Game-Theoretic Bandwidth Allocation and Task Offloading in Cloud–Edge Collaboration

Zhao Tong, *Senior Member, IEEE*, Yuanyang Zhang, Jing Mei, Cen Chen, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—The rapid growth of Internet of Things (IoT) devices has imposed higher demands on computational capabilities, which traditional cloud computing struggles to meet in real-time scenarios due to latency issues. Mobile edge computing (MEC) addresses these challenges by processing data at the network edge, thereby reducing latency and enhancing computational efficiency. However, MEC alone is insufficient for handling complex tasks, requiring more robust solutions. This article proposes a hybrid cloud–edge computing framework that enhances system performance by integrating cloud and edge computing. A game-theoretic model is used to optimize wireless bandwidth allocation, and a Stackelberg game mechanism is introduced to incentivize task offloading. This approach orchestrates resource allocation and task offloading dynamics through game theory, ensuring cost minimization and delay requirements are met while fostering cloud–edge collaboration. Theoretical analysis demonstrates the existence of Nash equilibria in both layers of the game, ensuring the system's stability and effectiveness in complex environments. Based on this, the GA-based resource allocation and offloading (GRAO) algorithm and the iterative game-theoretic offloading (IGTO) algorithm are proposed. Experimental results validate the proposed algorithms, showing that the IGTO algorithm reduces the average cost for mobile devices (MDs) by 49.8% compared to the best baseline, while enhancing overall performance for both MEC servers and the cloud.**

***Index Terms*—Cloud–edge collaboration, resource allocation, resource pricing, Stackelberg game, task offloading.**

NOMENCLATURE

A_{nm}	Task attributes of mobile device (MD) m in cell n.
B	Maximum number of PRBs in the cell.
b_n	PRB size per unit bandwidth.
C_{nm}^{loc}	Local computation cost of MD m.
D_{nm}	Task size of MD m in cell n.
ϵ	Path loss factor.
E_{nm}^{loc}	Local computation energy of MD m.
f_n^{edge}	Total CPU frequency of server n.
f_{nm}^{edge}	Computation resources for MD m by server n.
κ_{nm}	Power consumption factor of MD m.
$\mathcal{M}, M$	Set/Number of MDs.
p_{nm}	Power of MD m to MEC server n.
ϕ_{nm}	PRBs allocated to MD m in cell n.
R_{nm}	Data rate from MD m to MEC server n.
ρ_{nm}^{e}	Offloading ratio of MD m to server n.
$T_{nm}^{c,exe}$	Execution delay of MD m's task in the cloud.
T_{nm}^{d2e}	Transmission delay from MD m to server n.
T_{nm}^{e2c}	Transmission delay from server n to cloud.
T_{nm}^{max}	Maximum task delay of MD m in cell n.
u_{nm}	Price of resources charged by server n to MD m.
a_{nm}	Binary offloading variable of MD m.
B^{total}	Total bandwidth of the cell.
C_{nm}	Cycles to execute 1 bit of MD m's task.
C_{nm}^{off}	Offloading cost of MD m in cell n.
dis_{nm}	Distance from MD m to MEC server n.
E_{nm}^{d2e}	Transmission energy from MD m to server n.
f^{cloud}	CPU frequency of the cloud.
f_{nm}^{loc}	Local CPU frequency of MD m.
g_{nm}	Channel gain from MD m to server n.
Λ_{nm}.	Interference to MD m in server n area.
$\mathcal{N}, N$	Set/Number of MEC servers.
P_{nm}^{off}	Offloading cost paid by MD m to server n.
q_{nm}	Price of resources charged by cloud to server n.
ρ_{nm}^{c}	Offloading ratio of MD m to cloud.
σ^2	Constant noise power from MDs to MEC server.
T_{nm}^{c}	Total delay of MD m's task in the cloud.
T_{nm}^{e}	Total delay of MD m's task in the server n.
T_{nm}^{loc}	Local computation delay of MD m.

Received 6 December 2025; accepted 27 February 2026. Date of publication 4 March 2026; date of current version 11 May 2026. 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; and in part by Hunan Provincial Innovation Foundation For Postgraduate under Grant CX20240548. *(Corresponding author: Zhao Tong.)*

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

Cen Chen is with the School of Future Technology, South China University of Technology, Guangzhou 510641, China, and also with the Pazhou Laboratory, Guangzhou 510330, China (e-mail: chencen@scut.edu.cn).

Keqin Li is with the College of Information Science and Engineering and National Supercomputing Center, Hunan University, Changsha, Hunan 410081, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2026.3670700

T_{nm}^{off}	Maximum delay for offloading MD m's task.
v_n	Transmission rate from server n to cloud.
dis_{nm}	Uniform (50, 1000) m.
b_n	0.1 MHz.
B^{total}	100 MHz.
ϵ	4.
κ_{nm}	10^{-25}.
f_n^{edge}	10 GHz.
f^{cloud}	100 GHz.
σ^2	−100 dBm/Hz.
λ	Interference to MD m in MEC server n from other MDs.
v_n	1 Mb/s.
u_{nm}	1×10^{-8}.

I. Introduction

THE rapid development of the Internet of Things (IoT) has exerted significant pressure on the computational capabilities of MDs. With the increasing number of IoT devices, expected to exceed 75 billion by 2025, a vast amount of data needs real-time processing and analysis [1]. However, traditional centralized cloud computing models struggle to meet the computational demands of IoT devices due to challenges such as high latency, limited bandwidth, and poor scalability. For instance, scenarios like autonomous driving, smart homes, and Industrial IoT require low-latency, high-reliability computational capabilities, placing higher demands on existing cloud computing infrastructure [2], [3].

Mobile edge computing (MEC) has emerged as a crucial solution to the IoT computational bottleneck. MEC strategically deploys computing and storage resources at the network edge, close to data sources. This deployment significantly reduces data transmission latency [4]. Research indicates that MEC can significantly decrease data processing latency to the millisecond range, simultaneously and effectively easing the burden on the core network. By processing data at the network edge, MEC not only enhances computational efficiency but also improves data privacy and security, meeting the stringent real-time and reliability requirements of IoT applications. However, MEC alone may face limitations in computational capacity and coordination of edge nodes when handling complex tasks, making it insufficient to meet the complex demands of IoT [5]. Therefore, cloud–edge collaboration has gradually become an essential solution. Cloud–edge collaboration combines cloud computing and edge computing to further enhance the performance of IoT applications [6]. This approach leverages the powerful computational capabilities of the cloud and the geographical advantages of edge nodes, enabling dynamic allocation of computational resources and load balancing. Through this collaborative mechanism, IoT devices can perform initial processing at local edge nodes and offload complex computational tasks to the cloud, effectively addressing the enormous data processing challenges brought by IoT.

Clearly, cloud–edge collaboration enhances overall computational efficiency and provides a higher quality of service (QoS), better meeting the diverse needs of the IoT era. Despite existing research in this field, several challenges remain. In cloud or MEC computing models, the primary decision involves selecting the server for task offloading. In contrast, the cloud–edge collaborative model requires deciding whether MDs should offload tasks at all, followed by determining the offloading destination, thereby complicating the offloading process [7]. Furthermore, many studies focus on optimizing offloading decisions or resource allocation, such as transmission power, bandwidth, and computational resources, to improve system performance. Most prior research, however, has addressed these aspects in isolation rather than in an integrated manner. As the number of service requests surges, the energy consumption for cooling cloud servers escalates, making the establishment of effective incentive mechanisms to encourage cloud collaboration a critical challenge [8].

To address the aforementioned challenges, this article investigates the joint optimization of offloading decisions and wireless bandwidth allocation in hybrid cloud–edge computing systems. The goal is to minimize system costs while ensuring maximum tolerable delay. To address both resource overhead and monetary costs, the game-theoretic approach is introduced for joint offloading and resource allocation. This method optimizes the tradeoff between system benefits and expenses, effectively balancing these factors. The main contributions of this article are as follows.

1) Modeling the wireless bandwidth allocation for MDs as a game and proving the existence of a Nash equilibrium. Then, considering channel conditions, an offloading mechanism for MDs is proposed, ensuring that the cost of offloading tasks is advantageous, thereby incentivizing MDs to participate in the offloading process actively.
2) Using the Stackelberg game to describe the coupling relationship between MEC servers and the cloud, where MEC servers are the followers and the cloud is the leader. This multiobjective optimization problem aims to maximize the utilities of both.
3) Proving the existence of a Nash equilibrium with delay guarantees between MEC servers and the cloud using backward induction. The utility maximization problems for MEC servers and the cloud are formulated as strictly convex optimization problems, each having a unique solution.
4) Validating the effectiveness of the proposed algorithm through experimental results, which demonstrate significant improvements in MDs' cost savings and the utilities of MEC servers and the cloud.

The structure of this article is organized as follows. Section II reviews the related work. Section III presents the system model, describing the architecture of our proposed framework. In Section IV, we introduce the game-based bandwidth allocation mechanism, detailing the methods used for efficient bandwidth distribution among MDs. Section V delves into the Stackelberg game analysis for offloading incentives between the MEC servers and the cloud. Section VI provides the simulation and result analysis.

II. Related Work

This section reviews the relevant literature on task offloading schemes, pricing strategies, and game-theoretic approaches to offloading.

