Incentive-Driven Task Offloading and Collaborative Computing in Device-Assisted MEC Networks

Yang Li, Xing Zhang, *Senior Member, IEEE,* Bo Lei, Qianying Zhao, Min Wei, Zheyan Qu, and Wenbo Wang, *Senior Member, IEEE*

***Abstract*—Edge computing (EC), positioned near end devices, holds significant potential for delivering low-latency, energy-efficient, and secure services. This makes it a crucial component of the Internet of Things (IoT). However, the increasing number of IoT devices and emerging services place tremendous pressure on edge servers (ESs). To better handle dynamically arriving heterogeneous tasks, ESs and IoT devices with idle resources can collaborate in processing tasks. Considering the selfishness and heterogeneity of IoT devices and ESs, we propose an incentive-driven multi-level task allocation framework. Specifically, we categorize IoT devices into task IoT devices (TDs), which generate tasks, and auxiliary IoT devices (ADs), which have idle resources. We use a bargaining game to determine the initial offloading decision and the payment fee for each TD, as well as a double auction to incentivize ADs to participate in task processing. Additionally, we develop a priority-based inter-cell task scheduling algorithm to address the uneven distribution of user tasks across different cells. Finally, we theoretically analyze the performance of the proposed framework. Simulation results demonstrate that our proposed framework outperforms benchmark methods.**

***Index Terms*—Device-assisted mobile edge networks, incentive-driven, multi-level task allocation, bargaining game, double auction.**

I. Introduction

A. Research Background and Motivation

THE rapid development of the Internet of Things (IoT) and 5G technologies has led to an increased demand for computationally intensive and latency-sensitive tasks, such as face recognition, Internet of Vehicles, and AR/VR. Resource-constrained IoT devices cannot process these tasks locally. Traditional cloud computing introduces long backhaul link latency, failing to provide real-time computation [1]. Therefore, mobile edge computing (MEC) has emerged as a promising paradigm to support various 5G service scenarios, particularly low-latency services [2]. However, due to the limited computational capabilities of edge servers (ESs) and the increasing number of IoT devices, meeting the demand for numerous latency-sensitive, computationally intensive tasks with resource-limited ESs remains a significant challenge. Additionally, cost constraints prevent the continuous expansion of ES resources [3]. In this context, device-assisted edge computing has gained increased scholarly attention [4]–[13]. Leveraging the diversity among IoT devices and incorporating those with idle resources as resource providers to assist ESs in processing tasks can enhance the computing capacity of edge systems while improving the utilization of end-device resources [4].

Current research on device-assisted MEC networks primarily focuses on computation offloading and content caching [4]–[13]. In device-assisted computation offloading, users can offload tasks to edge servers (ESs) or nearby auxiliary devices. Offloading tasks to nearby auxiliary devices over device-to-device (D2D) links can reduce the load on the cellular network infrastructure and free up cellular bandwidth for other uses [4]–[7]. The main strategy of current device-assisted MEC caching approaches is to prioritize content placement and delivery from other end devices through D2D communication links. This is followed by content placement and delivery from the MEC server, and lastly, from the central cloud [8]–[13]. Generally, content caching has two main aspects: content placement and content delivery. Content placement studies focus on designing methods for optimally storing (placing) content files in the caches of ESs and IoT devices. Conversely, content delivery studies concentrate on transmitting the requested files to end devices.

Overall, device-assisted MEC research has made significant progress in computational offloading and content caching. Nevertheless, device-assisted MEC is still an emerging research area, and current state-of-the-art research has significant limitations. For example, most studies consider simple schemes, usually involving only one ES. Collaboration among different ESs is intriguing but rarely addressed. Existing collaboration schemes among ESs are excessively complex, preventing their application in realistic scenarios. Additionally, since assisting with processing tasks consumes energy, auxiliary IoT devices are reluctant to help ESs without incentives. However, research on incentive mechanisms in device-assisted edge computing is limited. To address this, we propose an incentive-driven multi-level task allocation scheme to encourage cooperation among participants for efficient task processing. Specifically, our research addresses the following challenges:

Manuscript received 4 November 2024; accepted 25 November 2024. This work is supported by the National Science Foundation of China under Grant 62071063, 62271062, and by the BUPT Excellent Ph.D. Students Foundation under Grant CX20241066. (Corresponding author: Xing Zhang.)

Y. Li, X. Zhang, Z. Qu, and W. Wang are with the School of Information and Communications Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: ly209991@bupt.edu.cn; zhangx@ieee.org; zheyanqu@bupt.edu.cn; wbwang@bupt.edu.cn).

B. Lei, Q. Zhao, and M. Wei are with Beijing Branch of China Telecom Co., Ltd., Beijing 100032, China (e-mail: leibo@chinatelecom.cn; zhaoqy50@chinatelecom.cn; weim6@chinatelecom.cn).

1) Device-assisted edge computing networks involve various participants with differing intentions. Without suitable incentives, efficient collaboration among participants may be hindered.
2) Varied task requirements, execution mode choices, and resource constraints result in a combinatorial optimization and mixed-integer problem, which is NP-hard.
3) In a multi-ES scenario, collaboration among ESs can enhance system performance but also introduces additional computational complexity, requiring a balance between algorithm performance and complexity.

B. Related Work

1) Incentive Mechanism for Resource Allocation: The incentive mechanisms for resource allocation in current research can be broadly categorized into Stackelberg game-based approaches [14]–[19], auction-based approaches [20]–[23], contract-theory-based approaches [24]–[28], and bargaining-based approaches [29]–[32].

The Stackelberg game divides players into leaders and followers. The leader acts first, and the followers choose the optimal response based on the leader's strategy. The leader's optimal strategy combines with the followers' optimal responses to form a Stackelberg equilibrium. Zhou *et al.* [14] used the Stackelberg game to model interactions between edge service providers (ESPs) and mobile users (MUs). They used backward induction to analyze the game problem, aiming to maximize the utility of both ESPs and MUs. Fan *et al.* [15] proposed a two-stage pricing scheme for MEC network slicing. The authors used a Stackelberg game to model the interactions between MEC network service providers (MEC-NSPs) and user devices (UDs). In the first stage, they optimized resource allocation, and in the second stage, they determined the slice pricing. Similarly, the studies in [16]–[19] used the Stackelberg game to design service pricing for resource providers.

Auction theories offer an efficient way to create a market where resource seekers bid for one or more resources, effectively providing a theoretical basis for resource allocation. Wang *et al.* [20] designed an online profit-maximizing multi-round auction mechanism for trading resources between edge clouds (sellers) and mobile devices (buyers) in competitive environments. Habiba *et al.* [21] proposed a repeated auction

TABLE I
OVERVIEW OF THE RELATED WORK.

Incentive Mechanism for Resource Allocation

Research	Incentive Mechanism	Participants	Objective
[14]–[19]	Stackelberg game	ESs and task IoT devices (TDs)	Design service pricing for ESs and maximize the utility of both ESs and TDs.
[20]–[23]	Auction theory	ESs and TDs	Ascertain resource allocation and transaction prices in MEC networks.
[24]–[28]	Contract theory	ESs and auxiliary IoT devices (ADs)	Motivate idle devices to provide resources and maximize the utility of ESs.
[29]–[32]	Bargaining game	ESs and TDs	Ascertain resource allocation and maximize the utility of both ESs and TDs.
Our study	Bargaining game, priority-based scheduling, and double action	ESs, TDs, and ADs	Motivate horizontal collaboration among ESs and vertical collaboration between ESs and ADs, and maximize the system utility.

