Stackelberg Game-Based Bandwidth Allocation and Resource Pricing for Multiuser in MEC System

Zhao Tong, *Senior Member, IEEE*, Yuanyang Zhang, Jing Mei, *Member, IEEE*, Wei Ai, Kenli Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—With the rapid development of artificial intelligence, a substantial number of computing-intensive applications have emerged in Internet of Things (IoT) devices. The mobile-edge computing (MEC) architecture enables the provision of abundant computing and storage resources in close proximity to end users (EUs), thereby effectively enhancing their Quality of Experience (QoE). Nonetheless, both the MEC server and EUs are self-interests, it is crucial to establish suitable incentive mechanism to promote active engagement from both parties in the offloading process. Therefore, we employ the Stackelberg game to describe the interaction process between EUs and the MEC server, and an optimal relationship between bandwidth and offloading task size is established to simplify the decision problem for EUs. Then, the optimal strategies for the MEC server and EUs are solved using reverse induction. Given the limited resources of the MEC server, we propose a dynamic programming-based resource allocation (DPRA) algorithm to maximize the revenue of the MEC server while ensuring the cost of each EU. The simulation results demonstrate that the DPRA algorithm can reduce latency and energy consumption costs, significantly outperforming other comparative strategies in terms of performance at both EUs and the MEC server.**

***Index Terms*—Dynamic bandwidth, resource allocation, resource pricing, Stackelberg game, task offloading.**

Manuscript received 8 November 2023; revised 20 March 2024; accepted 3 April 2024. Date of publication 11 April 2024; date of current version 26 June 2024. This work was supported in part by the Program of National Natural Science Foundation of China under Grant 62372172 and Grant 62072174; in part by the Distinguished Youth Science Foundation of Hunan Province, China, under Grant 2023JJ10030; in part by the National Natural Science Foundation of Hunan Province, China, under Grant 2022JJ40278 and Grant 2020JJ5370; and in part by the Scientific Research Fund of Hunan Provincial Education Department, China, under Grant 22A0026. *(Corresponding author: Jing Mei.)*

Zhao Tong, Yuanyang Zhang, and Jing Mei are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, Hunan, China (e-mail: tongzhao@hunnu.edu.cn; 202220294007@hunnu.edu.cn; Jingmei1988@163.com).

Wei Ai is with the College of Information Science and Engineering, Central South University of Forestry and Technology, Changsha 410004, Hunan, China (e-mail: aiwei@hnu.edu.cn).

Kenli Li is with the College of Information Science and Engineering, Hunan University, Changsha 410012, China (e-mail: lkl@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA, and also with the College of Information Science and Engineering, Hunan University Changsha 410012, China (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2024.3387440

I. Introduction

THE Internet of Things (IoT) is a crucial field in the 5G era with vast application prospects. It can connect various devices and objects, providing convenient, efficient, and secure service experiences. The IoT technology has penetrated into all aspects of human life, such as smart healthcare, intelligent manufacturing, autonomous driving cars, virtual reality, and so on [1]. It is predicted that the global IoT terminal count is expected to reach 25 billion by 2025. With the rapid development of artificial intelligence (AI), the demands for computing-intensive applications are growing. These applications require more powerful and efficient computing and storage resources to support complex tasks, such as deep learning and big data processing [2]. Nonetheless, the physical size of current devices is constrained, which in turn also limits the computational capability and battery life of mobile devices. As a result, it becomes challenging to ensure that end users (EUs) receive the necessary Quality of Experience (QoE) [3]. Traditional cloud computing systems possess strong computing and storage resources. However, the performance of highly time-sensitive task applications cannot always be effectively guaranteed. This is because such applications require data transmission over long distances, which may cause latency issues [4].

Mobile-edge computing (MEC), as a new paradigm arising from the convergence of mobile and cloud computing, deploys computing and storage resources on edge servers (ESs) closer to EUs, significantly reducing the amount of tasks sent to remote clouds. By reducing the resource consumption of user-owned devices, MEC provides better QoE for EUs. Compared to cloud computing, MEC has several advantages, including high reliability, low latency, low bandwidth demand, and data privacy protection [5]. However, the MEC server typically does not have as much resources as cloud computing center. If a significant number of EUs offload their computing tasks to the MEC server with limited computing and bandwidth resources, it may result in wireless link congestion. This congestion, in turn, negatively impacts the efficiency of task offloading. Therefore, proper allocation and management of wireless resources are crucial to ensure a smooth and efficient task offloading process. To achieve this goal, joint optimization of computing and related wireless resource allocation is necessary [6]. Meanwhile, the MEC server is unwilling to engage in the task offloading process without reward. Developing a reasonable pricing strategy to value the resources is necessary.

Therefore, it is essential to establish an appropriate incentive mechanism between the MEC server and EUs for resources allocation and pricing [7].

Some researchers have designed incentive strategies, such as market-based pricing, auction theory-based, and game theory-based. Among them, game theory serves as a mathematical instrument for analyzing the interaction among multiple decision makers. Game theory considers the predicted behavior and actual behavior of individuals in the game and explores their optimal strategies. In particular, the Stackelberg game is suitable for scenarios where game participants are in an unequal status [8]. In this situation, the leader has a leadership advantage and can take the initiative while followers need to make decisions after the leader. All game participants are self-interested and compete with each other to maximize their own interests. In this article, the MEC server, as a leader, will price its resources and sell them to EUs in need, while EUs will adjust the amount of resources purchased based on the given price. In addition, to our knowledge, there is a dearth of prior research on the joint optimization of bandwidth and computing resources based on Stackelberg game in MEC. Moreover, previous research work often applied algorithms to formalized problems, which resulted in high complexity when iteratively executing multiple algorithms. The main contributions are summarized as follows.

1) The interaction process between the MEC server and EUs is modeled as a Stackelberg game, in which the MEC server acts as a leader and maximizes its revenue by setting the price of bandwidth resources, while EUs act as followers and minimize their costs by deciding how much bandwidth to purchase.
2) A quantitative relationship between optimal bandwidth and offloading task size is established. The EUs' multivariable offloading decision problem is transformed into a single-variable optimization problem, and convex optimization is used to determine the optimal bandwidth for EUs.
3) Reverse induction is used to analyze the Stackelberg game, demonstrating the existence of a distinct Stackelberg equilibrium between the MEC server and EUs. Subsequently, a dynamic programming-based resource allocation (DPRA) algorithm is proposed for achieving the optimal solution.
4) A substantial amount of experiments are carried out to substantiate the efficacy of the proposed DPRA algorithm. The results of the experiments show the positive impact of the DPRA algorithm in reducing the costs incurred by EUs and enhancing the revenue earned by the MEC server, leading to a mutually beneficial outcome for both parties.

The remaining organization of this article is as follows. Section II presents the relevant literature in this field. In Section III, the system model is introduced. After giving the problem formulation in Section IV, the algorithm design and analysis is proposed in Section V. Section VI includes the parameter settings and performance evaluation. Finally, Section VII provides a summary of the work in this article, along with a discussion on future directions for further exploration.

II. Related Work

In this section, the optimization objective and offloading schemes in the MEC architecture are discussed, followed by resource allocation strategy based on economic theory.

A. Optimization Objective

The optimization objective is a critical element in MEC architecture. To ensure the QoE of EUs, optimization objective is typically quantified to express the EUs' experience and perception of the current network. Research often focuses on optimizing two aspects: 1) latency and 2) energy consumption. In [9], a combination of traditional optimization methods and deep reinforcement learning was applied by Zhu et al. to rapidly acquire the optimal offloading solution for tasks, aiming to minimize task computation latency. Gu et al. [10] investigated a UAV-MEC system that incorporated energy harvesting capabilities. In order to achieve an offloading solution that was both energy-efficient and secure, it transformed the optimization objective into a univariate convex problem. This algorithm helped alleviate the energy constraint issues of UAVs. The optimization objective of [11], [12], and [13] was minimizing latency and energy consumption. Tong et al. [11] put forth an optimization algorithm leveraging Lyapunov optimization technique, which aimed to minimize the overall energy consumption while ensuring long-term system stability within a specified time frame. Zhang et al. [12] utilized a deep neural network to acquire the optimal correlation between offloading ratios and wireless channels, while employing convex optimization to derive the solution for the subproblem of resource allocation. The proposed algorithm exhibited superiority in minimizing overall system energy consumption and achieving low processing latency. Chen et al. [13] proposed cloud–edge collaborative mobile computing offloading mechanism that integrated deep reinforcement learning. This mechanism considered both computational resources and offloading decisions. It featured fast convergence, strong stability, and the ability to achieve optimal offloading decisions with the lowest overall cost. In [14], a reputation and voting-based consensus mechanism was proposed by Liao et al. in MEC blockchain systems, which showed excellent performance concerning time consumption and consensus security.

The aforementioned research mainly focused on the technical aspect of resource allocation with the objective of striking a balance between latency and energy consumption. However, an economic perspective has not been considered. They assumed that the MEC server would selflessly provide services to EUs, without optimizing for both the MEC server and EUs simultaneously.