A. Offloading Scheme

Task offloading involves deciding how MDs will manage tasks, which can be classified into full and partial offloading. Full offloading transfers all tasks to edge or cloud servers, thereby reducing the computational load on MDs, ideal for devices with limited capabilities. Lin et al. [9] investigated a dual-timescale problem for multi-UAV-assisted MEC, introducing a DTTD3 algorithm that jointly optimized service caching, binary computation offloading, and resource allocation to minimize system delay under energy constraints. Zhang et al. [10] addressed the joint problem of dependent task offloading and resource allocation in heterogeneous MEC, developing a two-stage alternating optimization algorithm to minimize the total cost of energy and computing expenditure.

Li et al. [11] examined a multiuser wirelessly powered fog computing network, utilizing binary offloading to enhance energy efficiency while ensuring energy causality and computing rate requirements. Han et al. [12] analyzed an energy-efficient airborne edge computing system with UAVs for binary offloading via cellular networks, aiming to minimize energy consumption through joint optimization of UAV trajectory, task offloading, and resource allocation. Partial offloading splits tasks between local processing and offloading to edge or cloud servers, making it suitable for devices with moderate processing capabilities. Liu et al. [13] proposed a distributed massive MIMO-aided multitier VEC system with partial task offloading, jointly optimizing subchannel allocation, precoding, and offloading to minimize total delay and energy consumption. Huang et al. [14] developed a distributed multiobjective *RL*-based scheme for trajectory planning and offloading scheduling in air–ground cooperative MEC, reducing task backlog and improving energy efficiency in dynamic environments. Wang et al. [15] explored partial offloading and service function chain (SFC) mapping in an NFV-enabled MEC system, proposing a dual-agent deep reinforcement learning (CDADRL) algorithm to reduce execution delay, energy consumption, and usage charges. Zhang et al. [16] presented a three-tier cloud–edge integration network architecture, optimizing system energy consumption by jointly considering user association, power allocation, task offloading, and bandwidth assignment. Abbs et al. [17] proposed a decentralized learning-based scheme (DPTORA) for partial task offloading and resource allocation in dynamic vehicular edge networks, aiming to minimize overall latency and energy consumption through task profiling and multi-RSU distribution. In contrast to these works, this article integrates both full and partial offloading, leveraging the strengths of each to enhance overall system efficiency and flexibility.

B. Pricing Strategy

In MEC architecture, the diversity of resource providers and user demands may lead to a lack of motivation among resource owners to collaborate with MEC service providers. Therefore, it is necessary to design incentive mechanisms to rationally allocate and price resources, promoting resource sharing and maximizing profits. Given that different edge nodes have their own management models and pricing strategies, it is essential to study the allocation and pricing of MEC resources from an economic perspective. Baek et al. [18] explored three dynamic pricing mechanisms for IoT edge computing resource allocation, comparing their efficiency and fairness, and provided guidance for service providers in selecting the most appropriate pricing strategy. Xie et al. [19] explored price-based task offloading using Nash Q-learning in load-imbalanced vehicular multiaccess edge computing, optimizing vehicle utility and RSU profit through dynamic pricing and multiagent reinforcement learning. Siew et al. [20] introduced an innovative sharing economy business model for MEC and proposed dynamic pricing mechanisms to optimize resource allocation and maximize social welfare or platform profit. Wang et al. [21] proposed a dependence-aware microservice deployment strategy for edge computing by leveraging an attention mechanism for system representation and a modified soft actor–critic algorithm for decision making, which jointly optimizes system cost. Habiba et al. [22] proposed a novel MEC offloading service market architecture based on a repeated generalized second-price (GSP) auction model for dynamic resource allocation. This approach accommodates multiple resource suppliers and offloading users, ensuring efficient resource distribution and individual rationality while achieving a symmetric Nash equilibrium. Extensive numerical experiments validate the model's superior performance. Wang et al. [23] proposed a truthful online combinatorial auction-based mechanism for task offloading in mobile edge computing, which ensures truthfulness, individual rationality, and computational efficiency while maximizing social welfare.

C. Game-Theoretic Offloading

Game theory in MEC aids in analyzing and optimizing interactions between edge servers and end-users in distributed systems. In MEC, participants possess different resources and objectives. Game theory allows for the design of rational incentive mechanisms and strategies, enabling all parties to make optimal decisions in resource sharing and competition. This not only promotes efficient resource management and allocation but also enhances the overall system performance, ensuring a balance of interests among participants. Chen et al. [24] introduced dynamic and static noncooperative game algorithms for computation offloading in 5G MEC systems, focusing on minimizing energy consumption and latency under QoE preferences. Lin et al. [25] proposed a noncooperative game-theoretic approach for synergistic computation offloading among edge servers with imperfect information, which effectively reduces the system completion time through distributed decision making. Wu et al. [26] proposed COMEC, a novel computation offloading scheme that leveraged potential game theory to optimize edge–cloud collaboration in multicell networks, minimizing total delay and energy costs while accounting for intercell interference. Ding et al. [27]

TABLE I
OVERVIEW OF RECENT RESEARCHES

Author	Architectural Properties			Decision Objectives			Offloading Schemes	Technology
	Cloud	Edge	Device	Energy Consumption	Offloading Cost	Utility		
Lin *et al.* [9]	Yes	Yes	Yes	Yes	No	No	Binary	Deep Reinforcement Learning
Zhang *et al.* [10]	No	Yes	Yes	Yes	Yes	No	Binary	Genetic Algorithm
Li *et al.* [11]	No	Yes	Yes	Yes	No	No	Binary	Distributed Iterative Algorithm
Liu *et al.* [13]	Yes	Yes	Yes	Yes	Yes	No	Partial	Alternate Optimization
Huang *et al.* [14]	No	Yes	Yes	Yes	Yes	No	Partial	Distributed Multi-Objective RL
Wang *et al.* [15]	No	Yes	Yes	Yes	Yes	No	Partial	Deep Reinforcement Learning
Xie *et al.* [19]	No	Yes	Yes	Yes	Yes	Yes	Binary	Q-Learning
Siew *et al.* [20]	No	Yes	Yes	No	No	Yes	Binary	Economy-Inspired Model
Wang *et al.* [21]	Yes	Yes	Yes	Yes	Yes	Yes	Binary	Q-Learning
Habiba *et al.* [22]	Yes	Yes	Yes	No	Yes	Yes	Binary	Repeated Auction
Wang *et al.* [23]	No	Yes	Yes	No	Yes	Yes	Binary	Online Auction
Chen *et al.* [24]	No	Yes	Yes	Yes	Yes	No	Partial	Non-Cooperative Game
Wu *et al.* [26]	Yes	Yes	Yes	No	Yes	Yes	Binary	Potential Game
Ding *et al.* [27]	Yes	Yes	Yes	Yes	Yes	No	Partial	Potential Game
Hu *et al.* [28]	Yes	Yes	Yes	No	No	Yes	Partial	Minority Game Approach
Tong *et al.* [29]	No	Yes	Yes	Yes	Yes	Yes	Partial	Stackelberg Game
Zhao *et al.* [30]	Yes	Yes	Yes	No	Yes	Yes	Partial	Multi-Round Stackelberg Game
8Wang *et al.* [31]	No	Yes	Yes	Yes	No	Yes	Partial	Stackelberg Game
Ours	Yes	Yes	Yes	Yes	Yes	Yes	Combined	Hybrid Games

explored the optimization of computation offloading strategies in hierarchical and horizontal end–edge–cloud computing (EECC) architectures, proposing two potential game-based algorithms to balance costs, latency, and resource utilization in various scenarios. Hu et al. [28] addressed the challenges of task offloading in distributed edge computing environments using a minority game-based approach, optimizing task processing time under conditions of incomplete information and heterogeneous task loads. Tong et al. [29] introduced a Stackelberg game-based pricing strategy to optimize task offloading and resource allocation in mobile edge computing systems, enhancing system performance and efficiency. Recent works such as Zhao et al. [30] and Wang et al. [31] had also utilized Stackelberg games in MEC environments. However, these studies were primarily based on a single-layer game paradigm. Zhao et al. [30] focused on multiround pricing and offloading in containerized MEC networks, incorporating hierarchical container image structures and discount rates. Wang et al. [31] designed an incentive mechanism for task offloading in a multi-UAV-assisted MEC system, emphasizing user satisfaction. Both of these pioneering works significantly advanced the field, but their game models concentrated on a single decision-making layer. Unlike the previously mentioned studies, this article focuses on the allocation of both wireless and computing resources using a two-layer game framework.

Table I provides a comparison of the related works.

III. System Model

This section provides a comprehensive overview of the system model, including the hierarchical model framework, the communication model, and the computation model. The symbols and their descriptions mentioned in this article are shown in Nomenclature.

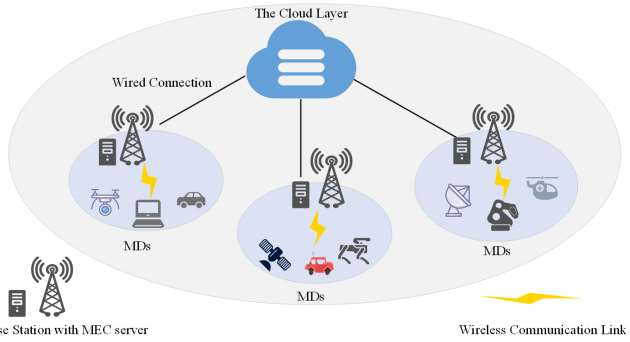


Fig. 1. Architecture of the three-layer MEC system.