Task Allocation in Device-Assisted MEC

Research	Scenario	Technical method	Objective
[33]	Single ES	Lyapunov optimization and variable substitution technique	Minimize the network-wide response latency and energy consumption.
[4]	Single ES	Deep reinforcement learning and graph theory	Minimize the maximum processing latency for all tasks.
[5]	Single ES	Iteration-based programme	Minimize the sum of task execution latency of all the devices.
[34]	Single ES	Block descent and potential game	Improve resource utilization and reduce computing latency.
[7]	Multiple ESs	Multi-armed Bandit	Minimize the weighted sum of task latency and service migration costs.
[6]	Multiple ESs	Game theory	Minimize the overall offloading cost in terms of processing delay and energy consumption.
[35]	Multiple ESs	Graph theory	Maximize the system utility.
Our study	Multiple ESs	Multi-level decision-making, which integrates bargaining game, priority-based scheduling, and double action	Motivate horizontal collaboration among ESs and vertical collaboration between ESs and ADs, and maximize the system utility.

model for dynamic resource allocation in MEC networks. The studies in [22] and [23] employ a double auction mechanism to ascertain matching outcomes and transaction prices for IoT devices and edge servers.

Contract theory offers valuable tools for designing incentive mechanisms. It involves two parties: agents (e.g., user equipment (UE)) and trustors (e.g., base stations (BSs) or ESs). This mechanism avoids the limitations of long convergence times and the need for multiple information exchanges between agents and trustors, which can result from iteration-based schemes like the Stackelberg game and auction mechanism. However, it must address the challenges posed by information asymmetry. Chen *et al.* [24] proposed a signaling-based incentive mechanism using contract theory to address information asymmetry in the D2D computation offloading problem. The authors in [25] proposed a contract incentive strategy based on deep reinforcement learning for a wireless-powered, UAV-assisted backscattering MEC system. Their objective was to maximize the long-term utility of hotspots while maintaining the stability of energy queues. Many studies have used contract theory to design incentive mechanisms, primarily to motivate idle devices to provide resources [26]–[28].

The bargaining game is a cooperative game with the features of incentive, self-enforcement, and satisfaction for all players [29]. It guarantees fairness and Pareto-optimality of the outcome and captures the potential of coordination among bargainers [30]. Shi *et al.* [31] investigated the service deployment problem in fog computing using the bargaining game concept to enhance service providers' revenues. Meskar *et al.* [32] proposed a fair resource allocation mechanism for heterogeneous servers in MEC networks. They employed the Kalai-Smorodinsky bargaining solution to ensure fairness properties, including envy-freeness, Pareto optimality, strategy-proofness, and sharing incentives.

However, the previous studies mainly focus on the interactions between ESs and task IoT devices (TDs), or ESs and auxiliary IoT devices (ADs), by designing incentives to motivate cooperation between two parties with conflicting interests. This paper considers a scenario involving three types of participants: ESs, TDs, and ADs. In this scenario, different ESs share common interests, while conflicts of interest exist among TDs, among ADs, and between TDs, ADs, and ESs. Therefore, none of the previously proposed incentive mechanisms can be directly applied. Unlike previous work, this paper thoroughly analyzes the interactions among the three types of participants and proposes a multi-level decision-making framework. This framework integrates bargaining games, priority-based scheduling, and double auctions, effectively leveraging the cooperative and competitive relationships among the participants to maximize system utility.

2) Task Allocation in Device-Assisted MEC: Researchers have done much work on task allocation in device-assisted MEC networks for single ES and multiple ES scenarios.

For scenarios involving a single ES, Peng *et al.* [33] investigated the joint optimization of collaborative partial offloading, transmission scheduling, and task allocation for device-assisted MEC networks. They proposed an online resource coordination and allocation scheme (ORCAS). Our previous work [4] proposed a joint task partitioning and parallel scheduling framework. This framework first divides computationally intensive tasks into multiple subtasks. Then, it schedules these subtasks to be processed in parallel on multiple ADs and the ES, enabling collaboration among the ADs and the ES. The studies in [5] and [34] both focused on minimizing the overall task execution delay within single-cell, device-assisted MEC networks. However, a single ES has limited computational resources. Although some studies [36]–[40] have investigated cloud cooperation to compensate for the limited resources of a single ES, this approach introduces significant backhaul delay. Leveraging collaboration among multiple ESs is an effective way to address this challenge.

For scenarios involving multiple ESs, Dai *et al.* [7] proposed a collaborative offloading framework that integrates migration cost and offloading willingness in a device-assisted MEC network to minimize the system cost. However, instead of leveraging collaboration among ESs, this work uses a learning-based approach to prompt IoT devices to select the best computing node. Yang *et al.* [6] proposed a game-theory-based offloading approach that treats the computation offloading process in a device-assisted MEC network as a competitive game for resources, thereby minimizing system overhead and the execution cost of individual tasks. However, as the number of ESs and IoT devices increases, the long convergence time of the game-theoretic algorithm can prevent its application in real-world scenarios. Hou *et al.* [35] proposed an online task allocation algorithm to optimize the task allocation policy in a device-assisted MEC network, maximizing the system utility. However, the proposed scheme only considers serially arriving tasks, meaning only one task's processing position can be determined at a time. In real IoT systems, tasks from multiple users often arrive simultaneously, requiring the processing locations of multiple tasks to be determined concurrently.

To our knowledge, few studies consider collaboration among multiple cells in device-assisted MEC networks. As the number of cells increases, the linear growth in the number of TDs, ADs, and ESs complicates problem-solving. The most relevant studies [6], [35] proposed schemes that were highly sensitive to the number of cells, rendering them unsuitable for online use in device-assisted MEC networks with a large number of cells. The challenge in implementing online assignment of concurrent task requests in multi-cell device-assisted MEC networks lies in optimizing system utility while considering algorithm complexity. Unlike previous work, this paper introduces a multi-level decision-making framework for managing concurrent task requests in multi-cell device-assisted MEC networks, enabling efficient collaboration among ESs, ADs, and TDs. Moreover, this framework allows the system to handle varying task complexity and resource availability, remains insensitive to the number of cells. It also adapts algorithmic complexity based on user request distribution by either concluding early or bypassing the second level of decision-making, demonstrating high flexibility and scalability.

Table I summarizes the closely related studies on incentive mechanisms for resource allocation and task allocation in device-assisted MEC networks, comparing these works with ours.