B. Offloading Schemes

In current research on MEC architecture, there are mainly two offloading schemes for computation tasks at the user end,

namely, binary offloading and partial offloading. Specifically, binary offloading refers to the task being treated as a whole, where it is either completely offloaded to the MEC server or executed entirely on the local device. Saleem et al. [15] was based on MEC enabled device-to-device collaboration, jointly addressing task allocation and power allocation, utilizing proximity-aware binary task offloading to accelerate task execution. Alghamdi et al. [16] introduced a binary computing offloading decision and then solved the offloading sequence decision problem using the principle of optimal stopping theory. Lin et al. [17] introduced a PDDQNLP algorithm to optimize energy efficiency and task offloading fairness in UAV-assisted MEC, demonstrating superior performance in single-UAV scenarios. Wu et al. [18] presented a binary offloading framework leveraging Lyapunov optimization and actor–critic networks to minimize energy consumption, and protect user privacy. In contrast to binary offloading, partial offloading can split the computation task into two parts, which can be executed on the local decides and the MEC server, respectively. Wu et al. [19] proposed a delay-aware energy-efficient task offloading algorithm that utilized virtual queues and perturbed Lyapunov optimization to minimize energy consumption and maintain low latency. Abouaomar et al. [20] proposed a resource representation scheme that utilized the Lyapunov optimization framework for dynamically allocating resources to edge devices. The proposed method offered low latency and optimal resource utilization services, outperforming other benchmark methods. Qu et al. [21] proposed a DMRO algorithm for efficient task offloading in heterogeneous IoT–edge–cloud computing environments. It combined deep reinforcement learning, meta-learning, and distributed computing to achieve optimal offloading decision making.

Our study belongs to the category of partial offloading. In addition, compared to the aforementioned research, our tasks can be executed concurrently on both local devices and the MEC server. This approach effectively reduces EUs time costs and improves execution efficiency.

C. Economic Pricing Strategy

Resource pricing contributes to achieving optimal resource allocation, thereby enhancing economic efficiency. Auction theory and Game theory are commonly regarded as the primary and most frequently employed methods in the field of resource pricing. Le et al. [22] proposed an auction-based framework for bandwidth trading in colocation MEC systems, which considers the tradeoff between delay and energy consumption. Luong et al. [23] introduced an optimal auction design utilizing deep learning techniques, and the proposed approach could offer a valuable tool for optimizing resource allocation in fog networks at large. In [24], a real combinatorial auction mechanism was employed by Su et al. to incentivize devices and edge clouds to participate in offloading service transactions. Unlike auction theory, game theory can investigate resource allocation problems involving multiple participants, including cooperative games, competitive games, and situations where cooperation and competition coexist. Furthermore, game theory has advantages in terms of interactions and dynamic changes, enabling better solutions to resource allocation issues. On the basis of noncooperative game theory, Li et al. [25] introduced a fine-grained computing offloading algorithm to analyze the interaction between the local and edge sides. Hossain et al. [26] combined vehicular edge computing with noncooperative game theory, allowing each vehicle to design its own offloading strategy to compete for resources and maximize utility. The noncooperative game mentioned above required the assumption that all participants disclose their strategies simultaneously, which is not always valid in practice. In contrast, the Stackelberg game enables participants to disclose their strategies in a predetermined order. Zhou et al. [27] adopted a Stackelberg game model to depict the relationship between cloud server (CS) and ES, enabling CS to allocate computing tasks to ES with available computing resources, thus reducing its own workload and costs. Liu and Fu [28] analyzed the interaction between users and MEC server based on the Stackelberg game, aiming to address the issue of equal distribution of edge cloud computing resources. However, this study did not consider the energy consumption of user devices. Applying the Stackelberg game to the three-tier architecture of cloud–edge–terminal, Chang and Wei [29] achieved energy sustainability and profit maximization. However, the tasks on the user side and the cloud–edge side cannot be executed simultaneously in this research. In [30], the interaction between the MEC server and users was modeled as a Stackelberg game by Tong et al. and the optimal solution was obtained using a PSO-based algorithm. However, this study only considered the optimization of a single resource and used equal distribution for communication resources.

Table I presents a comparison of related works.

III. System Model

A brief overview of the MEC model's scenario is provided at the beginning of this section. Furthermore, the local computing model and the process of offloading tasks for execution on the MEC server are discussed.

A. MEC Model

In this part, we consider a MEC model consisting of multiple EUs and an MEC server, as depicted in Fig. 1. The EUs are denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$. During an offloading period, EUs transmit their tasks to the MEC server through wireless communication links. The MEC server located in the region receives tasks and provides the computed results back to the EUs after task execution. The tasks from EUs are randomly generated. To enhance computing efficiency, partial offloading is applied, allowing the EUs to divide their tasks into two portions. One portion is processed locally, while the other portions is simultaneously offloaded to the MEC server for processing.

The communication part of this model utilizes frequency-division multiple access (FDMA) technology. Multiple EUs concurrently offload tasks to the MEC server by utilizing dedicated sub-bands for task transmission. Furthermore, the quasi-static channel model is adopted to approximate the

TABLE I
OVERVIEW OF RECENT RESEARCHES

Author	Technology	Optimization objective	Offloading schemes	Disadvantage
Zhu *et al.* [9]	Deep reinforcement learning	Latency	Partial offloading	Ignoring energy consumption
Gu *et al.* [10]	Convex optimization	Energy consumption, security	Partial offloading	Ignoring latency costs
Zhang *et al.* [12]	Deep reinforcement learning, convex optimization	Energy consumption and processing latency	Binary offloading	Ignoring economic benefits
Saleem *et al.* [15]	MINLP, genetic algorithm	Latency and energy consumption	Binary offloading	Not considering monetary costs
Alghamdi *et al.* [16]	Genetic algorithm	Total task execution latency	Binary offloading	Weak energy model
Lin *et al.* [17]	MINLP	Fairness of offloading and energy efficiency	Binary offloading	Task cannot be partially offloaded
Wu *et al. [18]*	Lyapunov optimization and actor-critic networks	Data queue stability and energy consumption	Binary offloading	Ignoring economic benefits
Abouaomar *et al.* [20]	Lyapunov optimization	Total delay	Partial offloading	Not consider energy consumption
Le *et al.* [22]	Auction theory	Delay and energy consumption	Not considering	Only bandwidth allocation was considered
Luong *et al.* [23]	Auction theory, deep learning	Quality of service	Partial offloading	Incentive compatibility and budget balance cannot be ensured
Hossain *et al.* [26]	Non-cooperative game	Processing time, payoff	Binary allocation	Not considering the revenue of server
Liu *et al.* [28]	Stackelberg game	Revenue, latency	Partial offloading	Ignoring energy consumption
Chang *et al.* [29]	Stackelberg game	Utility of the edge servers and cloud	Partial offloading	Edge servers and cloud cannot execute tasks simultaneously.
Tong *et al.* [30]	Stackelberg game	Revenue of both MECs and EUs	Partial offloading	Equal distribution of bandwidth

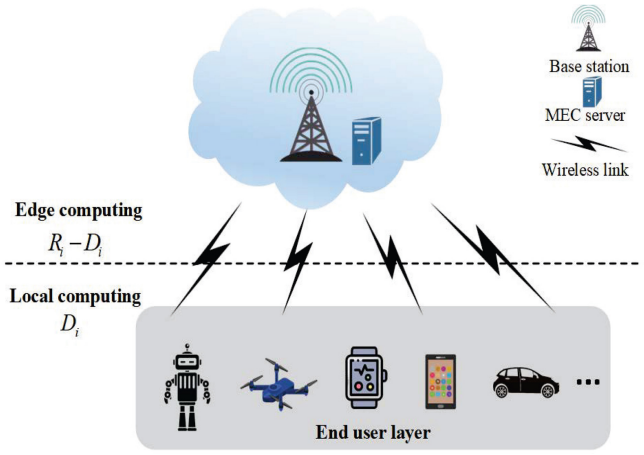


Fig. 1. Architecture of the multiuser MEC system.

TABLE II
NOTATIONS AND EXPLANATION

Symbol	Explanation
B^{edge}	Total transmission bandwidth
B_i	Transmission bandwidth of EU i
C_i	CPU cycles to compute 1-bit input data of EU i
D_i	Size of data to be offloaded of EU i
E_i^{loc}	Local energy consumption of EU i
E_i^{trans}	Energy comsumption from uplink transmission of EU i
F^{edge}	Total computing capacity of the MEC server
R_i	Total data size of EU i
T_i^{loc}	Local computing latency of EU i
T_i^{trans}	Uplink transmission latency of EU i
T_i^{exe}	Computational time from edge execution of EU i
f_i^{loc}	CPU frequency of EU i
f^{edge}	Total CPU frequency of MEC server
f_i^{edge}	CPU frequency allocated to EU i by MEC server
h_i	Channel gain between EU i and the MEC server
p_i	Transmission power of EU i to offload task
r_i	Transmission rate of EU i to offload task
ε_i	Power consumption factor of EU i
σ^2	Power spectral density of the noise
$\mathcal{N}$	The set of all EU

channel properties during an offloading period. The symbols used in this article are listed and explained in Table II.