A. Three-Layer Model

In this article, a hybrid cloud–edge computing architecture is investigated, which includes a cloud, multiple MEC servers, and several MDs, as shown in Fig. 1. The cloud provides abundant computing resources, though with higher transmission latency. The set of MEC servers is represented as $\mathcal{N} = \{1, 2, \ldots, N\}$. Compared to the cloud, MEC servers have limited capabilities in terms of data processing, each primarily serving MDs within their wireless coverage area. Within the service range of MEC servers, there are several MDs $\mathcal{M} = \{1, 2, \ldots, M\}$. Each MD hosts a task and offloads its computational tasks to other entities via a wireless link.

Due to the limited resources of MEC servers, a mechanism is required to determine whether to provide services for MDs. A binary variable a_{nm} is used to indicate if the MD m offloads the task. The MD m performs task offloading, $a_{nm} = 1$. In this process, ρ^e_{nm} and ρ^c_{nm} are used to represent the task load ratios of the MEC server and the cloud, respectively, with the total ratio satisfying $a_{nm} = \rho^e_{nm} + \rho^c_{nm} = 1$. When the MD m does not offload tasks (i.e., $a_{nm} = 0$), the task is fully computed

locally. The MEC servers, as the receivers of the offloaded tasks from MDs, determine the workload distribution among the MDs, itself, and the cloud.

B. Communication Model

This model assumes that MDs can communicate with MEC servers via wireless links. Each MD's task can be offloaded to the MEC server, with each MD communicating with only one MEC server at a time. For cloud-based remote processing, tasks are first transmitted from the MDs to the MEC server over a wireless link and then forwarded to the cloud via a wired link. This discussion centers on the uplink communication of tasks between MDs and MEC servers, as well as between MEC servers and the cloud, while excluding discussion of downlink communication. This decision is based on the fact that, for many applications such as facial recognition, the size of computed results is typically much smaller than that of input data.

Let $A_{nm} = \{C_{nm}, D_{nm}, T_{nm}^{max}\}$ denote a computational task in MD m within cell n, where C_{nm} represents the CPU cycles required to process 1 bit of data, D_{nm} denotes the total amount of data, and T_{nm}^{max} indicates the maximum tolerable delay. The communication model assumes an independent complex Gaussian random channel between MDs and MEC servers, which exhibits Rayleigh fading. This means the channel gain remains constant within a time slot but varies independently across different slots. In line with modern cellular standards, like 5G New Radio (NR), the system bandwidth is partitioned into physical resource blocks (PRBs), which serve as the fundamental units for spectrum allocation. The total cell bandwidth is $B_{\text{total}} = b_n \times B$, where b_n is the bandwidth of a single PRB, and B is the total number of PRBs. Although PRBs are allocated orthogonally in the time–frequency domain, with each PRB assigned exclusively to one MD, practical nonidealities such as intercarrier interference break the perfect orthogonality. This results in interuser interference within the cell [32], [33]. Therefore, the data transmission rate R_{nm} from MD m to MEC server n over this link can be calculated as

$$R_{nm} = b_n \phi_{nm} log_2 \left(1 + \frac{p_{nm} g_{nm}}{\sigma^2 + \sum_{i \in M \setminus \{m\}} p_{ni} g_{ni}}\right) \tag{1}$$

where p_{nm} denotes the transmission power of MD m, σ^2 represents the constant noise power, and g_{nm} denotes the channel gain between MD m and MEC server n. The channel gain can be defined as

$$g_{nm} = dis_{nm}^{-\epsilon} \tag{2}$$

where dis_{nm} represents the distance from MD m to MEC server n, and ϵ is the path loss factor.

For computing offloading, MDs incur additional time and energy overhead due to the transmission of tasks to the MEC server via wireless access. The transmission delay and energy consumption for offloading the task from MD m to MEC server n can be expressed as

$$T_{nm}^{d2e} = \frac{D_{nm}}{R_{nm}} \tag{3}$$

$$E_{nm}^{d2e} = p_{nm} T_{nm}^{d2e}. \tag{4}$$

Since the cloud and MEC servers are connected via wired links, the transmission rate can be set to a fixed value. Assuming the transmission rate is v_n, the time for task A_{nm} to be transmitted from MEC server n to the cloud can be expressed as

$$T_{nm}^{e2c} = \frac{\rho_{nm}^c D_{nm}}{v_n}. \tag{5}$$

We note that while v_n is fixed, the overall offloading delay T_{nm}^{off} remains adaptive to dynamic wireless conditions through R_{nm}, ϕ_{nm}, and offloading ratios.

C. Computation Model

1) Local Computation: In the local computation, the computation time and energy consumption of the task at MD m can be expressed as

$$T_{nm}^{loc} = \frac{D_{nm} C_{nm}}{f_{nm}^{loc}} \tag{6}$$

$$E_{nm}^{loc} = \kappa_{nm} \left(f_{nm}^{loc}\right)^2 D_{nm} C_{nm} \tag{7}$$

where f_{nm}^{loc} represents the computational capability of MD m, i.e., the number of CPU cycles per second. κ_{nm} denotes the power consumption factor, determined by the chip architecture.

Based on (6) and (7), the overhead of locally processing the task can be obtained as

$$C_{nm}^{loc} = w_{nm}^t T_{nm}^{loc} + w_{nm}^e E_{nm}^{loc} \tag{8}$$

where w_{nm}^t and w_{nm}^e are the weight parameters for the computation time and energy of MD m, respectively.

2) Edge Computation: In the edge computation, the computational delay in MEC server n can be expressed as

$$T_{nm}^{e,exe} = \frac{\rho_{nm}^e D_{nm} C_{nm}}{f_{nm}^{edge}} \tag{9}$$

where f_{nm}^{edge} represents the computational resources allocated to MD m by the MEC server, i.e., the CPU processing frequency. The sum of the computational resources for all MDs should not exceed the total frequency of the MEC server, i.e., $\sum_{m \in M} f_{nm}^{edge} \leq f_n^{edge}$. To simplify the model, it is assumed that the MEC server equally distributes computational resources among all MDs. This assumption, commonly used in MEC literature [13], [15], ensures analytical tractability and user fairness.

3) Cloud Computation: When the MEC server decides to offload the remaining proportion ρ_{nm}^c to the cloud, the computational delay for the cloud to process the remaining tasks is

$$T_{nm}^{c,exe} = \frac{\rho_{nm}^c D_{nm} C_{nm}}{f^{cloud}} \tag{10}$$

where f^{cloud} represents the computational capability of the cloud server.

Assuming that MDs choose to offload the task, i.e., $a = 1$, the MEC server must then determine the proportion of the task to offload to the cloud. For the MD, the decision to offload tasks is based on the total overhead. Given the uncertainty of the offloading proportions ρ_{nm}^e and ρ_{nm}^c, the MD needs to consider the larger of the computational delays from the MEC server and the cloud.

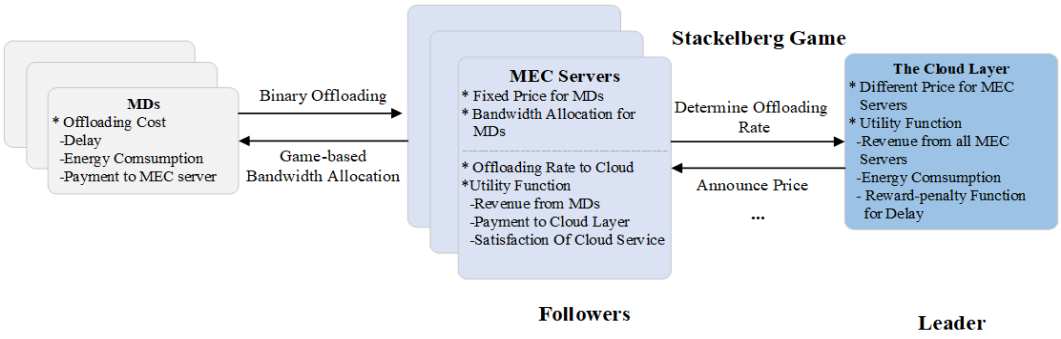


Fig. 2. Interaction process of three-layer MEC system.

If the entire task load of MD m is offloaded to the cloud, i.e., $\rho^c_{nm} = 1$, the maximum cloud delay for the IoT MD m is

$$T^c_{nm}\left(\rho^c_{nm}=1\right) = T^{d2e}_{nm} + T^{e2c}_{nm} + T^{c,exe}_{nm}. \tag{11}$$

Similarly, if the entire task load of MD m is executed on the MEC server, i.e., $\rho^e_{nm} = 1$, the maximum edge delay for the IoT MD m is

$$T^e_{nm}\left(\rho^e_{nm}=1\right) = T^{d2e}_{nm} + T^{e,exe}_{nm}. \tag{12}$$

However, when portions of the task load are offloaded to both the cloud and the MEC server, the actual end-to-end delay is uncertain and depends on the distribution of the workload between the two entities. The overall execution delay will be determined by the entity with the largest delay, as the total delay is bounded by the maximum delay among the cloud and edge layers. Therefore, the maximum offloading delay T^{off}_{nm} can be expressed as the larger of the delays from the MEC server and the cloud

$$T^{off}_{nm} = \max\left\{T^c_{nm}\left(\rho^c_{nm}=1\right), T^e_{nm}\left(\rho^e_{nm}=1\right)\right\}. \tag{13}$$

In addition, when MDs offload tasks to the MEC server for computation, the fee they need to pay to the MEC server is

$$P^{off}_{nm} = u_{nm} D_{nm} C_{nm} \tag{14}$$

where u_{nm} represents the unit price of resources charged by MEC server n to MD m.