C. Contribution and Organization

In this paper, we investigate the device-assisted MEC network depicted in Fig. 1. To promote effective collaboration among various system participants and achieve efficient task processing, we propose an incentive-driven multi-level task allocation scheme. The main contributions of this paper are summarized as follows:

1) We propose a hierarchical task assignment scheme with collaborative computing for device-assisted MEC networks. This multi-level task allocation strategy enables scalable and low-complexity scheduling decisions for concurrent heterogeneous tasks.
2) At the first level of decision-making, we model the interaction between each TD and the ES in each cell as a bargaining game to maximize their utility. Through bargaining, the offloading decision and payment fee for each TD are determined. This mechanism ensures fairness and Pareto optimality in task offloading between TDs and ESs.
3) If some ESs are overloaded after the first level of decision-making, they will collaborate with underloaded ESs to reassign tasks. We design a priority-based task filtering scheme and task scheduling algorithm for the second level of decision-making.
4) If all ESs' resources are still insufficient, we introduce a double auction mechanism to incentivize ADs to assist ESs in the third level of decision-making.
5) Numerical simulations show that the proposed scheme outperforms the benchmark methods in system utility, task offloading ratio, algorithm execution delay, and load balancing capability.

The rest of the paper is organized as follows. Section II introduces the system model and problem formulation. Section III presents the proposed multi-level task allocation algorithm. Section IV presents the theoretical analysis of the proposed framework. Section V discusses the simulation results and performance analysis. Finally, Section VI concludes the paper. The code is available at https://github.com/CPNGroup/Multi_Level_EC.

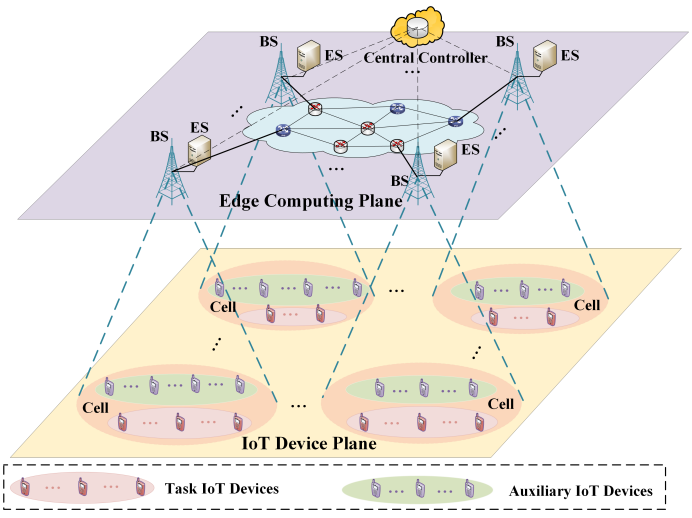


Fig. 1. System model of the device-assisted MEC network.

II. System Model and Problem Formulation

A. System Overview

As illustrated in Fig. 1, we consider an IoT network comprising M cells. Each cell is covered by a base station (BS) integrated with an ES, and all ESs are centrally controlled by an edge service provider (ESP) through a controller. For convenience, we use the same notation $\mathcal{M} = \{1, 2, ..., M\}$ to represent the sets of cells, BSs, and ESs. Moreover, several IoT devices are distributed in each cell. Similar to previous studies [4], [5], we categorize IoT devices into two types: task IoT devices (TDs) that generate tasks and can enhance their utility through computation offloading, and auxiliary IoT devices (ADs) with idle resources that can assist the ESs in their respective cells to process tasks. Assuming that in the mth cell, there are N_m TDs denoted as $\mathcal{N}_m = \{n_1^m, n_2^m, ..., n_{N_m}^m\}$ and K_m ADs denoted as $\mathcal{K}_m = \{k_1^m, k_2^m, ..., k_{K_m}^m\}$. Consequently, the sets of all TDs and ADs in the network can be denoted as $\boldsymbol{\mathcal{N}} = \{\mathcal{N}_1, ..., \mathcal{N}_M\}$ and $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, ..., \mathcal{K}_M\}$, respectively. Additionally, TDs, ADs and the ESP are typical interest-driven entities with conflicts of interest. TDs obtain edge computing services from the ESP and pay computation fees, while ADs assist the ESP in processing tasks and receive rewards from the ESP. The ESP earns revenue from providing services to TDs and can also profit by outsourcing tasks to ADs, acting as a middleman to capture a margin.

We assume that different cells reuse the same wireless resources, and IoT devices associated with the same BS utilize orthogonal spectrums for data transmission. Additionally, we assume that each IoT device within a cell occupies one sub-band, and the sub-band allocation among IoT devices is randomized. Consistent with [35], we consider a quasi-static network scenario, where each IoT device's channel remains unchanged until computation offloading is completed. Table II lists the main notations used in this paper.

B. Task Model

The task of TD n_i^m, $\forall n_i^m \in \boldsymbol{\mathcal{N}}$ is denoted as $H_{n_i^m} = \{L_{n_i^m}, C_{n_i^m}, t_{n_i^m}^{max}\}$. Here, $L_{n_i^m}$ denotes the size of $H_{n_i^m}$, $C_{n_i^m}$ denotes the number of CPU cycles required to complete $H_{n_i^m}$, and $t_{n_i^m}^{max}$ denotes the maximum tolerable delay for $H_{n_i^m}$. The parameters $\{L_{n_i^m}, C_{n_i^m}, t_{n_i^m}^{max}\}$ can be obtained from the task profiler [41]. We assume these tasks are atomic, and $x_{n_i^m}$ denotes the offloading decision of TD n_i^m, where $x_{n_i^m} = 1$ indicates that TD n_i^m offloads its task, and $x_{n_i^m} = 0$ indicates that TD n_i^m processes its task locally.

C. Execution Model

To fully utilize the ubiquitously distributed computational resources, collaboration can be applied among ESs in different cells (horizontal collaboration) and between the ES and ADs within each cell (vertical collaboration). We define the ES of each cell as the primary ES for all TDs in that cell, i.e., ES m is the primary ES for TD n_i^m, $\forall n_i^m \in \mathcal{N}_m$. Additionally, we denote ES m', $\forall m' \in \mathcal{M} \backslash \{m\}$ as the neighboring ES for TD n_i^m, $\forall n_i^m \in \mathcal{N}_m$. Based on the previous definitions, TD n_i^m has four execution modes for task processing: 1) local execution, 2) primary ES execution, 3) neighboring ES execution,

TABLE II
SUMMARY OF SYSTEM NOTATIONS

Notation	Description
$\mathcal{M}$	The set of BSs/cells/ESs
$\mathcal{N}$	The set of task IoT devices
$\mathcal{K}$	The set of auxiliary IoT devices
n_i^m	The ith TD in the mth cell
k_j^m	The jth AD in the mth cell
m	The mth BS/cell/ES
R_i^j	Data transfer rate between sender i and receiver j
$H_{n_i^m}$	The task generated by TD n_i^m
$L_{n_i^m}$	The size of $H_{n_i^m}$
$C_{n_i^m}$	CPU cycles required to process $H_{n_i^m}$
$t_{n_i^m}^{max}$	Maximum delay tolerated for task $H_{n_i^m}$
$V_{n_i^m}$	Value to TD n_i^m when $H_{n_i^m}$ is successfully processed
$x_{n_i^m}$	Offloading decision for TD n_i^m
$y_{n_i^m}$	Execution mode of $H_{n_i^m}$
$z_{n_i^m}$	Processing position of $H_{n_i^m}$
$u_{n_i^m}$	Utility of TD n_i^m
u_{esp}	Utility of the ESP
$u_{k_j^m}$	Utility of AD k_j^m
γ	Cost per unit of energy consumption
F_i	Total available resources for computing node i
$f_j^{n_i^m}$	Computational resources that node j needs to allocate for task $H_{n_i^m}$
$t_{a,b}^{n_i^m,tran}$	The transmission delay of task $H_{n_i^m}$ from computing node a to computing node b.
$cost_{a,b}^{n_i^m,tran}$	The cost of transferring task $H_{n_i^m}$ from computing node a to computing node b.
$cost_a^{n_i^m,comp}$	The cost of processing task $H_{n_i^m}$ on computing node a.
$u_m^{n_i^m}$	The gain that ES m can obtain from TD n_i^m.
$u_{esp}^{n_i^m,a}$	The utility gained by the ESP when assigning task $H_{n_i^m}$ to compute node a for processing.
$\alpha_{n_i^m}$	The fee that TD n_i^m needs to pay to the ESP
$\alpha_{k_j^m}$	The fee that the ESP needs to pay to AD k_j^m