B. Local Computing Model

To reflect the heterogeneity of the system, the amount of the input data, the computing complexity of the input data, and the local computing power of the user devices are all different. For EU i, the total data size is denoted by R_i, the CPU cycles to compute one bit of input data is denoted by C_i, and the user device CPU frequency is denoted by f_i^{loc}. Since the task of EU i can be split, D_i denotes the size of data that needs to be offloaded to the MEC server. Therefore, the size of the task processed locally by EU i is $(R_i - D_i)$. The local latency for EU i can be represented as

$$T_i^{\text{loc}} = \frac{C_i(R_i - D_i)}{f_i^{\text{loc}}}. \tag{1}$$

The local energy consumption for EU i can be expressed as

$$E_i^{\text{loc}} = \varepsilon_i \left(f_i^{\text{loc}}\right)^2 (R_i - D_i) C_i \tag{2}$$

where ε_i is the power consumption coefficient determined by the chip structure, and $\varepsilon_i(f_i^{\text{loc}})^2$ is the energy consumption the CPU running for a cycle.

C. Task Offloading and MEC Server Execution Model

The process of edge computing involves three steps: 1) uplink transmission; 2) tasks execution on the MEC server; and 3) downlink transmission. Nevertheless, due to the small data size in the downlink transmission, we opted to overlook latency and energy consumption at this stage to streamline the model [27], [31], [32].

1) Uplink Transmission: EUs offload their tasks to the MEC server through wireless channels. According to the Shannon formula, given the transmission power p_i of EU i and the channel gain h_i between EU i and MEC server, the transmission rate of EU i is

$$r_i = B_i \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right) \tag{3}$$

where B_i is the bandwidth allocated to the EU i, and σ^2 is the power spectral density of the noise.

Therefore, the uplink transmission latency for EU i can be expressed as

$$T_i^{\text{trans}} = \frac{D_i}{r_i}. \tag{4}$$

The energy consumption of uplink transmission for EU i can be expressed as

$$E_i^{\text{trans}} = p_i \frac{D_i}{r_i}. \tag{5}$$

2) MEC Server Execution: The computing speed assigned to the EU i by the MEC server is represented by f_i^{edge}. To facilitate analysis, this article assumes that the total computing speed is equally distributed, i.e., $f_i^{\text{edge}} = f^{\text{edge}}/N$, where f^{edge} denotes the total computing speed of the MEC server, namely, the total number of CPU frequency. Although this allocation strategy is simple, it is sufficient for the majority of MEC scenarios [30], [38], [39]. Therefore, the time generated during MEC server execution can be expressed as

$$T_i^{\text{exe}} = \frac{C_i D_i}{f_i^{\text{edge}}}. \tag{6}$$

Similar to existing studies, such as [33] and [34], this article focuses more on reducing energy consumption in battery-limited terminal devices. In this stage, the tasks are processed on the MEC server without consuming energy from user devices. Due to the implementation of wired access technology in the MEC server, energy scarcity is not a concern. Therefore, the energy loss during this stage is disregarded.

IV. Problem Formulation

In this section, the interaction process between EUs and the MEC server is first established as a Stackelberg game. Then, utility functions and optimization objectives of both the MEC server and EUs are derived.

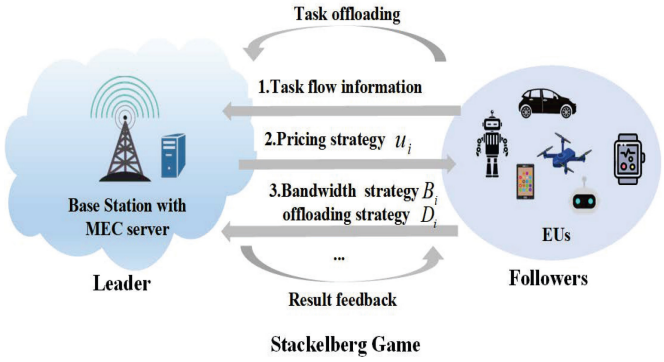


Fig. 2. Stackelberg game procedure.

A. Stackelberg Game in MEC System

In this article, we aim to optimize the utilities of both the MEC server, and the EUs. To ensure QoE for EUs, the model takes into consideration the optimization of both latency and energy consumption in the system. The EUs are given the ability to determine the size of tasks to be offloaded to the MEC server, as well as the associated bandwidth. Consequently, task assignment and bandwidth allocations are employed at the user's end to minimize the overall costs. Specifically, these costs include latency, energy consumption, and resource payment. On the MEC server side, the objective is to maximize revenue by pricing and selling resources to EUs. Furthermore, the pricing decision of the MEC server can be observed by EUs and used as an input for their bandwidth purchasing decision. The decision of EUs is constrained by the decision of the MEC server. This participant situation exhibits asymmetry, where one party takes a leading action and assumes that the other party will make a rational choice based on its own decision, which is consistent with the Stackelberg game. Therefore, the interaction process between the MEC server and EUs is taken as a single-leader–multi-follower Stackelberg game [35], [36]. EUs act as followers, while the MEC server assumes the role of the leader. The Stackelberg game procedure in this MEC system is depicted in Fig. 2.

B. Utility Function and Optimization Problem of MEC Server

Due to the maintenance and deployment costs of the MEC server, it gains revenue by selling its bandwidth resources to EUs. The utility of the MEC server can be described as

$$U^{\text{edge}} = \sum_{i=1}^{N} u_i B_i \tag{7}$$

where the unit bandwidth price for EU i is defined as u_i, and the bandwidth resources purchased by EU i are defined as B_i. The MEC server aims to maximize its revenue by selling bandwidth resources at a reasonable price. Given the practical constraints and the objective of maximizing utility, the optimization problem of the MEC server can be described as

$$\text{P1:} \ \arg\max U^{\text{edge}}(u_i)$$

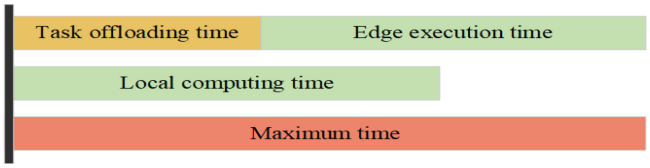


Fig. 3. Latency structure.

$$\text{s.t.} \sum_{i=1}^{N} B_i \le B^{\text{edge}} \tag{8}$$

where the constraint denotes that the total amount of bandwidth purchased by EUs cannot exceed the total system bandwidth.

C. Utility Function and Optimization Problem of EUs

For EUs, their costs consist of three parts: 1) latency; 2) energy consumption; and 3) payment to the MEC server [21], [30], [39]. Since both local computing and MEC server computing can be carried out concurrently in different terminals, the latency structure in this article is illustrated in Fig. 3.

The latency overhead for EU i can be described as

$$T_i = \max\left\{T_i^{\text{loc}}, T_i^{\text{trans}} + T_i^{\text{exe}}\right\}. \tag{9}$$

Therefore, the overall cost incurred by EU i can be expressed as

$$U_i^{\text{user}} = w_i^t T_i + w_i^e\left(E_i^{\text{loc}} + E_i^{\text{trans}}\right) + w_i^p u_i B_i \tag{10}$$

where $0 \le w_i^t, w_i^e w_i^p \le 1$, respectively, denote the weighting factors of latency, energy consumption, and payment [39], [40]. It should be noted that overall cost is defined as a linear combination of these three indicators because they can concurrently reflect the cost of task offloading. That is, longer latency, higher energy consumption, and higher resource procurement costs lead to increased EUs costs. Parameters can be selected based on EUs preferences. For example, when an EU has a low battery level, energy consumption becomes more crucial. To enhance its importance in overall optimization, we can dynamically adjust the energy consumption parameter.

Equation (10) can be equivalently expressed in a more specific form as

$$U_i^{\text{user}} = \begin{cases} w_i^e\left(\varepsilon_i\left(f_i^{\text{loc}}\right)^2(R_i - D_i)C_i + p_i\frac{D_i}{r_i}\right) + \\ w_i^t\left(\frac{(R_i - D_i)C_i}{f_i^{\text{loc}}}\right) + w_i^p u_i B_i \ , 0 \le D_i \le x_i \\ w_i^e\left(\varepsilon_i\left(f_i^{\text{loc}}\right)^2(R_i - D_i)C_i + p_i\frac{D_i}{r_i}\right) + \\ w_i^t\left(\frac{D_i}{r_i} + \frac{D_i C_i}{f_i^{\text{edge}}}\right) + w_i^p u_i B_i \ , x_i < D_i \le R_i \end{cases} \tag{11}$$

where x_i is equal to

$$x_i = \frac{R_i}{\frac{f_i^{\text{loc}}}{r_i C_i} + \frac{f_i^{\text{loc}}}{f_i^{\text{edge}}} + 1}. \tag{12}$$

Based on the pricing strategy provided by the MEC server, EUs tend to minimize their overall costs through their own bandwidth purchasing and task offloading strategies. It is required to ensure that the total purchased bandwidth by EUs does not exceed the system's bandwidth capacity. Meanwhile, there is another practical constraint. Within one offloading period, the MEC server has an upper limit on the available CPU cycles for computing tasks, and the total amount of data offloaded from all EUs to the MEC server must not exceed this threshold [30]. It is worth noting that F^{edge} and f^{edge} represent the computing capacity and computing speed of the MEC server's CPU, respectively. Based on the optimal price of MEC server u_i^* [41], the optimization problem of EU i can be formulated as

$$\begin{aligned} P2\text{: arg min} \ & U_i^{\text{user}}\left(B_i, D_i, u_i^*\right) \\ s.t. \ & \sum_{i=1}^{N} B_i \le B^{\text{edge}} \\ & \sum_{i=1}^{N} C_i D_i \le F^{\text{edge}}. \end{aligned} \tag{13}$$

It is evident that the optimization problems of P1 and P2 are coupled in a complex manner. The pricing strategy of the MEC server will have an impact on the amount of resources purchased by EUs, while the bandwidth purchased by EUs will, in turn, affect the MEC server's pricing strategy.