Based on (4), (13), and (14), the overhead of offloading processing the task can be obtained as

$$C^{off}_{nm} = w^t_{nm} T^{off}_{nm} + w^e_{nm} E^{off}_{nm} + w^p_{nm} P^{off}_{nm} \tag{15}$$

where w^p_{nm} represents the weighting factor for the user's offloading payment.

IV. Game-Based Bandwidth Allocation Mechanism

In this section, a bandwidth allocation model based on game theory is first established, and then the conditions for when the MDs should perform task offloading are determined. Finally, an algorithm is designed to solve the problem. The overall system interaction process is illustrated in Fig. 2.

A. Optimization Problem of Bandwidth Allocation

In cloud–edge computing systems, resource allocation and task offloading are highly coupled. To address this, the optimization problem is divided into two independent subproblems: bandwidth allocation and task offloading, which are solved separately. This decoupling effectively reduces computational complexity and ensures optimal solutions for each subproblem. Specifically, the bandwidth allocation problem focuses on maximizing system efficiency by optimizing transmission rates, while the task offloading problem ensures the optimality of offloading strategies through a game-theoretic model.

When multiple MDs offload computational tasks simultaneously through the same wireless channel, significant interference can reduce data transmission rates. Lower transmission rates increase wireless access energy consumption and extend transmission times for offloading tasks to the MEC server, negatively impacting user experience and overall network performance. Maximizing the sum of cell data transmission rates optimizes spectrum bandwidth utilization, a crucial goal in bandwidth allocation [13], [34], [35]. This optimization enhances network throughput and efficiency, improves edge computing effectiveness, and ensures users experience higher transmission rates and lower latency when offloading tasks. Thus, our bandwidth resource allocation objective is

$$\begin{aligned} &\max \sum_{m\in M} R_{nm}\left(\phi_{nm}\right) \\ &\text{s.t.} \quad \sum_{m\in M} B_n \phi_{nm} \le B^{total} \quad \forall n \end{aligned} \tag{16}$$

where the constraint indicates that the sum of the bandwidth allocated to all MDs does not exceed the system's total bandwidth.

B. Game Theory-Based Bandwidth Allocation Model

Game theory is a mathematical framework that analyzes strategic interactions among decision makers. It focuses on optimal strategies under resource constraints and competing interests. In this optimization objective, we aim to maximize the sum of data transmission rates in a cell to ensure optimal utilization of bandwidth resources, thereby enhancing overall

network throughput and efficiency. To achieve this goal, we employ game theory for problem solving.

The three essential components of game theory include: the participants in the game, the strategy sets for each participant, and the outcomes or payoffs each participant receives based on their strategy combinations. Specifically, in this resource allocation model, the participants in the game are the MDs. The bandwidth allocation strategy set for each MD m is represented as $\phi_{nm} \in S = \{1, 2, \ldots, B\}$, which can be described as

$$\begin{aligned}&\sum_{m\in M} R_{nm}\left(\phi_{nm}\right)\\&= R_{n1}\left(\phi_{n,1}, \phi_{n,-1}\right) + \cdots\\&\quad + R_{ni}\left(\phi_{n,i}, \phi_{n,-i}\right) + \cdots + R_{nm}\left(\phi_{n,m}, \phi_{n,-m}\right)\end{aligned} \tag{17}$$

where $\phi_{n,-i}$ represents the strategy set of all MDs except MD i, i.e., $\phi_{n,-i} = \{\phi_{n,1}, \ldots, \phi_{n,i-1}, \phi_{n,i+1}, \ldots, \phi_{n,m}\}$. The utility function of MD i, denoted as $R_{ni}(\phi_{n,i}, \phi_{n,-i})$, describes the benefit that the MDs get under different bandwidth allocation strategy combinations. This function depends on the amount of PRBs obtained by the MD and the competition with other MDs. In summary, the strategy formula for the bandwidth resource game is expressed as $\Omega = \{1, 2, \ldots, M; 1, 2, \ldots, B; R_{n1}, R_{n2}, \ldots, R_{nM}\}$.

In this game problem, choosing the optimal strategy involves the concept of Nash equilibrium. When the game reaches the Nash equilibrium, no participant can achieve a higher utility by unilaterally changing their strategy. The following presents the relevant lemmas and theorems.

Lemma 1: Every finite game has at least one Nash equilibrium. A game is considered finite if it satisfies the following conditions.

1) The number of players in the game is finite.
2) Each player has a finite number of available strategies.

Theorem 1: The game $\Omega = \{1, 2, \ldots, M; 1, 2, \ldots, B; R_{n1}, R_{n2}, \ldots, R_{nM}\}$ is a finite game and has at least one Nash equilibrium.

Proof: The process of MEC servers allocating bandwidth resources to multiple MDs can be described as a game. In this game, the participants are the MDs, the number of players is finite, and each user has a finite set of bandwidth allocation strategies. Therefore, our model conforms to the definition of a finite game. According to the theory of finite games, there exists at least one Nash equilibrium. □

C. First-Phase Binary Offloading Strategy

Using the game theory model, the existence of a Nash equilibrium in bandwidth allocation has been demonstrated, ensuring an optimal solution. Next, it is necessary to compare the costs of local computation and task offloading under these bandwidth allocation strategies. This analysis will help determine whether MDs should offload their tasks to MEC servers.

Based on this insight, we define the concept of beneficial cloud computing.

Definition 1: Given the computational offloading decision a_{nm}, the offloading by MD m (i.e., $a_{nm} = 1$) is considered beneficial if the cost of executing offloading (C_{nm}^{off}) does not exceed the cost of local computation (C_{nm}^{loc}), that is, if $C_{nm}^{off} \le C_{nm}^{loc}$.

In the field of MEC, the concept of beneficial offloading is crucial. It ensures that users do not suffer performance losses when choosing to offload tasks while improving the utilization of cloud resources and increasing the revenue for telecom operators. Therefore, it is essential to ensure that computational offloading is advantageous for users. Otherwise, users will prefer local computation to reduce costs.

Theorem 2: For MD m, if the interference it receives, $\Lambda_{nm} \equiv \sum_{i\in M\setminus\{m\}} p_{ni}g_{ni}$, satisfies $\Lambda_{nm} \le I_{nm}$, then the user can achieve beneficial offloading, $a_{nm} = 1$; otherwise, $a_{nm} = 0$. That is,

$$a_{nm} = \begin{cases}1, & \text{if } \Lambda_{nm} \le I_{nm}\\ 0, & \text{if } \Lambda_{nm} > I_{nm}.\end{cases} \tag{18}$$

Threshold is

$$I_{nm} = \frac{p_{nm}h_{nm}}{2^{\frac{D_{nm}}{b_n\phi_{nm}\left(\left(C_{nm}^{loc} - w_{nm}^{e}E_{nm}^{off} - w_{nm}^{p}P_{nm}^{off}\right)/w_{nm}^{t} - T_{nm}^{else}\right)}} - 1} - \sigma^2$$

where $T_{nm}^{else} = \max\{T_{nm}^{e,exe}, T_{nm}^{e2c} + T_{nm}^{c,exe}\}$.

Proof: According to (8) and (15), and Definition 1, where $C_{nm}^{off} \le C_{nm}^{loc}$, we have

$$C_{nm}^{loc} \ge w_{nm}^{t}\left(D_{nm}/R_{nm} + T^{else}\right) + w_{nm}^{e}E_{nm}^{off} + w_{nm}^{p}P_{nm}^{off}$$

resulting in

$$\begin{aligned}&R_{nm}\\&\ge D_{nm}\Big/\left(\left(C_{nm}^{loc} - w_{nm}^{e}E_{nm}^{off} - w_{nm}^{p}P_{nm}^{off}\right)/w_{nm}^{t} - T_{nm}^{else}\right).\end{aligned}$$

Based on (1), we obtain

$$\Lambda_{nm} \le \frac{p_{nm}h_{nm}}{2^{\frac{D_{nm}}{b_n\phi_{nm}\left(\left(C_{nm}^{loc} - w_{nm}^{e}E_{nm}^{off} - w_{nm}^{p}P_{nm}^{off}\right)/w_{nm}^{t} - T_{nm}^{else}\right)}} - 1} - \sigma^2.$$

At this point, we consider task offloading to be beneficial, thus proving Theorem 2. □

D. GA-Based Resource Allocation and Offloading

Theoretical analysis has determined that MDs can achieve task offloading under specific interference conditions. Next, it is essential to optimize bandwidth allocation and offloading decisions in practical applications. Furthermore, while Theorem 1 establishes the existence of Nash equilibria, these noncooperative equilibria are generally not Pareto optimal. This inherent inefficiency motivates the need for an optimization mechanism that can approximate a system-optimal and near-Pareto efficient solution. Given the problem's complexity and multivariate nature, traditional optimization algorithms may struggle to solve it effectively. Genetic algorithms (GA) conduct solution space searches through natural evolutionary processes, demonstrating robust capability in finding global optimal solutions. Compared to traditional optimization methods, GA better avoids local optima, thereby enhancing the likelihood of discovering global optimal solutions. To enhance the efficiency and convergence of the genetic algorithm, we optimized the crossover and mutation operators and incorporated a local search mechanism to refine solutions and

Algorithm 1 GA-Based Resource Allocation and Offloading (GRAO)

Input: Parameters from the defined model
Output: ϕ_{nm} and a_{nm}
Initialize population P with random solutions
Evaluate fitness of solutions in P
while $iter < Iter^{max}$ **do**
  Select parents for crossover
  Perform crossover and mutation operations
  Evaluate fitness of offspring
  Select individuals for the next generation
  $f^{max}(t)$ = maximum fitness value in P
  $iter \leftarrow iter + 1$
  if $|f^{max}(t) - f^{max}(t-1)| \leq \epsilon$ **then**
    break
  end if
end while
return ϕ_{nm}
Based on ϕ_{nm} calculate C^{loc} and C^{off} according to Eq. (8) and Eq. (15).
if $C^{loc} > C^{off}$ **then**
  Perform offloading, set $a_{nm} \leftarrow 1$
else
  Not perform offloading, set $a_{nm} \leftarrow 0$
end if

accelerate convergence. Furthermore, Halton sequences were employed for population initialization, improving the distribution of initial solutions and thereby facilitating more effective exploration of the solution space. Therefore, the proposed algorithm is based on GA for bandwidth allocation and combines the concept of beneficial offloading to determine the offloading strategy for MDs in Algorithm 1.