and 4) AD execution. We define $y_{n_i^m}$ as the execution mode of the task $H_{n_i^m}$, where $y_{n_i^m} \in \{loc, pe, ne, ad\}$ corresponds to the four execution modes described earlier, respectively. Combining with the previous definition of $x_{n_i^m}$, we can easily obtain $y_{n_i^m} = loc$ when $x_{n_i^m} = 0$, and $y_{n_i^m} \in \{pe, ne, ad\}$ when $x_{n_i^m} = 1$. The cost of relevant participants in the above four different execution modes is analyzed as follows:

1) Local execution mode: When TD n_i^m processes its task locally, dynamic voltage and frequency scaling techniques can be utilized to control the energy consumption [42]. Considering the maximum tolerable delay $t_{n_i^m}^{max}$ for task $H_{n_i^m}$, the CPU frequency $f_{n_i^m}^{n_i^m}$ used by TD n_i^m to process $H_{n_i^m}$ needs to satisfy $f_{n_i^m}^{n_i^m} \geq C_{n_i^m}/t_{n_i^m}^{max}$. According to [43], the energy consumption is proportional to the square of the computation frequency. We assume that under the incentive of individual rationality, all computing nodes process tasks at the lowest CPU frequency that satisfies the task demand. Therefore, the CPU frequency of TD n_i^m for local computation is $f_{n_i^m}^{n_i^m} = C_{n_i^m}/t_{n_i^m}^{max}$. At this point, the cost of TD n_i^m can be calculated as described in [43]:

$$cost_{n_i^m}^{loc} = \gamma k_{n_i^m} {f_{n_i^m}^{n_i^m}}^2 C_{n_i^m}, \tag{1}$$

where $k_{n_i^m}$ denotes the energy consumption coefficient of TD n_i^m determined by the chip architecture, and γ denotes the cost of unit energy consumption.

2) Primary ES execution mode: In the primary ES execution mode, task $H_{n_i^m}$ will be processed on ES m, as shown in Fig. 2(a). The delay of task $H_{n_i^m}$ transferring to ES m can be calculated as

$$t_{n_i^m,m}^{n_i^m,tran} = \frac{L_{n_i^m}}{R_{n_i^m,m}}, \tag{2}$$

where $R_{n_i^m,m} = Wlog_2(1 + \frac{p_{n_i^m}|h_{n_i^m}|^2}{I_{n_i^m}+N_0})$ denotes the uplink transmission rate of TD n_i^m, W is the channel bandwidth, $p_{n_i^m}$ denotes the transmit power of TD n_i^m, $h_{n_i^m}$ denotes the channel gain between TD n_i^m and ES m, $I_{n_i^m}$ denotes the inter-cell interference to TD n_i^m, and N_0 is the background noise power. Notably, because BS m and ES m are directly connected via optical fiber, the transmission delay between them is negligible and thus ignored. Considering the maximum tolerated delay of $H_{n_i^m}$, the execution time of this task on ES m cannot exceed $t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran}$, where the delay of the result return is ignored [44]. Therefore, as in the previous derivation process for the local execution mode, the computational resources allocated by ES m for the task $H_{n_i^m}$ are $f_m^{n_i^m} = \frac{C_{n_i^m}}{t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran}}$. The cost of TD n_i^m in this mode is calculated as

$$cost_{n_i^m}^{off} = \gamma p_{n_i^m} t_{n_i^m,m}^{n_i^m,tran} + \alpha_{n_i^m}, \tag{3}$$

where $\alpha_{n_i^m}$ is the fee that TD n_i^m needs to pay to the ESP. The cost of ES m is calculated as

$$cost_m^{n_i^m,comp} = \gamma k_m {f_m^{n_i^m}}^2 C_{n_i^m}. \tag{4}$$

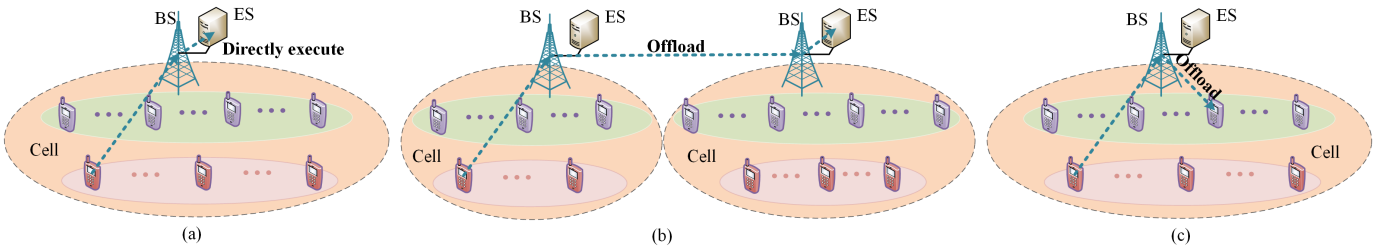


Fig. 2. Execution modes. (a) Primary ES execution. (b) Neighboring ES execution. (c) AD execution.

3) Neighboring ES execution mode: As shown in Fig. 2(b), task $H_{n_i^m}$ can be processed on ES m', $\forall m' \in \mathcal{M}\backslash\{m\}$ via forwarding from ES m. The transmission delay of task $H_{n_i^m}$ from ES m to ES m' is calculated as

$$t_{m,m'}^{n_i^m,tran} = \frac{L_{n_i^m}}{R_{m,m'}}, \tag{5}$$

where $R_{m,m'}$ denotes the transmission rate from ES m to ES m'. Similar to the previous analysis, the execution time of $H_{n_i^m}$ on ES m' is $t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran} - t_{m,m'}^{n_i^m,tran}$, which represents the maximum tolerable delay minus the transmission delay of the task offloading from TD n_i^m to ES m, and then forwarding it to ES m'. Therefore, the computational resources allocated by ES m' for $H_{n_i^m}$ are $f_{m'}^{n_i^m} = \frac{C_{n_i^m}}{t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran} - t_{m,m'}^{n_i^m,tran}}$.

