol. 17, no. 4, pp. 1492–1503, Jul./Aug. 2024.

[22] A. Fresa and J. PrakashChampati, “Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 7, pp. 2025–2039, Jul. 2023.

[23] H. Peng, Y. Zhan, D.-H. Zhai, X. Zhang, and Y. Xia, “Egret: Reinforcement mechanism for sequential computation offloading in edge computing,” *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3541–3554, Nov./Dec. 2024.

[24] X. Cao, S. Wang, and X. Wu, “Energy-efficient resource allocation in intelligent reflecting surface aided wireless powered mobile edge computing systems,” in *Proc. 2024 IEEE Int. Conf. Commun. Workshops*, 2024, pp. 840–845.

[25] Y. Lu and J. Zhao, “Reconfigurable intelligent surface aided wireless powered mobile edge computing,” in *Proc. IEEE 20th Consum. Commun. Netw. Conf.*, 2023, pp. 501–507.

[26] X. Zhu, F. Hao, L. Ma, C. Luo, G. Min, and L. T. Yang, “DRL-based joint optimization of wireless charging and computation offloading for multi-access edge computing,” *IEEE Trans. Services Comput.*, vol. 18, no. 3, pp. 1352–1367, May/Jun. 2025.

[27] X. Dong, Z. Wan, and C. Deng, “Optimization of wireless power transfer for wireless-powered mobile edge computing,” in *Proc. 2022 Int. Conf. Comput. Commun. Netw.*, 2022, pp. 1–7.

[28] J. Wen et al., “Diffusion-model-based incentive mechanism with prospect theory for edge AIGC services in 6G IoT,” *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34187–34201, Nov. 2024.

[29] R. Malik and M. Vu, “Energy-efficient joint wireless charging and computation offloading in MEC systems,” *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 5, pp. 1110–1126, Aug. 2021.

[30] Y. Zhong et al., “Generative diffusion-based contract design for efficient ai twin migration in vehicular embodied AI networks,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 5, pp. 4573–4588, May 2025.

[31] P. Chen, B. Lyu, H. Guo, and Z. Yang, “Sum computational bits maximization for full-duplex enabled wireless-powered mobile edge computing systems. In *2022 IEEE/CIC Int. Conf. Commun. China (ICCC)*, pp. 256–261, 2022.

[32] M. Poposka, Z. Hadzi-Velkov, and T. Shuminoski, “Proportional fairness in wireless powered mobile edge computing networks. In *2023 30th Int. Conf. Syst. Signals Image Process.*, 2023, pp. 1–5.

[33] X. He, Y. Shen, X. Wang, S. Wang, S. Xu, and J. Ren, “Online scheduling for energy minimization in wireless powered mobile edge computing,” in *Proc. 2022 IEEE Wireless Commun. Netw. Conf.*, 2022, pp. 1146–1151.

[34] X. Zhang, L. Wang, X. Wu, L. Xu, and A. Fei, “Energy-efficient computation offloading and data compression for UAV-mounted MEC networks,” in *Proc. IEEE Glob. Commun. Conf.*, 2023, pp. 6904–6909.

[35] B. Lai et al., “Resource-efficient generative mobile edge networks in 6G era: Fundamentals, framework and case study,” *IEEE Wireless Commun.*, vol. 31, no. 4, pp. 66–74, Aug. 2024.

[36] Y. Zheng, T. Zhang, R. Huang, and Y. Wang, “Computing offloading and semantic compression for intelligent computing tasks in MEC systems,” in *Proc. 2023 IEEE Wireless Commun. Netw. Conf.*, 2023, pp. 1–6.

[37] 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 Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[38] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1018–1026.

[39] P. Lai et al., “Optimal edge user allocation in edge computing with variable sized vector bin packing,” in *Proc. 16th Int. Conf. Service-Oriented Comput.*, Hangzhou, China, 2018, pp. 230–245 .

Xianlong Jiao (Member, IEEE) received the BE, ME and PhD degrees in computer science and technology from the National University of Defense Technology, Changsha, China, in 2003, 2005 and 2011, respectively. He was an exchanging student with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, from 2009 to 2010. He was a postdoctor with the College of Information System and Management, National University of Defense Technology, Changsha, China, from 2012 to 2015. He is currently a research fellow with the College of Computer Science, Chongqing University, Chongqing, China. His current research interests include the Internet of Things and mobile edge computing.

Yunhui Chen received the bachelor degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2022. He is currently working toward the master degree in computer science and technology with the College of Computer Science, Chongqing University, in 2023 His current research interests include the Internet of Things and mobile edge computing.

Xiang Wu received the bachelor degree in computer science and technology from Changjiang Normal University, Chongqing, China, in 2023. He is currently working toward the master degree in computer science and technology with the College of Computer Science, Chongqing University, in 2023. His current research interests include the Internet of Things, and mobile edge computing.

Ruiyuan Li received the BE and MS degrees from Wuhan University, China, in 2013 and 2016, respectively, and the PhD degree from Xidian University, China, in 2020. He is an associate professor with Chongqing University, China. He is the director of Start Lab (Spatio-Temporal Art Lab). He was the Head of Spatio-Temporal Data Group in JD Intelligent Cities Research, leading the research and development of JUST (JD Urban Spatio-Temporal data engine). Before joining JD, he had interned in Microsoft Research Asia from 2014 to 2017. His research interests include focuses on spatio-temporal data management and urban computing.

Songtao Guo (Senior Member, IEEE) received his the BS, MS and PhD degrees in computer software and theory from Chongqing University, Chongqing, China, in 1999, 2003 and 2008, respectively. He was a professor with Chongqing University from 2011 to 2012, and a professor with Southwest University from 2013 to 2018. At present, he is a full professor with Chongqing University, China. He was a senior research associate with the City University of Hong Kong from 2010 to 2011, and a visiting scholar with Stony Brook University, New York, USA, from May 2011 to May 2012. His research interests include mobile edge computing, federal learning, wireless sensor networks, and wireless ad hoc networks.

Yong Ma received the MS degree in computer science from Xidian University, in 2003 and, the PhD degree in computer science from Wuhan University, in 2006. In 2018, he worked on the integrated control and dispatching of energy in microgrid with Mälardalens University, Sweden. He is currently a professor with the School of Computer Information Engineering, Jiangxi Normal University. His current research interests include cloud computing, edge computing, and data science.

Jiannong Cao (Fellow, IEEE) received the BSc degree in computer science from Nanjing University, China, in 1982, and the MSc and PhD degrees in computer science from Washington State University, USA, in 1986 and 1990, respectively. He is currently the Otto Poon Charitable Foundation Professor of data science and the chair professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He has published five coauthored, nine co-edited books, and more than 500 papers in major international journals and conference proceedings. His research interests include distributed systems and blockchain, wireless sensing and networking, Big Data and machine learning, and mobile cloud and edge computing. He is a member of Academia Europaea and an ACM Distinguished Member.