The time complexity of this proposed Algorithm 1 is $O(Iter^{max} \times N)$, where $Iter^{max}$ denotes the maximum number of iterations and N is the population size. Specifically, each iteration involves operations such as selection, crossover, mutation, and fitness evaluation across the entire population, each operating at $O(N)$ time complexity. It is worth noting that the fitness evaluation, which is the core computational step, involves processing all users (U) and servers (S). This step is crucial for navigating the complex solution space of the NP-hard optimization problem we aim to solve. Moreover, the space complexity primarily depends on storing the population and fitness values, amounting to $O(N)$, indicating a linear relationship with the population size.

V. Stackelberg Game Analysis for Offloading Incentives

This section introduces the optimization objectives of MEC servers and the cloud, establishing their interaction as a Stackelberg game. Then, the backward induction method is used to solve it.

A. Cloud–Edge Collaboration Problem Formulation

When MEC servers receive computational tasks from MDs, they must meet the resource demands to process these tasks effectively. To optimize task handling, MEC servers may lease idle computing resources from the core cloud. This approach helps minimize MEC server operational costs while enhancing task processing efficiency. Simultaneously, the cloud layer benefits by efficiently utilizing its idle resources through receiving and processing these offloaded tasks. This cooperative arrangement between MEC servers and the core cloud ensures cost-efficiency for MEC operations and maximizes resource utilization within the core cloud infrastructure.

B. Utility Functions for MEC Servers and the Cloud

1) MEC Server Utility: In the interaction between MEC servers and the cloud, the primary goal of MEC servers is to offload tasks to the cloud as much as possible to reduce their own computational costs. During this process, a higher offloading ratio ρ^c_{nm} is preferable. However, from a global perspective, if tasks offloaded by all MEC servers except for server n increase, the available resources of server n will decrease. Therefore, the evaluation of cloud service incorporates the relationship between these two parameters by employing a logarithmic function commonly used in economics. Apart from assessing satisfaction with cloud services, MEC server utility also includes revenue obtained from MDs and payments for cloud resources. The utility function of MEC server n is defined as

$$U_n^{edge} = \sum_{m\in M} a_{nm} \Bigg[\left(u_{nm} - \rho^c_{nm} q_{nm}\right) D_{nm} C_{nm} + \eta \log_2 \left(1 + \frac{\rho^c_{nm} D_{nm} C_{nm}}{1 + \sum_{j\in N\setminus\{n\}} \sum_{i\in M} \rho^c_{nm} D_{nm} C_{nm}}\right)\Bigg] \tag{19}$$

where η is the satisfaction level toward the workload offloading.

2) Utility of the Cloud: For the cloud, its utility is composed of revenue, energy consumption for executing offloaded tasks, and a reward–penalty function. This function relates to the execution delay at the cloud and the experimental constraints of tasks. When $T^c_{nm} > T^{max}_{nm}$, it acts as a penalty function; when $T^c_{nm} < T^{max}_{nm}$, it functions as a reward. The utility function of the cloud is defined as

$$U^{cloud} = \sum_{j\in M\setminus\{n\}} \sum_{i\in M} a_{nm} \left(\rho^c_{nm} q_{nm} D_{nm} C_{nm} - \frac{\lambda \rho^c_{nm} D_{nm} C_{nm}}{f^c} + \psi\left(T^{max}_{nm} - T^c_{nm}\right)\right) \tag{20}$$

where ψ and λ are the controlling factors.

C. Optimization Problem for MEC Servers and the Cloud

The interaction between the cloud and the MEC server can be modeled as a Stackelberg game with a single leader and multiple followers. In the first stage, the cloud, acting as the leader, sets optimal prices for its computing resources. This pricing strategy influences the decisions of the MEC servers. In the second stage, the MEC servers, as followers, determine the optimal proportion of computing resources to lease from the cloud based on the prices set by the cloud.

1) Cloud Optimization Objectives: Based on the optimal offloading quantity of the MEC server, the cloud's optimization objectives can be formulated as

$$P1: \max \quad U_n^{cloud}\left(q_{nm}|\rho_{nm}^{*c}\right) \\ \text{s.t.} \quad q_{nm}^{min} \le \rho_{nm}^{c} \le q_{nm}^{max} \tag{21}$$

where q_{nm}^{min} and q_{nm}^{max} represent the minimum and maximum pricing by the cloud, respectively.

2) Optimization Objectives of MEC Servers: In this scenario, the offloading ratio of MEC servers depends not only on the response of the cloud but also on the behavior of other MEC servers. Therefore, the optimization objectives of MEC servers can be formulated as

$$P2: \max \quad U_n^{edge}\left(\rho_{nm}^{c}|\rho_{-n}^{*c}, q_{nm}^{*}\right) \\ \text{s.t.} \quad 0 \le \rho_{nm}^{c} \le 1 \tag{22}$$

where $\rho_{-n}^{*c} = \{\rho_{11}^{*c}, \rho_{12}^{*c}, \ldots, \rho_{n-1,m}^{*c}, \rho_{n+1,1}^{*c}, \ldots, \rho_{nm}^{*c}\}$ represents the optimal secondary task offloading ratio of MDs under other servers except MEC server n.

D. Problem Analysis

According to (21) and (22), cloud pricing directly influences the offloading decisions of MEC servers, while these offloading decisions simultaneously impact cloud pricing. This bidirectional dependence results in the coupling of the optimization problem. To address this coupling, backward induction is particularly suitable for dynamic games with sequential decision processes. This method allows us to systematically solve the game problem by working backward from the final stage to the initial stage, ensuring that each decision at any stage is optimal for subsequent stages.

Definition 2: MEC server strategy set is $\rho^{*c} = \{\rho_{11}^{*c}, \rho_{12}^{*c}, \ldots, \rho_{nm}^{*c}\}$. A Nash equilibrium in the game is reached when no participant can achieve higher utility by changing their strategy, i.e., $U_n^{edge}(\rho_{nm}^{*c}|\rho_{-n}^{*c}) \ge U_n^{edge}(\rho_{nm}^{c}|\rho_{-n}^{*c})$.

Theorem 3: In the game, as followers, there exists a Nash equilibrium point among the MEC servers, and MEC server n has an optimal offloading strategy given by

$$\rho_{nm}^{*c} = \frac{\eta}{q_{nm} D_{nm} C_{nm}} - \frac{1+\Psi}{D_{nm} C_{nm}} \tag{23}$$

where

$$\Psi = \sum_{j \in N \setminus \{n\}} \sum_{i \in M} \rho_{nm}^{c} D_{nm} C_{nm}.$$

Proof: The utility function U_n^{edge} of MEC server n with respect to the task of MD m has a first-order derivative equal to

$$\frac{\partial U_n^{edge}}{\partial \rho_{nm}^{c}} = -q_{nm} D_{nm} C_{nm} + \frac{\eta}{(1+\Psi)/D_{nm} C_{nm} + \rho_{nm}^{c}}. \tag{24}$$

The second-order derivative is equal to

$$\frac{\partial^2 U_n^{edge}}{\partial^2 \rho_{nm}^{c}} = \frac{-\eta}{\left((1+\Psi)/D_{nm} C_{nm} + \rho_{nm}^{c}\right)^2}. \tag{25}$$

In the second-order partial derivative, the denominator is a squared term greater than 0, and since the utility of the MEC server is proportional to the satisfaction function, $\eta > 0$, the numerator is negative. Thus, $(\partial^2 U_n^{edge})/(\partial^2 \rho_{nm}^{c}) < 0$. According to the relationship between convex functions and second-order partial derivatives, this function is concave. That is, when the pricing strategy of the cloud is fixed, there exists an optimal offloading strategy for the MEC server that maximizes the utility function. The optimal offloading strategy is obtained when $\partial U_n^{edge}/\partial \rho_{nm}^{c} = 0$. Thus, Theorem 3 is proven. □

Due to the boundary conditions of the offloading strategy of MDs, i.e., $0 \le \rho_{nm}^{c} \le 1$, the corresponding cloud-resource price boundaries are given by

$$q_{nm}^{min} = \frac{\eta}{1 + \Psi + D_{nm} C_{nm}} \tag{26}$$

$$q_{nm}^{max} = \frac{\eta}{1+\Psi}. \tag{27}$$

The optimal offloading strategy for the MEC server should lie between d_{nm}^{min} and d_{nm}^{max}. Therefore, the optimal offloading strategy for MEC n satisfies

$$\rho_{nm}^{*c} = \begin{cases} 0, & \text{if } q_{nm} > q_{nm}^{max} \\ \dfrac{\eta}{q_{nm} D_{nm} C_{nm}} - \dfrac{1+\Psi}{D_{nm} C_{nm}}, & \text{if } q_{nm}^{min} < x \le q_{nm}^{max} \\ 1, & \text{if } q_{nm} \le q_{nm}^{min}. \end{cases} \tag{28}$$

When the pricing of cloud resources is too high, MEC servers choose not to offload; when the pricing is too low, MEC servers offload the task completely.