V. Algorithm Design and Analysis

In this section, the optimization problem is first simplified to a univariate convex optimization problem. Second, the existence of Nash equilibrium among EUs is proven. Subsequently, the optimization problem of the MEC server is solved. Finally, resource allocation is performed using the DPRA algorithm.

A. Problem Simplification

Reverse induction is a commonly used problem-solving method in game theory. It can effectively solve problems involving interdependent decision making between EUs and the MEC server. The method adopts a two-stage game solution. In the first stage, problem $P2$ is solved, where EUs obtain their optimal bandwidth purchase strategy B^* and offloading strategy D^* by solving an optimization problem with given price strategy u. In the second stage, problem $P1$ is solved, where the MEC server obtains the optimal pricing strategy u^* by solving another optimization problem based on EUs' optimal bandwidth strategy B^* and offloading strategy D^*.

After the MEC server presents the bandwidth price for each EU, the optimization problem $P2$ of EUs only involves the purchased bandwidth and offloading task size. The comprehensive expenses of EU i can be simplified as

$$U_i^{\text{user}} = \begin{cases} k_{i1} D_i + b_{i1}, & 0 \le D_i \le x_i \\ k_{i2} D_i + b_{i2}, & x_i < D_i \le R_i \end{cases} \tag{14}$$

where k_{i1}, k_{i2}, b_{i1}, and b_{i2} are expressed as

$$k_{i1} = w_i^e p_i / r_i \ - w_i^t C_i / f_i^{\text{loc}} \ - w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i$$

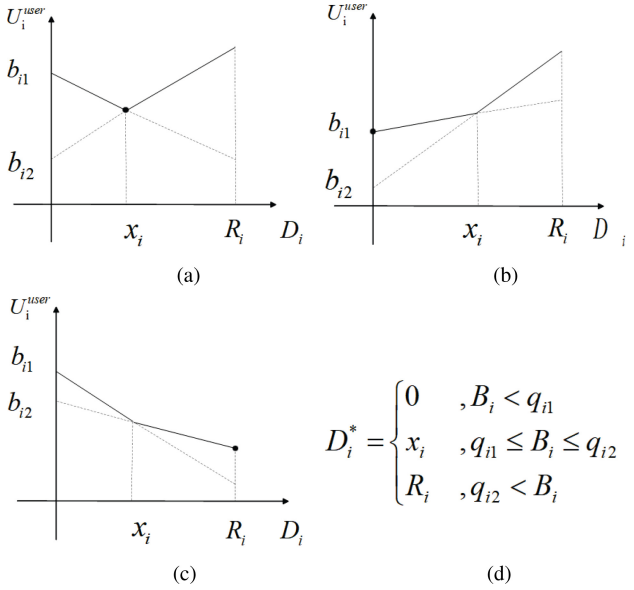


Fig. 4. U_i^{user} with respect to slope. (a) $k_{i1} \le 0 \le k_{i2}$, (b) $0 < k_{i1} < k_{i2}$, (c) $k_{i1} < k_{i2} < 0$, and (d) Relationship between bandwidth and offloading size.

$$\begin{aligned} k_{i2} &= w_i^e p_i / r_i + w_i^t C_i / f_i^{\text{edge}} + w_i^t / r_i - w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i \\ b_{i1} &= w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 R_i C_i + w_i^p u_i B_i + w_i^t C_i R_i / f_i^{\text{loc}} \\ b_{i2} &= w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 R_i C_i + w_i^p u_i B_i. \end{aligned} \tag{15}$$

It is evident that k_{i1}, k_{i2}, b_{i1}, and b_{i2} satisfy $k_{i1} < k_{i2}$ and $0 < b_{i2} < b_{i1}$. According to the relationship between the slope and 0, the relationship between bandwidth and offloading data size is discussed under different circumstances, as shown in Fig. 4.

As shown in Fig. 4(a), when $k_{i1} \le 0 \le k_{i2}$, the bandwidth of EU i satisfies $q_{i1} \le B_i \le q_{i2}$, where

$$q_{i1} = \frac{w_i^e p_i}{\left(\frac{w_i^t C_i}{f_i^{\text{loc}}} + w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i\right) \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right)} \tag{16}$$

$$q_{i2} = \frac{w_i^t + w_i^e p_i}{\left(w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i - \frac{w_i^t C_i}{f_i^{\text{edge}}}\right) \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right)}. \tag{17}$$

q_{i1} and q_{i2} denote the lower and upper bounds of bandwidth, respectively.

As shown in Fig. 4(a), EU i achieves the minimum overall cost at task offloading size x_i when $k_{i1} \le 0 \le k_{i2}$. At this point, the task is not only executed locally but also partially offloaded to the MEC server.

In Fig. 4(b), when $0 < k_{i1} < k_{i2}$, the overall cost of EU i achieves its minimum at offloading size 0, indicating that task is executed only locally.

In Fig. 4(c), when $k_{i1} < k_{i2} < 0$, the overall cost of EU i achieves its minimum at task offloading size R_i, indicating the whole task is computed on the MEC server.

To sum up, the optimal offloading strategy D_i^* for EU i can be illustrated in Fig. 4(d). However, when EUs purchase more bandwidth resources and conduct complete offloading, it may lead to a few EUs monopolizing server computing resources. In order to achieve resource balance and fairness, when the situation shown in Fig. 4(c) occurs, the MEC server incentivizes EUs to choose the amount of task offloading x_i. Additionally, this method can effectively utilize local computing resources, reduce server load, and facilitate the rational use of resources.

Through the above analysis, when EU i is about to perform edge computing, the task offloading size is denoted as x_i. In this case, the overall cost of the two scenarios in (11) is equal. Therefore, the utility function of EU i can be transformed as

$$\begin{aligned} U_i^{\text{user}}(B_i) = &\left(\frac{w_i^e p_i}{r_i} - w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i - \frac{w_i^t C_i}{f_i^{\text{loc}}}\right) x_i \\ &+ w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^2 C_i + w_i^p u_i B_i + \frac{w_i^t R_i C_i}{f_i^{\text{loc}}}. \end{aligned} \tag{18}$$

B. Optimization of EUs

Definition 1: There exists Nash equilibrium among EUs with $B^* = \{B_1^*, B_2^*, \ldots, B_N^*\}$. At this point, there is a utility function $U_i^{\text{user}}(B_i^*, B_{-i}^*) > U_i^{\text{user}}(B_i, B_{-i}^*)$, where B_{-i}^* is the best strategy for other EUs excluding EU i.

Nash equilibrium possesses a desirable property of self-stability in academia, enabling users in an equilibrium state to attain mutually satisfactory solutions without any incentive to deviate. This characteristic is crucial for price optimization problems since individuals, as distinct entities, may act based on their own interests.

Theorem 1: In the game, with the MEC server serving as the leader and the EUs functioning as followers, the optimal strategy for the bandwidth of EU i can be represented as

$$B_i^* = \frac{\sqrt{\alpha_i R_i \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right) / w_i^p u_i} - f_i^{\text{loc}} / C_i}{\beta_i} \tag{19}$$

where

$$\alpha_i = w_i^e p_i \left(1 + f_i^{\text{loc}} / f_i^{\text{edge}}\right) + w_i^e \varepsilon_i \left(f_i^{\text{loc}}\right)^3 + w_i^t \tag{20}$$

$$\beta_i = \left(1 + f_i^{\text{loc}} / f_i^{\text{edge}}\right) \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right). \tag{21}$$

Proof: For EU i, the first-order derivative of the utility function U_i^{user} is equal to

$$\frac{\partial U_i^{\text{user}}}{\partial B_i} = \frac{-\alpha_i R_i \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right)}{\left(f_i^{\text{loc}} / C_i + \beta_i B_i\right)^2} + w_i^p u_i. \tag{22}$$

The second-order derivative of U_i^{user} is equal to

$$\frac{\partial^2 U_i^{\text{user}}}{\partial^2 B_i} = \frac{2\alpha_i \lambda_i R_i \log_2\left(1 + \frac{p_i h_i}{\sigma^2}\right)}{\left(f_i^{\text{loc}} / C_i + \beta_i B_i\right)^3}. \tag{23}$$

As both the numerator and denominator are positive, the second derivative of the utility function is positive. Therefore, the utility function of EU i is strictly convex, indicating the existence of a Nash equilibrium among EUs. According to (22) and (23), the uniqueness of strategy $B = \{B_1^*, B_2^*, \ldots, B_n^*\}$ can

be proven. When $\partial U_{\mathrm{i}}^{\mathrm{user}}/\partial B_i = 0$, we obtain the optimal bandwidth for EUs from (22). ■

Lemma 1: For a given pricing strategy from the MEC server, each EU has a unique optimal bandwidth strategy.