According to Fig. 2(b), TD n_i^m still offloads its task to ES m. Therefore, the transmission cost of TD n_i^m in this mode is the same as in the primary ES execution mode. Meanwhile, the fee charged by the ESP to TD n_i^m remains unchanged, regardless of the processing location of its offloaded task. This is because the fee is determined during the initial bargaining game between TD n_i^m and ES m, and it does not vary based on the final task scheduling decisions. As a result, TDs only need to offload tasks without concerning the actual processing locations. This ensures fairness among TDs, as they are all charged the same price for the same quality of service, preventing any potential bias that could arise from changes in task processing locations. The transmission cost for ES m is calculated as

$$cost_{m,m'}^{n_i^m,tran} = \gamma p_m \frac{L_{n_i^m}}{R_{m,m'}}, \tag{6}$$

where p_m represents the transmission power of ES m. The computational cost for ES m' is calculated as

$$cost_{m'}^{n_i^m,comp} = \gamma k_{m'} {f_{m'}^{n_i^m}}^2 C_{n_i^m}. \tag{7}$$

4) AD execution mode: Each ES can delegate some tasks to ADs within its cell and provide rewards to the ADs that assist in handling the tasks, as shown in Fig. 2(c). Notably, consistent with [4], [5], [33], we assume that ADs can only assist the ES with tasks offloaded by TDs from the same cell. If task $H_{n_i^m}$ is delegated to AD k_j^m for processing, the delay of transferring task $H_{n_i^m}$ from ES m to AD k_j^m is calculated as

$$t_{m,k_j^m}^{n_i^m,tran} = \frac{L_{n_i^m}}{R_{m,k_j^m}}, \tag{8}$$

where $R_{m,k_j^m} = Wlog_2(1 + \frac{p_m |h_{k_j^m}|^2}{I_{k^m} + N_0})$ denotes the transmission rate from ES m to AD k_j^m, $h_{k_j^m}$ denotes the channel gain between AD k_j^m and ES m, and $I_{k_j^m}$ denotes the inter-cell interference to AD k_j^m.

Consistent with the previous analysis, the execution time of task $H_{n_i^m}$ on AD k_j^m is $t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran} - t_{m,k_j^m}^{n_i^m,tran}$. The computational resources allocated by AD k_j^m for task $H_{n_i^m}$ are $f_{k_j^m}^{n_i^m} = \frac{C_{n_i^m}}{t_{n_i^m}^{max} - t_{n_i^m,m}^{n_i^m,tran} - t_{m,k_j^m}^{n_i^m,tran}}$. In this mode, the cost of TD n_i^m remains the same, and the cost of ES m is calculated as

$$cost_{m,k_j^m}^{n_i^m,tran} = \gamma p_m \frac{L_{n_i^m}}{R_{m,k_j^m}} + \alpha_{k_j^m}, \tag{9}$$

where $\alpha_{k_j^m}$ denotes the reward that ES m pays for AD k_j^m. The cost of AD k_j^m is calculated as

$$cost_{k_j^m}^{n_i^m,comp} = \gamma k_{k_j^m} {f_{k_j^m}^{n_i^m}}^2 C_{n_i^m}. \tag{10}$$

D. Utility Model

The considered network scenario involves various participants, including TDs, ADs, and the ESP. Next, the utility of each participant is analyzed to facilitate effective collaboration among the three parties.

1) Utility of TDs: Based on the cost analysis of TDs in the previous subsection, the utility of TD n_i^m can be expressed as

$$u_{n_i^m} = \mathbb{I}(x_{n_i^m} = 0) u_{n_i^m}^{loc} + \mathbb{I}(x_{n_i^m} = 1) u_{n_i^m}^{off}, \tag{11}$$

where $\mathbb{I}(\cdot)$ is the indicator function and equals 1 (resp., 0) if the condition is true (resp., false). $u_{n_i^m}^{loc} = \mathbb{I}(F_{n_i^m} \geq f_{n_i^m}^{n_i^m})(V_{n_i^m} - cost_{n_i^m}^{loc})$ denotes the utility of TD n_i^m to handle its task locally, where $F_{n_i^m}$ denotes the available computational resources of TD n_i^m, and $V_{n_i^m}$ denotes the utility that TD n_i^m can obtain when the task $H_{n_i^m}$ is successfully processed.

2) Utility of ADs: We use an auction mechanism to incentivize ADs to assist ESs in processing tasks, assuming that the lowest acceptable price reported by AD k_j^m for bidding is $a_{k_j^m}$ \$/cycle. When ES m delegates tasks requiring $C_{k_j^m}$ CPU cycles to AD k_j^m and the ESP pays $\alpha_{k_j^m}$ to AD k_j^m, the utility of AD k_j^m is calculated as

$$u_{k_j^m} = \alpha_{k_j^m} - a_{k_j^m} C_{k_j^m}. \tag{12}$$

3) Utility of the ESP: The utility of the ESP is the sum of the utility of all ESs. For tasks processed locally, the ESP can't obtain any utility. For the offloaded task $H_{n_i^m}$, there are three cases to discuss separately.

First, to facilitate the subsequent description, we use $z_{n_i^m}$ to denote the processing location of task $H_{n_i^m}$. Combining the previous definitions of $x_{n_i^m}$ and $y_{n_i^m}$ with the analysis of the four task execution modes, we define the following:

- $z_{n_i^m} = loc$ means $H_{n_i^m}$ is processed locally, *i.e.*, $x_{n_i^m} = 0$ and $y_{n_i^m} = loc$;
- $z_{n_i^m} = m$ means $H_{n_i^m}$ is processed on ES m, *i.e.*, $x_{n_i^m} = 1$ and $y_{n_i^m} = pe$;
- $z_{n_i^m} = m', \forall m' \in \mathcal{M}\backslash\{m\}$ means $H_{n_i^m}$ is processed on ES m' , *i.e.*, $x_{n_i^m} = 1$ and $y_{n_i^m} = ne$;
- $z_{n_i^m} = k_j^m, \forall k_j^m \in \mathcal{K}_m$ means $H_{n_i^m}$ is processed on AD k_j^m, *i.e.*, $x_{n_i^m} = 1$ and $y_{n_i^m} = ad$;

According to the above definitions and the cost analysis of ESs in Section II-C, if $z_{n_i^m} = m$, the utility obtained by the ESP is $u_{esp}^{n_i^m,m} = \alpha_{n_i^m} - cost_m^{n_i^m,comp}$. If $z_{n_i^m} = m', \forall m' \in \mathcal{M}\backslash\{m\}$, the utility obtained by the ESP is $u_{esp}^{n_i^m,m'} = \alpha_{n_i^m} - cost_{m,m'}^{n_i^m,tran} - cost_{m'}^{n_i^m,comp}$. If $z_{n_i^m} = k_j^m, \forall k_j^m \in \mathcal{K}_m$, the utility obtained by the ESP is $u_{esp}^{n_i^m,k_j^m} = \alpha_{n_i^m} - cost_{m,k_j^m}^{n_i^m,tran} -$

$\beta_{n_i^m}^{k_j^m}$, where $\beta_{n_i^m}^{k_j^m}$ denotes the fee that the ESP pays to AD k_j^m for processing task $H_{n_i^m}$.