Theorem 4: When MEC servers adopt the optimal offloading strategy, the utility function of the cloud has a unique Nash equilibrium point

$$q_{nm}^{*} = \sqrt{\frac{\eta\left((\lambda+\psi)/f^{c} + \psi/DnmC_{nm}\right)}{1+\Psi}}. \tag{29}$$

Proof: For the cloud utility U^{cloud}, according to (29), the first derivative with respect to the task price is

$$\frac{\partial U^{cloud}}{\partial q_{nm}} = \frac{\eta\left((\lambda+\psi)/f^{c} + \psi/D_{nm}C_{nm}\right)}{q_{nm}^{2}} - (1+\Psi). \tag{30}$$

The second derivative is

$$\frac{\partial^2 U^{cloud}}{\partial^2 q_{nm}} = \frac{-2\eta\left((\lambda+\psi)/f^{c} + \psi/D_{nm}C_{nm}\right)}{q_{nm}^{3}}. \tag{31}$$

Since $q_{nm} > 0$, $\lambda > 0$, and $\psi > 0$, the utility function is less than 0. This implies that the cloud utility function is a strictly concave function, and there exists a Nash equilibrium. Theorem 4 is thus proved. □

Theorems 3 and 4 collectively prove the existence of a unique Stackelberg equilibrium $(q_{nm}^{*}, \rho_{nm}^{c*})$ for the proposed game. This equilibrium is Pareto optimal within the hierarchical structure. This Pareto optimality stems from the fact that it is derived from strictly convex optimization problems via backward induction, ensuring that any deviation would worsen the utility of the cloud or the MEC servers.

E. IGTO Algorithm

To solve for the optimal offloading strategy ρ^{c} for MEC servers and the optimal pricing strategy q^{*} for the cloud in the studied Stackelberg game, we employ Algorithm 2 the IGTO

Algorithm 2 Iterative Game-Theoretic Offloading

1: **Input:** ϕ_{nm} and a_{nm}
2: **Output:** Offloading amounts ρ^{*c}_{nm} and resource pricing q^*_{nm}
3: Initialize q_{nm} between Eq. (26) and Eq. (27)
4: **while** $t < Iter^{max}$ **do**
5: **for** each MEC server i **do**
6: Update $\rho^c_{nm} \leftarrow \rho^c_{nm} + \alpha \frac{\partial U^{edge}_n}{\partial \rho^{*c}_{nm}}$
7: **end for**
8: Compute new price $q_{nm}(t)$ **based on** Eq. (29) **with current** ρ^c_{nm}
9: **if** $|q_{nm}(t) - q_{nm}(t-1)| < \epsilon$ **then**
10: break;
11: **else**
12: $t \leftarrow t + 1$
13: **end if**
14: **end while**
15: **return** $\rho^{*c}, , q^*$

algorithm. Initially, the cloud's starting bid is set between q^{min}_{nm} and q^{max}_{nm}. Each MEC server then iterates its strategy based on the cloud's pricing using the gradient update $\rho^c_{nm} \leftarrow \rho^c_{nm} + \alpha(\partial U^{edge}_n)/(\partial \rho^{*c}_{nm})$. The main content of this algorithm is to simulate the game process through iterations, generating a temporary optimal strategy set at each step. Ultimately, if the difference in cloud pricing between successive iterations falls within a precision threshold, we consider the game to have reached equilibrium, thus finding an approximate optimal solution for both the MEC servers and the cloud.

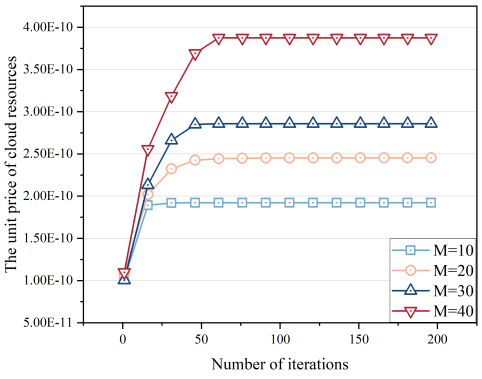
The time complexity of this algorithm is primarily determined by two nested loops. The outer loop (lines 1–14) runs at most $Iter^{max}$ times in the worst case, and during each iteration, the inner loop (lines 5–7) traverses the M MEC servers, updating the offloading amount and pricing for each. Since each operation within an iteration is constant time, the overall time complexity is $O(Iter^{max} \times M)$. This linear scalability with respect to the number of servers (M), coupled with a typically low $Iter^{max}$ required for convergence in practice, ensures the algorithm's high efficiency and suitability for real-time decision making in dynamic edge computing environments.

VI. Simulation and Result Analysis

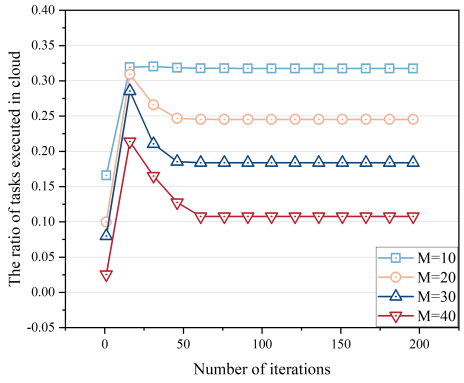
This section introduces the parameter settings and validates the proposed algorithm's performance through convergence, parameter, and comparison experiments, demonstrating its effectiveness and robustness under various conditions.

A. Parameter Settings

Simulations were conducted using Python to validate the proposed algorithm. The network architecture in these simulations includes MDs, MEC servers, and the cloud. The Opportunistic Network Environment (ONE) simulator was employed as a tool for initializing the positions of MEC servers and MDs [36], with Python scripts facilitating interaction with the simulator and integrating the initialization process into the overall simulation framework. Experimental parameters were primarily derived from [37] and [38].



(a)



(b)

Fig. 3. Iterative process of Stackelberg game. (a) Unit price of cloud resources. (b) Ratio of tasks executed in cloud.

The number of cells is set to 2 ($M = 2$), with each cell containing ten MDs. Each MD is heterogeneous. The local CPU computation capabilities of the MDs were set randomly within the range of 0.1–1.0 GHz, while task sizes ranged from 100 to 500 KB. Due to varying task complexities, the number of cycles required to execute 1 bit of a task was randomly selected between 500 and 1000 cycles/slot. In addition, the transmission power of the MDs was set between 0.05 and 0.1 W. The costs for MDs include delay, energy consumption, and payment factors, which are selected based on actual conditions. For instance, the energy consumption factor can be increased when the device battery is low. To emphasize equal importance, the weights for these three factors are set to 1/3 each. For the GRAO algorithm, the population size was set to 50 and the maximum number of generations to 200. For the IGTO algorithm, the learning rate α was set to 0.1 and the maximum number of iterations was 1000. Other experimental parameters are shown in Nomenclature.

B. Convergence Performance

The convergence experiment aims to verify whether a system with multiple MEC servers and a single cloud can reach an equilibrium point in a Stackelberg game, thereby supporting the system's game strategy. As shown in Fig. 3, as the number of iterations increases, the pricing of cloud

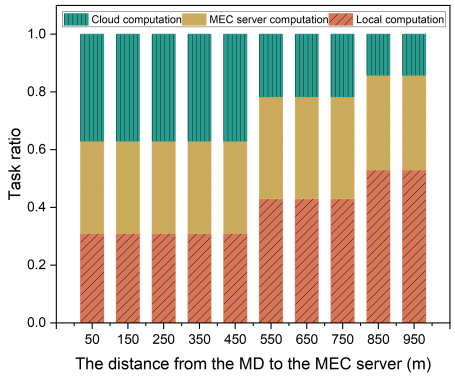


Fig. 4. Influence of distance from the MD to the MEC server.

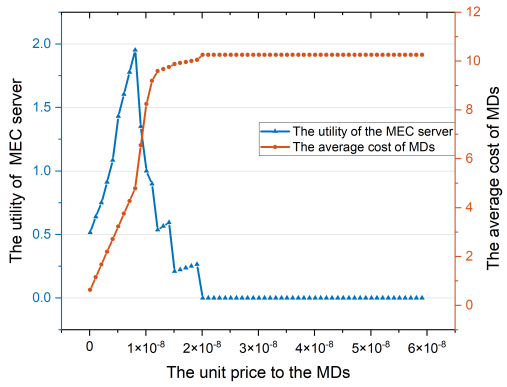


Fig. 5. Influence of unit price to MDs.

resources and the offloading ratio of MEC servers to the cloud gradually converge to an equilibrium point. Furthermore, as the number of MEC servers increases, the unit price of cloud resources rises, while the proportion of tasks executed in the cloud decreases. Specifically, this occurs because more MEC servers participating in the game leads to a higher demand for cloud execution, prompting the cloud to adopt a more aggressive pricing strategy, thus increasing resource prices. Concurrently, intensified resource competition causes the cost of offloading to the cloud to rise, resulting in a decreased offloading ratio to the cloud. Finally, when the number of MDs is relatively small (e.g., ten), resource competition remains weak, allowing prices and offloading rates to stabilize quickly. However, as the number of MEC servers grows, the offloading rate initially fluctuates before eventually reaching equilibrium due to increased resource competition.