Proof: The first and second partial derivatives of the optimal bandwidth B_i^* with respect to u_i can be obtained as follows:

$$\frac{\partial B_i^*}{\partial u_i} = \frac{-\sqrt{\alpha R_i \log_2\left(1+\frac{p_i h_i}{\sigma^2}\right)}}{2\beta_i \sqrt{w_i^p} u^{\frac{3}{2}}} \tag{24}$$

$$\frac{\partial^2 B_i^*}{\partial^2 u_i} = \frac{-3\sqrt{\alpha_i R_i \log_2\left(1+\frac{p_i h_i}{\sigma^2}\right)}}{4\beta_i \sqrt{w_i^p} u^{\frac{5}{2}}}. \tag{25}$$

The negative values of the first and second partial derivatives indicate that as the MEC server sets a lower price, EUs tend to purchase more bandwidth. Furthermore, the optimal bandwidth B_i^* is a monotonically decreasing upper convex function of the pricing u_i, resulting in a unique and optimal strategy B_i^*. ■

C. Optimization of the MEC Server

Definition 2: If $U^{\mathrm{edge}}(u_i^*, B_i^*) > U^{\mathrm{edge}}(u_i, B_i^*)$, then a unique Stackelberg equilibrium exists between EUs and the MEC server.

Theorem 2: In the game, the MEC server, as the leader, has an optimal pricing strategy for each EU, which can be represented as

$$u_i^* = \frac{\alpha_i R_i C_i^2 \log_2\left(1+\frac{p_i h_i}{\sigma^2}\right)}{4 w_i^p \left(f_i^{\mathrm{loc}}\right)^2}. \tag{26}$$

Proof: By substituting the optimal bandwidth strategy B_i^* and the corresponding data size x_i into the utility function U^{edge}, we can obtain $U^{\mathrm{edge}}(u_i, B_i^*)$. The MEC server adjusts pricing strategy to control the bandwidth purchased by EUs. For the MEC server, the first-order derivative of the utility function $U^{\mathrm{edge}}(u_i, B_i^*)$ to u_i is

$$\frac{\partial U^{\mathrm{edge}}}{\partial u_i} = \frac{\sqrt{\alpha_i R_i \log_2\left(1+\frac{p_i h_i}{\sigma^2}\right)/4 w_i^p u_i} - f_i^{\mathrm{loc}}/C_i}{\beta_i}. \tag{27}$$

The second-order derivative of the utility function U^{edge} to u_i is

$$\frac{\partial^2 U^{\mathrm{edge}}}{\partial^2 u_i} = -\frac{\sqrt{\alpha_i R_i \log_2\left(1+\frac{p_i h_i}{\sigma^2}\right)}}{4\beta_i \sqrt{w_i^p} u^{\frac{3}{2}}}. \tag{28}$$

According to the convex optimization theorem, the second derivative of the utility function for the MEC server is negative, indicating strict concavity of U^{edge}. Consequently, it can be concluded that $P1$ is a convex optimization problem, and the optimal pricing strategy u_i^* is unique at the same time. Therefore, Theorem 2 is validated. ■

D. DPRA Algorithm

Based on the aforementioned analysis, we have determined the optimal bandwidth pricing for EUs by the server in P1. Additionally, in P2, the bandwidth purchase strategy for each EU is B_i^* and the offloading size is x_i. It is important to note that, due to limited resources of the MEC server, the sum of the optimal resource allocation for all EUs may exceed the system limit. Therefore, we need to determine which EUs to allocate resources to. An indicator function $\theta_i \in \{0, 1\}$ is needed to determine which EUs can offload their tasks. This leads to the formulation of optimization problem P3, where we transform the objectives of solving resource allocation and pricing into the decision of which EUs should receive resource allocation

$$\begin{aligned} \text{P3:} \quad & \max_{\theta_i \in \{0,1\}} U^{\mathrm{edge}} = \sum_{\mathrm{i}=1}^{N} \theta_i u_i^* B_i^* \\ \text{s.t.} \quad & \sum_{\mathrm{i}=1}^{N} \theta_i B_i^* \le B^{\mathrm{edge}}. \end{aligned} \tag{29}$$

The knapsack problem is NP-hard when expressed as decision problems. Its possible solutions can be efficiently verified in polynomial time, and it can be reduced to other NP-complete problems using polynomially transformations. For the optimization problem P3, the MEC server utility is further increased when the EUs offload their tasks. Based on this, the 0-1 knapsack problem can be translated into a special case of maximizing MEC's utility. Specifically, the set of items to be placed in the knapsack in the 0-1 knapsack problem is regarded as the EUs that offload their tasks on the MEC in the problem. The weight of each item $w[i]$ is assigned as the bandwidth resources allocated to EU i in the problem. Due to the limited communication capacity, the total bandwidth of the system has an upper limit, which can be regarded as the maximum capacity of the knapsack in the 0-1 knapsack problem. Therefore, the 0-1 knapsack problem can be simplified as a special case of P3. So, if there exists an algorithm to solve P3, it can also be used to solve the 0-1 knapsack problem, which means P3 is an NP-hard problem.

As for the value of the items $v[i]$, there are two possibilities: when the optimal bandwidth B_i^* is less than the lower bounds of bandwidth q_{i1}, the cost of local computing by EU i will be smaller than partial offloading. Therefore, the value $v[i]$ obtained by offloading task for EU i is updated as 0 at this time, and subsequent this EU will not be able to offload task. Otherwise, the value of the item $v[i]$ will be the revenue $u_i^* B_i^*$ obtained from MEC server performing the task of EU i. The complete pseudocode process is presented in Algorithm 1.

The optimal value solution (OVS) algorithm consists of a loop (lines 5–12) that is used to determine the value of items for offloading. Consequently, the time complexity of the OVS algorithm is $O(N)$.

Since the knapsack problem is NP problem, finding the optimal solution efficiently is not feasible [37]. To tackle this problem, a method based on dynamic programming is adopted to obtain the optimal solution of this problem.

$\varphi[i][k]$ represents the maximum revenue that can be obtained by the MEC server selling k units of bandwidth to the

Algorithm 1 OVS

Input: B^{edge}, C_i, F^{edge}, R_i, f_i^{loc}, f^{edge}, h_i, p_i, σ^2
Output: B_i^*, x_i, u_i^*, $V = \{v_1, v_2, ..., v_N\}$
1: Initialize w_i^t, w_i^e, w_i^p;
2: The MEC server decides the optimal unit price u_i^* of the bandwidth for each EU according to Eq. (26);
3: Each EU decides the optimal bandwidth strategies B_i^* according to Eq. (19);
4: Each EU decides the offloading strategies x_i according to Eq. (12);
5: **for** $i = 1$ to N **do**
6: EU i computes the lower bounds of bandwidth q_{i1};
7: **if** $B_i^* < q_{i1}$ **then**
8: $v[i] = 0$;
9: **else**
10: $v[i] = u_i^* B_i^*$;
11: **end if**
12: **end for**
13: return V

Algorithm 2 DPRA

Input: B_i^*, x_i, B^{edge}, F^{edge}, $V = \{v_1, v_2, ..., v_N\}$
Output: θ_i
1: **for** $i = 0$ to N **do**
2: **for** $k = 0$ to B^{edge} **do**
3: $\varphi[i][k] = \varphi[i-1][k]$;
4: **if** $k < B^{\text{edge}}$ **then**
5: $\varphi[i][k] = \varphi[i-1][k]$;
6: **else if** $\varphi[i-1][k - B_i^*] + v[i] > \varphi[i-1][k]$ and $\sum_{s=1}^{i} \theta_i C_s x_s + C_i x_i \le F^{\text{edge}}$ **then**
7: $\varphi[i][k] = \varphi[i-1][k - B_i^*] + v[i]$;
8: **else**
9: $\varphi[i][k] = \varphi[i-1][k]$;
10: **end if**
11: **end for**
12: **end for**
13: **for** $i = N$ to 1 **do**
14: **if** $\varphi[i][k] = \varphi[i-1][k]$ **then**
15: $\theta_i = 0$;
16: **else**
17: $\theta_i = 1$;
18: **end if**
19: **end for**

first i EUs. As shown in Algorithm 2, there are three possible values for $\varphi[i][k]$.

1) If the MEC server's bandwidth and computing resources are insufficient to complete the offloaded task of EU i, the MEC server will not process the task of EU i. At this time, the revenue of the MEC server is the same as the value of the offloading strategy of the previous $(i-1)$ EUs, which means $\varphi[i][k] = \varphi[i-1][k]$.
2) If the bandwidth and computing resources of the MEC server can satisfy the requirements of EU i's task, it is still necessary to determine whether to offload the task. This is because allowing task offloading does not necessarily maximize the revenue of the MEC server. The maximum benefit that the server can achieve when the ith EU does not offload the task is $\varphi[i-1][k]$, and the maximum revenue of the MEC server when the ith EU offload the task is $\varphi[i-1][k-B_i^*]+v[i]$. $\varphi[i][k]$ will be equal to the larger of the two values.
3) If the maximum benefit of the MEC server offloading task for EU i is equal to the maximum benefit without offloading the task, then it indicates that the value of offloading the task for EU i is $v[i] = 0$. The MEC server will not process the tasks for EU i, which means $\varphi[i][k] = \varphi[i-1][k]$.