Consequently, the utility of the ESP can be calculated as

$$\begin{aligned} u_{esp} = \sum_{m\in\mathcal{M}}\sum_{n_i^m\in\mathcal{N}_m}\mathbb{I}(x_{n_i^m}=1)\Big\{ & \\ & \mathbb{I}(y_{n_i^m}=pe)u_{esp}^{n_i^m,m}+ \\ & \mathbb{I}(y_{n_i^m}=ne)\Big\{\sum_{m'\in\mathcal{M}\backslash\{m\}}\mathbb{I}(z_{n_i^m}=m')u_{esp}^{n_i^m,m'}\Big\}+ \\ & \mathbb{I}(y_{n_i^m}=ad)\Big\{\sum_{k_j^m\in\mathcal{K}_m}\mathbb{I}(z_{n_i^m}=k_j^m)u_{esp}^{n_i^m,k_j^m}\Big\} \\ \Big\}. & \end{aligned} \tag{13}$$

E. Problem Formulation

ESs, ADs, and TDs form a multilayer edge structure in IoT networks. Our goal is to optimize task allocation and service pricing jointly, improve the utilization of end-device resources, and incentivize efficient collaboration among participants in the network. Therefore, considering the available resources, task demands, and utility of different participants, the optimization problem is formulated to maximize the system utility through optimizing task allocation and service pricing, which is given by

$$\begin{aligned} \mathcal{P}_1: \max_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\boldsymbol{A},\boldsymbol{P}} \; & u_{esp}+\sum_{m\in\mathcal{M}}\{\sum_{n_i^m\in\mathcal{N}_m}u_{n_i^m}+\sum_{k_j^m\in\mathcal{K}_m}u_{k_j^m}\} \\ \text{s.t. } & (a): x_{n_i^m}\in\{0,1\},\forall m\in\mathcal{M},\forall n_i^m\in\mathcal{N}_m; \\ & (b): y_{n_i^m}\in\{loc,pe,ne,ad\},\forall m\in\mathcal{M},\forall n_i^m\in\mathcal{N}_m; \\ & (c): z_{n_i^m}=y_{n_i^m}=loc, if\ x_{n_i^m}=0; \\ & (d): z_{n_i^m}=m, if\ x_{n_i^m}=1\ and\ y_{n_i^m}=pe; \\ & (e): z_{n_i^m}\in\mathcal{M}\backslash\{m\}, if\ x_{n_i^m}=1\ and\ y_{n_i^m}=ne; \\ & (f): z_{n_i^m}\in\mathcal{K}_m, if\ x_{n_i^m}=1\ and\ y_{n_i^m}=ad; \\ & (g): \alpha_{n_i^m}\geq 0,\forall m\in\mathcal{M},\forall n_i^m\in\mathcal{N}_m; \\ & (h): p_{k_j^m}\geq a_{k_j^m},\forall m\in\mathcal{M},\forall k_j^m\in\mathcal{K}_m; \\ & (i): \sum_{m\in\mathcal{M}}\sum_{n_i^m\in\mathcal{N}_m}\mathbb{I}(z_{n_i^m}=q)f_q^{n_i^m}\leq F_q,\forall q\in\mathcal{M}\cup\mathcal{K}; \\ & (j): u_{esp}\geq 0; \\ & (k): u_{n_i^m}\geq 0,\forall m\in\mathcal{M},\forall n_i^m\in\mathcal{N}_m; \\ & (l): u_{k_j^m}\geq 0,\forall m\in\mathcal{M},\forall k_j^m\in\mathcal{K}_m. \end{aligned} \tag{14}$$

The first term of the objective function is the ESP's utility, and the second term is the sum of the utility for all TDs and ADs. $\boldsymbol{X}=\{x_{n_i^m}|m\in\mathcal{M},n_i^m\in\mathcal{N}_m\}$, $\boldsymbol{Y}=\{y_{n_i^m}|m\in\mathcal{M},n_i^m\in\mathcal{N}_m\}$, and $\boldsymbol{Z}=\{z_{n_i^m}|m\in\mathcal{M},n_i^m\in\mathcal{N}_m\}$ denote the offloading decisions, execution modes, and processing locations of all tasks, respectively. $\boldsymbol{A}=\{\alpha_{n_i^m}|m\in\mathcal{M},n_i^m\in\mathcal{N}_m\}$ represents the task processing fees that all TDs need to pay to the ESP. $\boldsymbol{P}=\{p_{k_j^m}|m\in\mathcal{M},k_j^m\in\mathcal{K}_m\}$ represents the reward per CPU cycle that the ESP pays to ADs for assisting in task processing.

Constraints $(14a)-(14f)$ denote the range for values of the variables related to task allocation. Constraint $(14g)$ ensures that the computation fee paid by each TD to the ESP cannot be negative, and constraint $(14h)$ ensures that the reward per CPU cycle paid by the ESP to any AD assisting in task processing cannot be lower than its minimum acceptable price. Constraint $(14i)$ implies that the total amount of computing resources allocated by each computing node cannot exceed its available resources. Constraints $(14j)-(14l)$ ensure that the utility of the ESP, TDs, and ADs is not negative, i.e., the individual rationality constraint must be satisfied.

Theorem 1: The system utility maximization problem $\mathcal{P}_1$ is NP-hard.

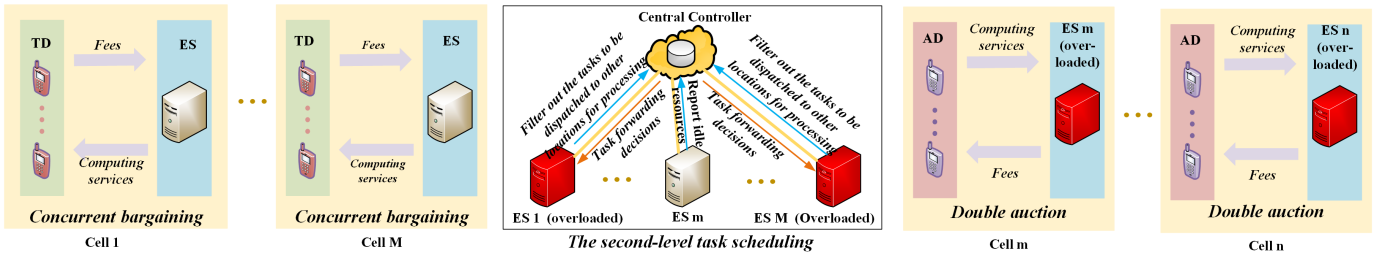
Proof: We consider a specific case of the system utility maximization problem. First, we simplify the network model to a single cell. Then, we assume that the values of $\boldsymbol{X},\boldsymbol{A},\boldsymbol{P}$ are given. At this point, the remaining optimization variables are the processing locations of all offloaded tasks. According to the analysis in Section II-C, each offloaded task brings different system utility and requires different computational resources when processed on different compute nodes. We can treat each compute node as a backpack with different capacities. Each offloaded task can be treated as an item with different values and weights when placed in different backpacks. Therefore, the problem is transformed into maximizing the total value of the items placed in all backpacks, while ensuring the sum of the weights of the items in each backpack does not exceed its capacity limit. This is known as the multi-knapsack problem, a well-known NP-hard problem [45]. Since this specific case of the problem can be mapped to a multi-knapsack problem, it can be inferred that problem $\mathcal{P}_1$ is also NP-hard. □

According to the above analysis, the problem remains NP-hard even after several simplifications. Therefore, finding a near-optimal solution to the original problem with low computational complexity becomes a key challenge. In the following sections, we will present a multi-level task scheduling framework that is scalable and can be applied to large-scale multi-cell IoT networks.