C. Parameter Experiments

Fig. 4 illustrates the impact of the distance between MDs and the MEC server on the offloading ratios for local computation, MEC server computation, and cloud computation. First, it is observed that as the distance increases, the proportion of local computation gradually rises. This is because increasing distance deteriorates channel conditions, leading to higher transmission delays and energy consumption. As a result, the cost of offloading tasks to the MEC server or the cloud becomes higher, making MDs more inclined to perform computations locally to minimize transmission delay and energy usage. In addition, with increasing distance, the proportion of cloud computation also gradually decreases. The utility function of the cloud includes a penalty function related to execution time. As the user distance grows, the delay in transmitting tasks to the cloud increases, resulting in a larger penalty function. Consequently, the proportion of tasks offloaded to the cloud decreases to avoid the extra costs and reduced utility associated with high delays.

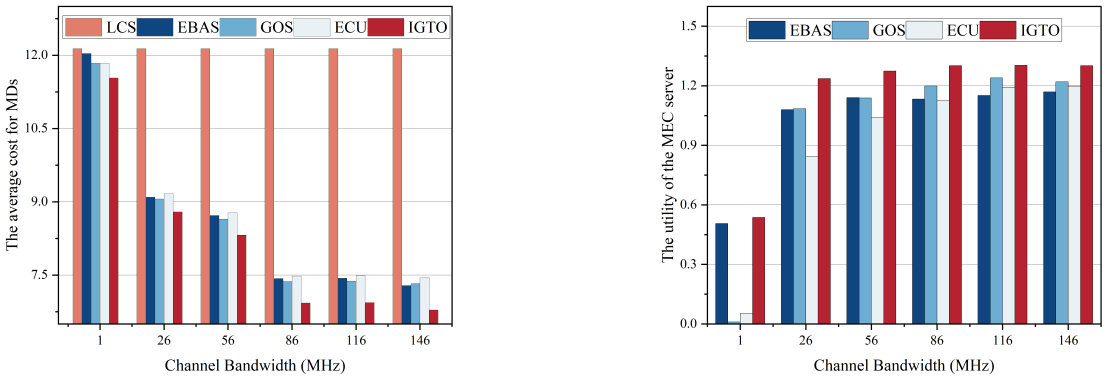
Fig. 5 illustrates the impact of the MEC server's unit pricing on the average cost for MDs and the utility of the MEC server. As the resource pricing set by the MEC server increases progressively, the user expenditure initially rises and then stabilizes. At the outset, higher resource prices lead users to offload more tasks, causing an increase in costs. Subsequently, as prices continue to rise, more users opt for local computation, which slows the increase in costs and eventually stabilizes them. Simultaneously, the MEC server's utility first increases and then decreases, eventually approaching zero. The rise in prices reduces the economic benefit of offloading tasks, leading to a decrease in the number of users opting for offloading, and thus the server's utility diminishes.

It is observed that the MEC server's utility sharply declines after an initial local increase. This is primarily because users' decisions to offload tasks depend on a comparison between the local computation costs and the offloading costs. Within a certain price range, the number of users choosing to offload tasks remains constant, leading to an increase in server utility. However, when prices become too high, users reduce their offloading activities, and server utility decreases. Ultimately, the analysis reveals an optimal resource pricing point that maximizes MEC server utility. When the price is too low, user willingness to offload tasks is high, but server revenue is insufficient. Conversely, when the price is too high, users choose not to offload tasks, resulting in reduced server utility. Therefore, there exists an optimal pricing equilibrium that maximizes the utility of the edge server, which was determined and utilized in our experiments.

D. Comparative Experiments

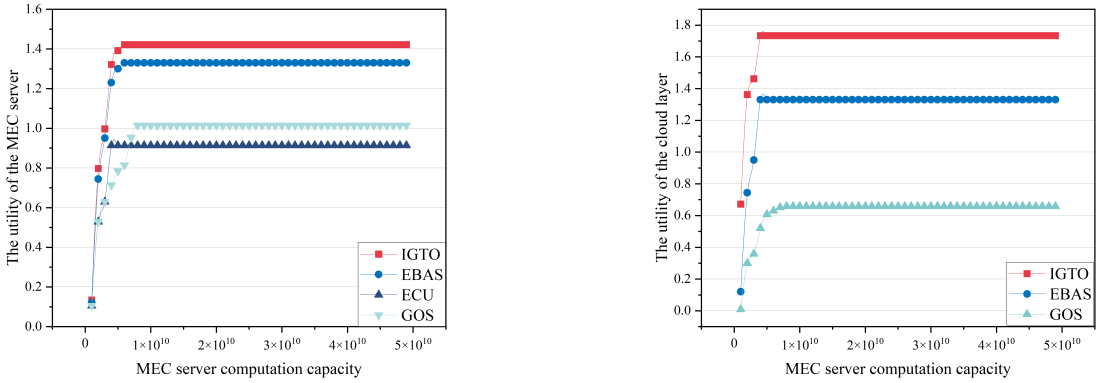
To demonstrate the superiority of the proposed algorithm, we compared it with the following four algorithms across three dimensions.

1) *Equal Bandwidth Allocation Strategy (EBAS):* This algorithm allocates bandwidth resources equally among all MEC servers without considering actual load or demand.
2) *Game-Based Offloading Strategy (GOS):* This algorithm determines the optimal offloading strategy based on game theory but does not optimize the pricing strategy.
3) *Local Computing Strategy (LCS):* All tasks are processed locally on the MDs without any offloading to MEC servers or the cloud.
4) *Edge Computing Unloading (ECU):* Under this algorithm, MDs decide to offload tasks to MEC servers or process them locally based on cost considerations, without offloading tasks to the cloud.



(a) (b)

Fig. 6. Comparison with channel bandwidth. (a) Average cost for MDs. (b) Utility of MEC server.



(a) (b)

Fig. 7. Comparison with computation capacity. (a) Average cost for MDs. (b) Utility of MEC server.

Fig. 6 illustrates the average cost of MDs and MEC server utility under varying MEC server computational capacities for five different strategies. Initially, it is observed that with the increase in total bandwidth, the average MDs cost under each strategy gradually decreases and eventually stabilizes. Compared to the local computing strategy, the strategies involving task offloading significantly reduce user costs. As the bandwidth increases, the MEC server utility also gradually rises and eventually stabilizes. This phenomenon is due to the reduction in offloading costs with increased bandwidth, leading to an increase in the number of MDs offloading tasks. As the number of MDs stabilizes, both the costs and the MEC server utility stabilize as well. In the EBA algorithm, the failure to consider different users' bandwidth needs leads to wastage of bandwidth resources for some users, while others cannot efficiently offload tasks due to insufficient bandwidth. This increases the overall communication cost, resulting in high user costs and low MEC server revenue. In the GOS algorithm, the lack of further optimization in cloud pricing results in unreasonable resource prices in certain situations, affecting the offloading ratio of the MEC server and thereby impacting overall utility. In the ECU algorithm, the MEC server does not fully utilize cloud resources, and there is no parallel processing between the MEC server and the cloud. This increases task execution delay and reduces task execution efficiency.

Fig. 7 shows the utility of the MEC server and cloud under varying MEC server computational capacities across five different strategies. All strategies exhibit a similar trend: as the computational capacity of the MEC server increases, the utility of both the MEC server and the cloud initially rises and then stabilizes. This is because enhanced computational capacity reduces offloading costs, allowing more MDs to offload tasks to the MEC server, thereby increasing overall system utility. However, once the computational capacity reaches a certain level, further enhancements have a limited effect on the offloading volume, causing utility to stabilize. Under varying MEC server computational capacities, the proposed algorithm excels in resource allocation and pricing optimization through game-theoretic offloading and cloud pricing strategies. This significantly enhances the utility of both the MEC server and the cloud, making it superior to the other compared algorithms across various metrics.

Fig. 8 shows the impact of varying numbers of MDs on the utility of MEC servers and the cloud across five different strategies. As the number of MDs increases, more tasks are offloaded to the MEC servers, which subsequently transfer some of these tasks to the cloud, thereby enhancing the utility

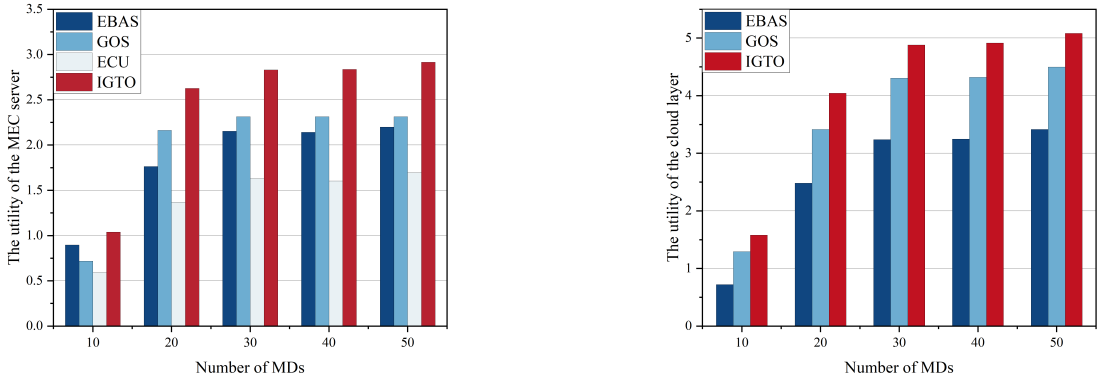


Fig. 8. Comparison with number of MDs. (a) Average cost for MDs. (b) Utility of MEC server.

of the MEC servers. However, with the intensified competition among MEC servers, the prices for cloud resources also rise, leading to an increase in the cloud's utility. Despite this, the overall resources of MEC servers are limited, causing the number of tasks offloaded to MEC servers to eventually balance out. As a result, the utility trends for both MEC servers and the cloud stabilize. The proposed algorithm demonstrates superior performance in resource allocation and pricing optimization, leveraging game-theoretic offloading and cloud pricing strategies to significantly enhance the utility of both MEC servers and the cloud, outperforming other compared algorithms in various metrics.