For the DPRA algorithm, it contains a two levels of loops (lines 1–12). The first-level loop iterates N times, while the second-level loop iterates B^{edge} times. Consequently, the time complexity of this part is $O(N \times B^{\text{edge}})$. After these steps, backtracking is used to determine which EUs perform task offloading (lines 12–19). Since it is necessary to traverse the state space, the time complexity of this part is $O(N)$. In summary, the time complexity of the DPRA algorithm is $O(N \times B^{\text{edge}})$.

VI. Numerical Results

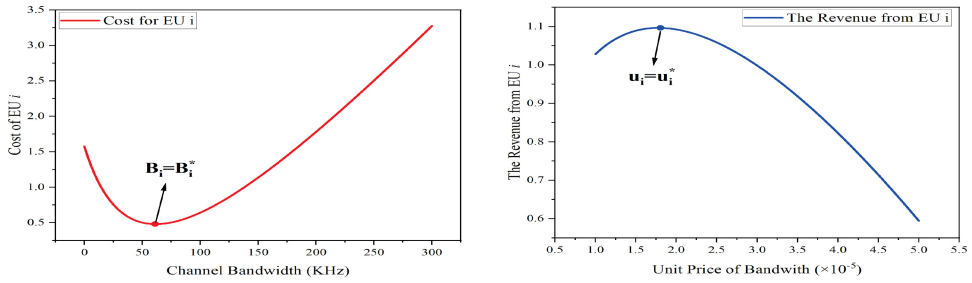
In this section, simulation experiments are conducted to assess the efficacy of the proposed DPRA algorithm. First, experimental validation is conducted on the previously mathematically proven Stackelberg game equilibrium. Second, the influence of some parameters on the utilities of EUs and the MEC server is evaluated. Finally, we compare the proposed algorithm with other strategies.

A. Simulation Setting

The proposed algorithm is simulated using Python programming language. The parameter values used in the experiments are primarily derived from [28]. It is assumed that there are 30 EUs in the experiment, each with its own local computing capability and generating different computing tasks. The power spectral density of the noise is -174 dBm/Hz. The channel gain from MEC server to EUs is uniformly distributed by the set $[-50, -30]$ dBm. This article considers three weighting factors, namely, weighting factors for latency, energy consumption, and payment to the MEC server. These factors can be adjusted according to the actual requirements of the device. In the energy-saving scenario, $w_i^e > w_i^t$, whereas in the latency-sensitive scenario, $w_i^e < w_i^t$. In this article, we assume that all three weighting factors are equal to $1/3$. In addition, Table III summarizes other key parameters in the experiments.

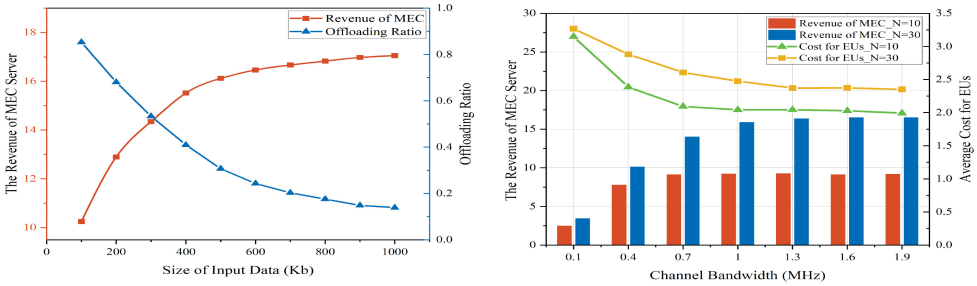
B. Stackelberg Equilibrium

In Stackelberg equilibrium, the MEC server assumes that it already knows what strategies the followers will adopt to maximize their own interests and thus formulate the pricing strategy. Then after learning of the strategy of the MEC server, the EUs will choose their own optimal strategies. After carefully considering their own interests in light of the opponent's response, both the MEC server and the EUs can devise the optimal strategy to achieve their respective



(a) (b)

Fig. 5. Stackelberg equilibrium. (a) Optimal bandwidth of EU i. (b) Optimal price of the server.



(a) (b)

Fig. 6. Influence of parameters. (a) Influence of data size. (b) Influence of bandwidth.

TABLE III
SIMULATION PARAMETERS SETTING

Parameter	Value
B^{edge}	1.2 MHz
C_i	500 - 800 cycles/slot uniformly
F^{edge}	8×10^9
R_i	100 - 300Kb uniformly
f_i^{loc}	0.1 - 0.5 GHz uniformly
f^{edge}	100 GHz
h_i	10^{-6}
p_i	0.1W
ε_i	10^{-25}

objectives. The optimal bandwidth for EU i is B_i^*, and the optimal pricing for the MEC server to EU i is u_i^*, shown in Fig. 5.

Fig. 5(a) is the utility of EU i with a fixed MEC server price $u_i = u_i^*$. The bandwidth of EU i is represented on the horizontal axis, and the minimum overall cost is achieved when the bandwidth is $B_i = B_i^*$. Similarly, Fig. 5(b) is the revenue obtained by the MEC server from EU i with fixed optimal bandwidth $B_i = B_i^*$. When the revenue of the MEC server is maximized, the optimal pricing $u_i = u_i^*$. The above indicates that both EUs and the MEC server cannot change their strategies to optimize their utilities in the Stackelberg equilibrium.

C. Impact of Parameters

As shown in Fig. 6, the model parameters have a certain impact on the experiment.

Fig. 6(a) illustrates the impact of task size on the MEC revenue and the offloading ratio of EUs. As depicted in the figure, the MEC server's revenue rises as the task size increases and then tends to converge. This is attributed to the fact that the larger the task size, the more bandwidth resources are needed by EUs, and the server sells more resources. However, as the bandwidth and computing resources of the MEC server reach the capacity limits, the sale of resources becomes restricted, causing the revenue to converge. Furthermore, as the task size of each EU expands, the limited resources of the MEC server result in a reduction in the number of EUs capable of obtaining offloading resources. Consequently, the proportion of tasks that can be offloaded diminishes gradually.

Fig. 6(b) shows the impact of total bandwidth capacity on the MEC server revenue and the average cost of EUs. As the total bandwidth increases, the number of EUs that can access the bandwidth resources also increases, and the MEC server sells more resources accordingly. This results in an increase in MEC server revenue and a decrease in average cost for EUs. However, as the bandwidth capacity of the MEC server reaches a sufficiently large level, both the revenue and the average cost for EUs tend to stabilize. On the one hand, this is due to upper limit on the number of EUs, the bandwidth resources are already abundant. On the other hand, although bandwidth resources are increasing, the total computing capacity remains unchanged. With an increasing number of EUs, the allocation of computing resources to EUs becomes constrained, resulting in higher average cost when there are 30 EUs compared to 10 EUs. Based on Fig. 6, both the MEC server and EUs will benefit from the trade. When EUs offload more tasks to the MEC server, EUs' average cost decreases, and the MEC server's profits increase. That is to say the proposed DPRA algorithm promotes collaboration between the MEC server and EUs with a mutually beneficial outcome, demonstrating the efficacy of the algorithm.

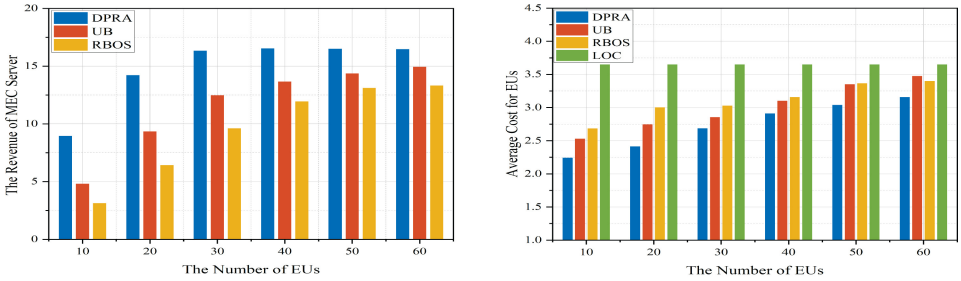


Fig. 7. Comparison with numbers of EUs. (a) Revenue of MEC server. (b) Average cost for EUs.

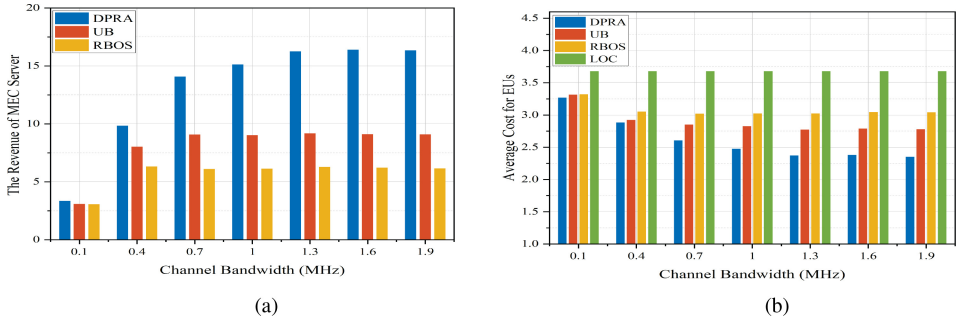


Fig. 8. Comparison with channel bandwidth. (a) Revenue of MEC server. (b) Average cost for EUs.