III. Algorithm design

A. Overview of the Multi-Level Decision-Making Framework

This subsection first provides an overall overview of the designed framework. As shown in Fig. 3, the proposed task scheduling framework is divided into three levels. The first level of decision-making uses a bargaining game to determine the initial offloading decision and the computational fee to be paid to the ESP for each TD, as shown in Fig. 3(a). Notably, the first level of decision-making is executed in parallel in each cell. After the first level of decision-making, assuming all the offloaded tasks are using the primary ES execution mode, we can determine whether each ES is overloaded. The overloaded ESs need to screen out tasks that will be dispatched to other locations for processing. As shown in Fig. 3(b), in the second level of decision-making, the overloaded ESs report the screened tasks to the central controller, while the underloaded ESs report their available resources. Subsequently, the central controller decides on the tasks with the



(a) First-level decision-making (b) Second-level decision-making (c) Third-level decision-making

Fig. 3. Logic diagram of the multi-level decision-making framework.

neighboring ES execution mode and their specific processing locations. Therefore, if no ES is overloaded after the first level of decision-making, all offloaded tasks will use the primary ES computation mode, and the decision-making process ends. If all ESs are overloaded, the second level of decision-making will be skipped, and the third level of decision-making will be executed directly. As shown in Fig. 3(c), in the third level of decision-making, the ESs that are still overloaded will assign certain tasks to ADs in their cells through a double auction. Finally, tasks that are still not assigned a processing location will be processed locally. Notably, the third level of decision-making is executed in parallel in cells with overloaded ESs. Additionally, as described in Section II-C, the fees paid to the ESP by TDs that offload tasks will not change after the first level of decision-making; only the processing location may change.

The following subsections will describe the decision-making process at each level in detail.

B. First Level of Decision-Making

As shown in Fig. 3(a), in the first level of decision-making, a parallel bargaining game is employed to determine the initial offloading decision and payment fee for each TD. The first level of decision-making is executed in parallel in each cell, with all TDs and the ES in a cell bargaining simultaneously. This approach not only accelerates the algorithm but also ensures fairness [46]. The parallel game mechanism guarantees fairness among TDs by preventing the outcome of one bargaining round from influencing the next. Moreover, the bargaining game, grounded in its theoretical foundation and the symmetry between both parties, ensures fairness between the TD and ES [47], [48].

In a bargaining game, two players are bargaining with each other, either reaching an agreement point in a set $\mathcal{A}$ or reaching a disagreement point $\mathcal{D}$. For the bargaining game between ES m and TD n_i^m, the set $\mathcal{A}$ corresponds to the set of feasible computational fees that TD n_i^m pays to ES m for offloading. The disagreement point $\mathcal{D}$ corresponds to the situation where TD n_i^m decides to process the task locally, implying that the bargaining process breaks down. The Nash bargaining solution assigns an outcome to the bargaining game, which can be either an agreement point or a disagreement point. This outcome maximizes the product of the parties' surpluses relative to the disagreement point [30].

To further reduce the information exchange time introduced by the bargaining game, we solve the Nash bargaining solution for each bargaining game, treating it as the bargaining outcome for both parties. At this point, neither party will have an incentive to change the outcome. Next, we solve the Nash bargaining solution for TD n_i^m and ES m. Since the utility of TD n_i^m and ES m at the disagreement point is $u_{n_i^m}^{loc}$ and 0, respectively, the first level of decision-making problem can be formulated as

$$\begin{aligned} \mathcal{P}_2: \max_{x_{n_i^m},\alpha_{n_i^m}} \ & (u_{n_i^m} - u_{n_i^m}^{loc}) u_m^{n_i^m} \\ \text{s.t. } & (a): u_{n_i^m} \geq 0, \\ & (b): u_m^{n_i^m} \geq 0, \\ & (c): x_{n_i^m} \in \{0,1\}, \\ & (d): \alpha_{n_i^m} \geq 0, \end{aligned} \tag{15}$$

where $u_m^{n_i^m} = \mathbb{I}(x_{n_i^m} = 1) u_{esp}^{n_i^m,m}$ denotes the gain that ES m can obtain from TD n_i^m. Conditions $(15a)$ and $(15b)$ imply that both TD n_i^m and ES m are rational individuals. Conditions $(15c)$ and $(15d)$ specify the range of values for the optimization variables.

In conjunction with Section II-D, the problem $\mathcal{P}_2$ can be reformulated as

$$\begin{aligned} \mathcal{P}_3: \max_{x_{n_i^m},\alpha_{n_i^m}} \ & ((1 - x_{n_i^m}) u_{n_i^m}^{loc} + x_{n_i^m} u_{n_i^m}^{off} - u_{n_i^m}^{loc}) x_{n_i^m} u_{esp}^{n_i^m,m} \\ \text{s.t. } & (a): (15a) - (15d). \end{aligned} \tag{16}$$

When $x_{n_i^m} = 0$, it can be deduced that $\alpha_{n_i^m} = 0$, $u_{n_i^m} \geq 0$, $u_m^{n_i^m} = 0$, and the value of the objective function is zero. When $x_{n_i^m} = 1$, the problem $\mathcal{P}_3$ transforms to

$$\begin{aligned} \mathcal{P}_4: \max_{\alpha_{n_i^m}} \ & (u_{n_i^m}^{off} - u_{n_i^m}^{loc}) u_{esp}^{n_i^m,m} \\ \text{s.t. } & (a): u_{n_i^m}^{off} \geq 0, \\ & (b): u_{esp}^{n_i^m,m} \geq 0, \\ & (c): (15d). \end{aligned} \tag{17}$$

Therefore, $x_{n_i^m} = 1$ is a Nash solution if and only if $u_{n_i^m}^{off} - u_{n_i^m}^{loc} > 0$ and $u_{esp}^{n_i^m,m} > 0$. Next, we need to determine the optimal value of $\alpha_{n_i^m}$ when $x_{n_i^m} = 1$ is a Nash solution. First, we expand the previous two inequalities to obtain

$$V_{n_i^m} - \gamma p_{n_i^m} t_{n_i^m,m}^{n_i^m,tran} - \alpha_{n_i^m} - u_{n_i^m}^{loc} > 0, \tag{18}$$

and

$$\alpha_{n_i^m} - \gamma k_m {f_m^{n_i^m}}^2 C_{n_i^m} > 0. \tag{19}$$

Then, we can determine the range of values for $\alpha_{n_i^m}$ when $x_{n_i^m} = 1$ is a Nash solution:

$$\alpha_{n_i^m}^{min} < \alpha_{n_i^m} < \alpha_{n_i^m}^{max}, \tag{20}$$

where $\alpha_{n_i^m}^{min} = \gamma k_m {f_m^{n_i^m}}^2 C_{n_i^m}$ and $\alpha_{n_i^m}^{max} = V_{n_i^m} - \gamma p_{n_i^m} t_{n_i^m,m}^{n_i^m,tran} - u_{n_i^m}^{loc}$.