VII. Conclusion

This article proposes a hybrid cloud–edge computing framework designed to overcome computational and latency challenges in IoT environments. By incorporating a game-theoretic model for wireless bandwidth allocation and a Stackelberg game-based task offloading mechanism, the proposed approach efficiently balances system cost and delay constraints. Theoretical analysis establishes the existence of Nash equilibria, which ensures system stability and operational efficiency across different architectural layers. To evaluate performance, extensive simulations were conducted using the proposed GRAO and IGTO algorithms. Results show that the IGTO algorithm reduces the average cost for mobile devices by 49.8% and improves the utility of MEC servers by 34.2% compared to the best baseline, underscoring its substantial performance benefits. These findings confirm that the proposed algorithms enhance cost efficiency for mobile devices while boosting performance for both MEC servers and the cloud. This research not only contributes to optimized resource allocation and task offloading in hybrid cloud–edge systems but also offers practical algorithms applicable to real-world IoT scenarios, enabling high performance and low latency. Future work will focus on evaluating the scalability of the proposed algorithms in highly dynamic and large-scale IoT environments to further strengthen their practical applicability.

References

[1] E. H. Houssein, M. A. Othman, W. M. Mohamed, and M. Younan, "Internet of Things in smart cities: Comprehensive review, open issues, and challenges," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34941–34952, Nov. 2024.

[2] M. M. Sadeeq, N. M. Abdulkareem, S. R. M. Zeebaree, D. Mikaeel Ahmed, A. S. Sami, and R. R. Zebari, "IoT and cloud computing issues, challenges and opportunities: A review," *Qubahan Academic J.*, vol. 1, no. 2, pp. 1–7, Mar. 2021.

[3] B. Cinar and J. P. Bharadiya, "Cloud computing forensics; challenges and future perspectives: A review," *Asian J. Res. Comput. Sci.*, vol. 16, no. 1, pp. 1–14, May 2023.

[4] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[5] 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.

[6] J. Yao et al., "Edge-cloud polarization and collaboration: A comprehensive survey for AI," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 6866–6886, Jul. 2022.

[7] A. Ometov, O. Molua, M. Komarov, and J. Nurmi, "A survey of security in cloud, edge, and fog computing," *Sensors*, vol. 22, no. 3, p. 927, Jan. 2022.

[8] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 6, pp. 742–763, Dec. 2020.

[9] N. Lin, X. Han, A. Hawbani, Y. Sun, Y. Guan, and L. Zhao, "Deep reinforcement learning-based dual-timescale service caching and computation offloading for multi-UAV assisted MEC systems," *IEEE Trans. Netw. Service Manage.*, vol. 22, no. 1, pp. 605–617, Feb. 2025.

[10] G. Zhang et al., "Dependency-aware joint task offloading and resource allocation in heterogeneous mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 23, no. 12, pp. 19444–19458, Dec. 2024.

[11] H. Li, K. Xiong, Y. Lu, B. Gao, P. Fan, and K. B. Letaief, "Distributed design of wireless powered fog computing networks with binary computation offloading," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2084–2099, Apr. 2023.

[12] H. Han, C. Zhan, J. Lv, and C. Xu, "Energy minimization for cellular-connected aerial edge computing system with binary offloading," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9558–9571, Mar. 2024.

[13] Y. Liu, S. Liang, K. Wang, W. Chen, Y. Li, and G. K. Karagiannidis, "Distributed massive MIMO-aided task offloading in satellite-terrestrial integrated multi-tier VEC networks," *IEEE Trans. Veh. Technol.*, vol. 74, no. 9, pp. 14882–14886, Sep. 2025.

[14] Y. Huang, M. Dong, Y. Mao, W. Liu, and Z. Gao, "Distributed multi-objective dynamic offloading scheduling for air–ground cooperative MEC," *IEEE Trans. Veh. Technol.*, vol. 73, no. 8, pp. 12207–12212, Aug. 2024.

[15] X. Wang, H. Xing, F. Song, S. Luo, P. Dai, and B. Zhao, "On jointly optimizing partial offloading and SFC mapping: A cooperative dual-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 8, pp. 2479–2497, Aug. 2023.

[16] Y. Zhang, H. Zhang, K. Sun, J. Huo, N. Wang, and V. C. M. Leung, "Partial computation offloading in satellite based three-tier cloud-edge integration networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 2, pp. 836–847, Feb. 2023.

[17] Z. Abbas, S. Xu, and X. Zhang, "An efficient partial task offloading and resource allocation scheme for vehicular edge computing in a dynamic environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 2, pp. 2488–2502, Feb. 2025.

[18] B. Baek, J. Lee, Y. Peng, and S. Park, "Three dynamic pricing schemes for resource allocation of edge computing for IoT environment," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4292–4303, May 2020.

[19] J. Xie, F. Zheng, W. Wen, and Y. Jia, "Price-based task offloading for load-imbalance vehicular multi-access edge computing," in *Proc. IEEE 99th Veh. Technol. Conf.*, Jun. 2024, pp. 1–6.

[20] M. Siew, D. Cai, L. Li, and T. Q. S. Quek, "Dynamic pricing for resource-quota sharing in multi-access edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2901–2912, Oct. 2020.

[21] C. Wang et al., "Dependency-aware microservice deployment for edge computing: A deep reinforcement learning approach with network representation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14737–14753, Dec. 2024.

[22] U. Habiba, S. Maghsudi, and E. Hossain, "A repeated auction model for load-aware dynamic resource allocation in multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 7, pp. 7801–7817, Jul. 2024.

[23] X. Wang et al., "Truthful online combinatorial auction-based mechanisms for task offloading in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 7, pp. 6488–6502, Jul. 2025.

[24] J. Chen, Q. Deng, and X. Yang, "Non-cooperative game algorithms for computation offloading in mobile edge computing environments," *J. Parallel Distrib. Comput.*, vol. 172, pp. 18–31, Aug. 2023.

[25] 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. 2025.

[26] L. Wu, P. Sun, Z. Wang, Y. Li, and Y. Yang, "Computation offloading in multi-cell networks with collaborative edge-cloud computing: A game theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2093–2106, Mar. 2024.

[27] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[28] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2139–2154, Sep. 2020.

[29] 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.

[30] M. Zhao, Z. Yang, Z. He, F. Xue, and X. Zhang, "Multi-round Stackelberg game-based pricing and offloading in containerized MEC networks," *IEEE Trans. Green Commun. Netw.*, vol. 9, no. 1, pp. 191–206, Mar. 2025.

[31] 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.

[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] 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.

[34] M. Zeng, R. Du, V. Fodor, and C. Fischione, "Computation rate maximization for wireless powered mobile edge computing with NOMA," in *Proc. IEEE 20th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2019, pp. 1–9.

[35] S. Mao et al., "Computation rate maximization for intelligent reflecting surface enhanced wireless powered mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10820–10831, Oct. 2021.

[36] A. Keränen, T. Kärkkäinen, and J. Ott, "Simulating mobility and DTNs with the ONE," *J. Commun.*, vol. 5, no. 2, pp. 92–105, Feb. 2010.

[37] Z.-L. Chang, C.-Y. Wang, and H.-Y. Wei, "Flat-rate pricing and truthful offloading mechanism in multi-layer edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6107–6121, Sep. 2021.

[38] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

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 at Georgia State University, Atlanta, GA, USA. He is currently a Professor with the College of Information Science and Engineering of Hunan Normal University, Changsha, and is the Young Backbone Teacher of Hunan Province, China. 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 algorithms, and big data.

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

Yuanyang Zhang received the B.S. degree in software engineering from the Central South University of Forestry and Technology, Changsha, China, in 2022. She is currently pursuing the master's degree with the College of Information Science and Engineering, Hunan Normal University, Changsha.

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

Jing Mei 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 conferences and journals. Her research interests include parallel and distributed computing and cloud computing.

Cen Chen (Senior Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2019.

Currently, he is a Professor with the School of Future Technology, South China University of Technology, Guangzhou, China. He has published several research articles in international conferences and journals on machine learning algorithms and parallel computing, such as MICRO, HPCA, DAC, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, AAAI, ICDM, ICPP, ICDCS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE IEEE TRANSACTIONS ON CYBERNETICS. His research interests include parallel and distributed computing, computer architecture, machine learning, and deep learning.

Dr. Chen is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of computer science with the State University of New York, New York, NY, USA. He is also a Distinguished Professor at Hunan University, Changsha, China. He has authored or co-authored over 850 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. 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 currently serves or has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.