D. Performance Comparison

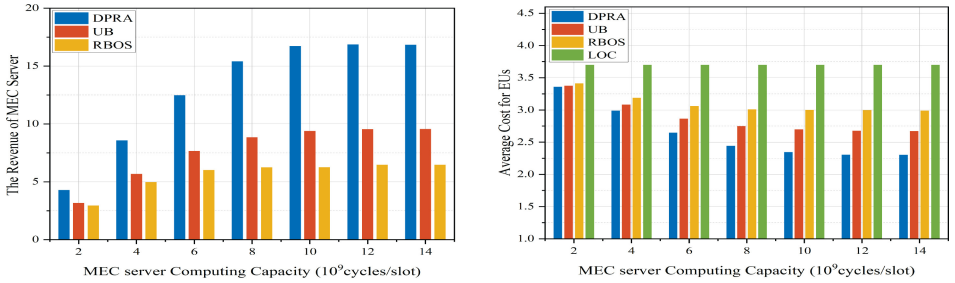
Due to differences in experimental settings, utility functions, etc. compared to other papers, it is difficult to directly compare the proposed DPRA algorithm with the latest algorithms. In order to assess the effectiveness of the our algorithm, we conducted experiments to compare the DPRA algorithm with the following bandwidth strategies across three dimensions: 1) the number of EUs; 2) total bandwidth capacity; and 3) total computing capacity. This comparative experiment aims to evaluate the utility of the MEC server and the average cost for EUs.

1) *Random Bandwidth With Optimal Size (RBOS):* In the RBOS strategy, each EU selects the optimal offloading data size, but the bandwidth is randomly generated within a given range.
2) *Uniform Bandwidth (UB):* In the UB strategy, similar to [30], the bandwidth for each EU is uniformly allocated and set to the middle value of the set $P = \{(p_{11}, p_{12}), \ldots, (p_{n1}, p_{n2})\}$ to maintain stable profit.
3) *Local Offloading Computing (LOC):* In the LOC strategy, the whole task of EU i is executed locally without task offloading.

The experiments for different numbers of EUs are shown in Fig. 7. First, several strategies exhibit similar trends. As depicted in Fig. 7(a), the sold bandwidth resources exhibit an upward trend as the number of EUs in the system rises, leading to a corresponding growth in the revenue of the MEC server. However, due to the overall resource constraints of the server, the final revenue tends to stabilize. As shown in Fig. 7(b), the average cost for EUs increases with an expanding number of EUs. This is because as the number of EUs grows, the competition for bandwidth and computing resources becomes more intense, leading to a decrease in the proportion of EUs capable of performing task offloading. This, in turn, results in increased latency and energy consumption costs.

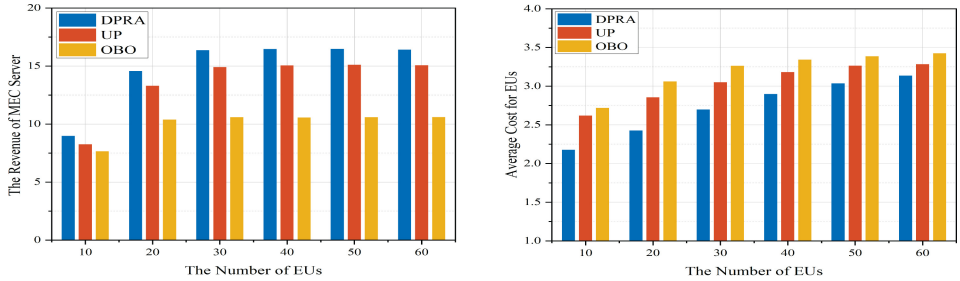
The revenue of the MEC server and the average cost of EUs under different strategies with varying total bandwidth resources is shown in Fig. 8. Compared to other strategies, the DPRA strategy significantly improves the profitability of the MEC server, as experimental results demonstrate that its revenue is twice that of the RBOS and UB algorithms. Additionally, the DPRA strategy maintains the optimal average cost for EUs, affirming the rationality of DPRA's bandwidth allocation. In contrast, the RBOS and UB algorithms fail to adequately consider the individual demands of each EU and perform differentiated allocation to maximize server revenue, thus falling short of the effectiveness achieved by the DPRA algorithm.

Fig. 9 displays the changes in MEC server revenue and average cost for EUs under different computing capabilities. Similar to Fig. 8, DPRA outperforms other strategies in terms of MEC server profitability and average cost for EUs. The proposed DPRA algorithm demonstrates optimal performance in all the experiments conducted. In the RBOS strategy, the allocation of computing resources is based on the optimal relationship between task offloading size and purchased bandwidth. However, due to the unreasonable allocation of bandwidth resources, it also leads to inappropriate allocation of computing resources, resulting in high overall costs for users. In the UB strategy, although the task offloading size is consistent with our proposed algorithm, unnecessary delays and energy consumption costs are incurred during transmission due to the inability of bandwidth to meet the specific requirements of each EU. This also highlights the



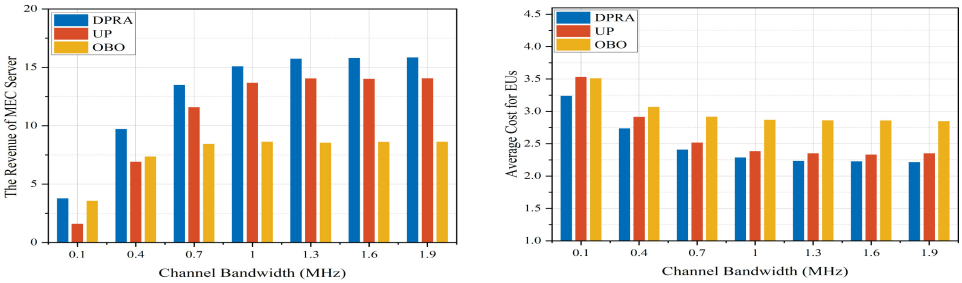
(a) (b)

Fig. 9. Comparison with computing capacity. (a) Revenue of MEC server. (b) Average cost for EUs.



(a) (b)

Fig. 10. Comparison with numbers of EUs. (a) Revenue of MEC server. (b) Average cost for EUs.



(a) (b)

Fig. 11. Comparison with channel bandwidth. (a) Revenue of MEC server. (b) Average cost for EUs.

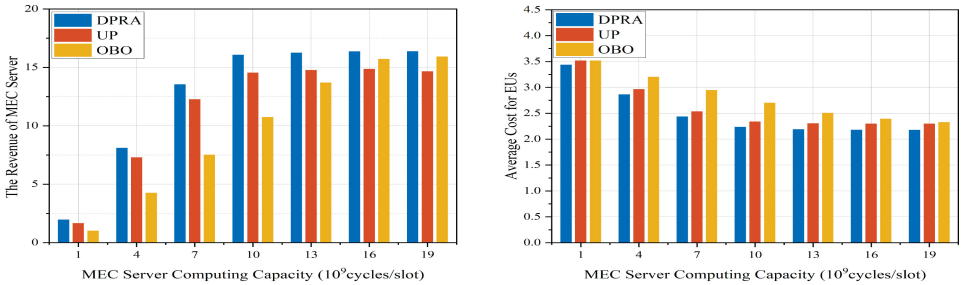
importance of reasonable allocation of bandwidth resources. Both of these strategies result in low resource utilization, impacting system performance and efficiency, due to inappropriate bandwidth allocation strategies. Finally, as a classic benchmark test, the LOC strategy performs the worst in all the experiments mentioned above. Therefore, this demonstrates that task offloading in the MEC architecture can effectively improve the QoE for EUs.

To conduct a more comprehensive evaluation of the proposed DPRA algorithm, it is compared with various pricing and offloading strategies.

1) *Uniform Pricing (UP):* In the UP strategy, the MEC server uniformly price the bandwidth resources sold to EUs based on the principle of fairness.
2) *Optimal Bandwidth Offloading (OBO):* In the OBO strategy, EUs purchase an optimal amount of bandwidth, but the amount of offloading data size is random.

As the number of EUs increases, the variations of MEC server revenue and average costs for EUs under different strategies are depicted in Fig. 10. First, the growth trends resemble those observed in Fig. 7. Additionally, in Fig. 10(b), the rate of cost reduction for EUs is slower compared to the MEC server's profit in Fig. 10(a). This is attributed to the noncooperative competition among EUs within the system. With an increase in the number of participating EUs in task offloading, the MEC server can offload more tasks to boost its profitability. Second, among all the strategies, DPRA exhibits the best performance. This is because the UP strategy does not adjust bandwidth pricing differently based on EUs' varying levels of price acceptance. Consequently, some EUs may incur excessive costs or fail to obtain the desired QoE, thus not fully leveraging the advantages of the MEC server. In the OBO strategy, EUs are unable to fully utilize the MEC server's computing resources to minimize latency and energy consumption costs.

Due to the heterogeneity of the MEC system, the upper limit of MEC system resources will not be the same value. The impact of bandwidth resources and computing capacity on the revenue of the MEC server and the average cost of EUs under different strategies is shown in Figs. 11 and 12. Both bandwidth and computing resources impose constraints



(a) (b)

Fig. 12. Comparison with computing capacity. (a) Revenue of MEC server. (b) Average cost for EUs.

on participants in the MEC architecture. Among the three strategies, as resources increase, the MEC server revenue shows an growth and then tends to stabilize, while the average cost for EUs gradually decreases and eventually becomes stable. As shown in Fig. 11, although the OBO strategy adopts optimal bandwidth allocation, the impracticality of computing resource allocation causes the assigned computing resources to quickly reach their limit, thereby restricting the number of EUs able to perform task offloading. This results in MEC profits being only half of what our proposed algorithm achieves, even when bandwidth resources are abundant. Similarly, as illustrated in Fig. 12, when computing resources are insufficient (e.g., 1×10^9 cycles/slot), even with adequate bandwidth resources, bandwidth sales will still be restricted due to failure to meet EUs' minimum requirements of computing resources. As computing resources become sufficient, the algorithm gradually improves its performance, but it still tends to reach a balance due to the constraints of the bandwidth resource upper limit.

VII. Conclusion and Future Work

This article utilizes the Stackelberg game to depict the interaction process between the MEC server and EUs. It addresses the problem of revenue balance faced by the MEC server in the presence of limited computing and communication resources, while ensuring the QoE for EUs. This article first establishes the optimal relationship between bandwidth and task offloading size to simplify decision making for EUs. Subsequently, the optimal strategies for the MEC server and EUs are derived through backward induction, and a collaborative offloading strategy is devised using the DPRA algorithm. Simulation experiments are conducted to evaluate the proposed DPRA algorithm's performance. The results demonstrate that the DPRA algorithm outperforms other comparative strategies, illustrating its effectiveness in optimizing resource utilization and achieving lower latency and energy consumption levels. This has practical significance in promoting the development of MEC technology and enhancing the QoE for EUs.

Future research can further explore more complex game model, such as multiuser and multiserver game. Additionally, establishing cooperative relationships among MEC servers can facilitate offloading for EUs, meeting their increasing computational demands.

References

[1] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the Internet of Things (IoT) forensics: challenges, approaches, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1191–1221, 2nd Quart., 2020.

[2] F. Al-Doghman, N. Moustafa, I. Khalil, N. Sohrabi, Z. Tari, and A. Y. Zomaya, "AI-enabled secure microservices in edge computing: Opportunities and challenges," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1485–1504, Mar./Apr. 2023.

[3] C. Tselios and G. Tsolis, "On QoE-awareness through virtualized probes in 5G networks," in *Proc. IEEE 21st Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, 2016, pp. 159–164.

[4] A. Sunyaev, "Cloud computing" in *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*. Cham, Switzerland: Springer, 2020, pp. 195–236.

[5] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023.

[6] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, 2019.

[7] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[8] X. Huang, B. Zhang, and C. Li, "Incentive mechanisms for mobile edge computing: Present and future directions," *IEEE Netw.*, vol. 36, no. 6, pp. 199–205, Nov./Dec. 2022.

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

[10] X. Gu, G. Zhang, M. Wang, W. Duan, M. Wen, and P.-H. Ho, "UAV-aided energy-efficient edge computing networks: Security offloading optimization," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4245–4258, Mar. 2022.

[11] Z. Tong, J. Cai, J. Mei, K. Li, and K. Li, "Dynamic energy-saving offloading strategy guided by Lyapunov optimization for IoT devices," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19903–19915, Oct. 2022.

[12] X. Zhang, X. Zhang, and W. Yang, "Joint offloading and resource allocation using deep reinforcement learning in mobile edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3454–3466, Sep./Oct. 2022.

[13] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 8, pp. 364–375, May 2022.

[14] Z. Liao and S. Cheng, "RVC: A reputation and voting based blockchain consensus mechanism for edge computing-enabled IoT systems," *J. Netw. Comput. Appl.*, vol. 209, Jan. 2023, Art. no. 103510.

[15] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[16] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Data quality-aware task offloading in mobile edge computing: An optimal stopping theory approach," *Future Gener. Comput. Syst.*, 117, pp. 462–479, Apr. 2021.

[17] N. Lin, H. Tang, L. Zhao, S. Wan, A. Hawbani, and M. Guizani, "A PDDQNLP algorithm for energy efficient computation offloading in UAV-assisted MEC," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8876–8890, Dec. 2023.
[18] G. Wu et al., "Combining Lyapunov optimization with actor–critic networks for privacy-aware IIoT computation offloading," *IEEE Internet Things J.*, early access, Jan. 22, 2024, doi: 10.1109/JIOT.2024.3357110.
[19] H. Wu, J. Chen, T. N. Nguyen, and H. Tang, "Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2117–2128, Feb. 2023.
[20] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Resource provisioning in edge computing for latency-sensitive applications," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11088–11099, Jul. 2021.
[21] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021.
[22] T. H. T. Le et al., "Auction mechanism for dynamic bandwidth allocation in multi-tenant edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15162–15176, Dec. 2020.
[23] N. C. Luong, Y. Jiao, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "A machine-learning-based auction for resource trading in fog computing," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 82–88, Mar. 2020.
[24] Y. Su, W. Fan, Y. Liu, and F. Wu, "A truthful combinatorial auction mechanism towards mobile edge computing in industrial Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1678–1691, Apr.–Jun. 2023.
[25] C. Li, Q. Zhang, and Y. Luo, "A jointly non-cooperative game-based offloading and dynamic service migration approach in mobile edge computing," *Knowl. Inf. Syst.*, vol. 65, pp. 2187–2223, Jan. 2023.
[26] M. D. Hossain et al., "Dynamic task offloading for cloud-assisted vehicular edge computing networks: A non-cooperative game theoretic approach," *Sensors*, vol. 22, no. 10, p. 3678, 2022.
[27] H. Zhou, Z. Wang, N. Cheng, D. Zeng, and P. Fan, "Stackelberg-game-based computation offloading method in cloud–edge computing networks," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16510–16520, Sep. 2022.
[28] Z. Liu and J. Fu, "Resource pricing and offloading decisions in mobile edge computing based on the Stackelberg game," *J. Supercomput.*, vol. 78, no. 6, pp. 7805–7824, 2022.
[29] Z.-L. Chang and H.-Y. Wei, "Flat-rate pricing for green edge computing with latency guarantee: A Stackelberg game approach," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.
[30] Z. Tong, X. Deng, J. Mei, L. Dai, K. Li, and K. Li, "Stackelberg game-based task offloading and pricing with computing capacity constraint in mobile edge computing," *J. Syst. Archit.*, vol. 137, Apr. 2023, Art. no. 102847.
[31] 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.
[32] Q. Wu et al., "Joint computation offloading, role, and location selection in hierarchical multicoalition UAV MEC networks: A Stackelberg game learning approach," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18293–18304, Oct. 2022.
[33] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.
[34] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[35] G. Owen *Game Theory*. Bingley, U.K.: Emerald Group Publ., 2013.
[36] M. Maschler, S. Zamir, and E. Solan, *Game Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[37] D. S. Hochba, "Approximation algorithms for NP-hard problems," *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.
[38] M. Wang, L. Zhang, P. Gao, X. Yang, K. Wang, and K. Yang, "Stackelberg game-based intelligent offloading incentive mechanism for a multi-UAV-assisted mobile edge computing system," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15679–15689, Sep. 2023.
[39] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6949–6960, Sep. 2022.
[40] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.
[41] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5444–5455, Aug. 2020.

Zhao Tong (Senior Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2014.

From 2017 to 2018, he was a Visiting Scholar with Georgia State University, Atlanta, GA, USA. He is currently an Associate Professor (the young backbone teacher of Hunan Province) with the College of Information Science and Engineering, Hunan Normal University, Changsha. He has published more than 25 research papers in international conferences and journals. His research interests include parallel and distributed computing systems, resource management, machine learning algorithm, and big data.

Dr. Tong is a Senior Member of the China Computer Federation.

Yuanyang Zhang is currently pursuing the master's degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China.

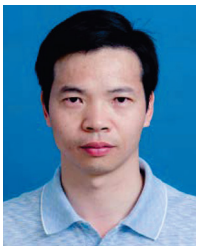
Her research interests mainly revolve around the areas of mobile-edge computing and game theory.

Jing Mei (Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2015.

She is currently an Associate Professor with the College of Information Science and Engineering, Hunan Normal University, Changsha. She has published 12 research articles in international conference and journals. Her research interests include parallel and distributed computing and cloud computing.

Wei Ai received the Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, in 2017.

Her research interests include date mining, big data, parallel computing, and cloud computing.



Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2003.

He was a Visiting Scholar with the University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 2004 to 2005. He is currently the Dean and a Full Professor of Computer Science and Technology with Hunan University, Changsha, China, and the Deputy Director of National Supercomputing Center in Changsha, Changsha. He has published more than 150 research papers in international conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, ICPP, and CCGrid. His major research areas include parallel computing, grid and cloud computing, and high-performance computing.

Prof. Li serves on the editorial board of the IEEE TRANSACTIONS ON COMPUTERS. He is an Outstanding Member of CCF.

Keqin Li (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York, New Paltz, NY, USA. He is also a Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored over 850 journal articles, book chapters, and refereed conference papers. His current research interests include cloud computing, fog computing and mobile-edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, high-performance computing, computer architectures and systems, CPU–GPU hybrid and cooperative computing, computer networking, machine learning, and intelligent and soft computing.

Dr. Li has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.