Therefore, $x_{n_i^m}^1 = 0$ and $\alpha_{n_i^m}^1 = 0$ constitute the Nash bargaining solution between TD n_i^m and ES m when $\alpha_{n_i^m}^{min} \geq \alpha_{n_i^m}^{max}$. Additionally, the values of the optimization variables associated with the task $H_{n_i^m}$ are determined as $x_{n_i^m}^* = 0, y_{n_i^m}^* = loc, z_{n_i^m}^* = loc, \alpha_{n_i^m}^* = 0$. When $\alpha_{n_i^m}^{min} < \alpha_{n_i^m}^{max}$, $x_{n_i^m}^1 = 1$ is a Nash solution. At this point, we can expand the objective function of $\mathcal{P}_4$ as follows:

$$-\alpha_{n_i^m}^2 + B\alpha_{n_i^m} + C, \tag{21}$$

where $B = V_{n_i^m} - \gamma p_{n_i^m} t_{n_i^m,m}^{n_i^m,tran} - u_{n_i^m}^{loc} + \gamma k_m {f_m^{n_i^m}}^2 C_{n_i^m}$ and $C = -V_{n_i^m}\gamma k_m {f_m^{n_i^m}}^2 C_{n_i^m} + \gamma^2 p_{n_i^m} t_{n_i^m,m}^{n_i^m,tran} k_m {f_m^{n_i^m}}^2 C_{n_i^m} + \gamma u_{n_i^m}^{loc} k_m {f_m^{n_i^m}}^2 C_{n_i^m}$. Therefore, equation (21) is a typical univariate quadratic concave function, meaning it has a unique maximum point. Combined with the range of values for $\alpha_{n_i^m}$, we can ascertain the optimal solution for $\alpha_{n_i^m}$ as follows:

$$\alpha_{n_i^m}^1 = \begin{cases} \frac{1}{2}B & \text{if } \alpha_{n_i^m}^{min} < \frac{1}{2}B < \alpha_{n_i^m}^{max}, \\ \alpha_{n_i^m}^{min} & \text{if } \alpha_{n_i^m}^{min} \geq \frac{1}{2}B, \\ \alpha_{n_i^m}^{max} & \text{if } \alpha_{n_i^m}^{max} \leq \frac{1}{2}B. \end{cases} \tag{22}$$

At the first level of decision-making, we obtain the initial offloading decision and payment fee for each TD. As described in Section III-A, we also need to determine whether each ES is overloaded. For the overloaded ESs, we need to filter out the tasks to be dispatched to other locations for processing.

To determine whether each ES is overloaded, we assume that all tasks decided to be offloaded in the first level of decision-making will use the primary ES execution mode. Then, the amount of computational resources that ES $m, \forall m \in \mathcal{M}$ needs to provide can be expressed as

$$f_m^1 = \sum_{n_i^m \in \mathcal{N}_m} x_{n_i^m}^* f_m^{n_i^m}. \tag{23}$$

If $f_m^1 > F_m$, it indicates that ES m is overloaded. Otherwise, ES m is underloaded. For the underloaded ESs, all offloaded tasks in their cells will employ the primary ES execution mode. Therefore, the optimization variables for these tasks are determined as $x_{n_i^m}^* = 1, y_{n_i^m}^* = pe, z_{n_i^m}^* = m, \alpha_{n_i^m}^* = \alpha_{n_i^m}^1, \forall n_i^m \in \{n_i^m | f_m^1 \geq F_m, x_{n_i^m} = 1\}$.

For the overloaded ES m, we need to filter out tasks to be dispatched to other locations for processing. For this purpose, we design a priority-based task filtering algorithm, where the priority function is defined as

$$O_{n_i^m} = \frac{u_m^{n_i^m}}{f_m^{n_i^m}}. \tag{24}$$

A larger value of the priority function indicates that ES m gains more utility per unit of computational resource by processing the task locally. Hence, the task is given higher priority to execute on ES m. The designed task filtering algorithm is presented in Algorithm 1.

In Algorithm 1, $\mathcal{I}_m$ denotes the set of indexes for tasks to be offloaded in the first level of decision-making. F_m denotes the total amount of computational resources for ES m. $\mathcal{I}_m^{loc}$ denotes the set of indexes for tasks to be processed at ES m, and $\mathcal{I}_m^{off}$ denotes the set of indexes for tasks to be dispatched to other locations for processing. Furthermore, according to the description in Sec III-A, Algorithm 1 is executed in parallel across all overloaded ESs. All tasks in $\mathcal{I}_m^{loc}$ will use the primary ES execution mode, with the optimization variables for these tasks determined as $x_{n_i^m}^* = 1, y_{n_i^m}^* = pe, z_{n_i^m}^* = m, \alpha_{n_i^m}^* = \alpha_{n_i^m}^1, \forall i \in \mathcal{I}_m^{loc}$.

Algorithm 1: Priority-Based Task Filtering Algorithm (For overloaded ES $m \in \mathcal{M}$)

input : $\mathcal{I}_m$ and F_m.
output: $\mathcal{I}_m^{loc}$ and $\mathcal{I}_m^{off}$.

Initialize $f_m^{left} = F_m, \mathcal{I}_m^{loc} = \phi, \mathcal{I}_m^{off} = \phi$;
Calculate the priority function value of all tasks in $\mathcal{I}_m$ according to (24). Then, rank the task indexes in descending order based on these values to obtain $\mathcal{I}_m'$;
for i *in* $\mathcal{I}_m'$ **do**
  if $f_m^{n_i^m} \leq f_m^{left}$ **then**
    $f_m^{left} = f_m^{left} - f_m^{n_i^m}$;
    $\mathcal{I}_m^{loc} = \mathcal{I}_m^{loc} \cup \{i\}$;
  else
    $\mathcal{I}_m^{off} = \mathcal{I}_m^{off} \cup \{i\}$;
return $\mathcal{I}_m^{loc}$, $\mathcal{I}_m^{off}$

C. Second Level of Decision-Making

As described in Section III-A, the second level of decision-making identifies the tasks that will use the neighboring ES execution mode and their specific processing locations. At this level, we categorize all ESs into two sets: $\mathcal{M}^{adequate}$ (the set of underloaded ESs) and $\mathcal{M}^{inadequate}$ (the set of overloaded ESs) based on the first level of decision-making. Next, we implement a two-tier prioritization-based decision-making approach. Specifically, in the outer layer, we prioritize the ESs in the set $\mathcal{M}^{inadequate}$. Subsequently, the inner layer decisions are made sequentially based on these established priorities. The priority function for ES $m_2 \in \mathcal{M}^{inadequate}$ is defined as

$$Q_{m_2} = f_{m_2}^1 - F_{m_2}. \tag{25}$$

The larger the value of Q_{m_2}, the more severely ES m_2 is overloaded, resulting in a higher priority.

In the inner layer decision for ES m_2, we use a greedy algorithm to determine if each task in $I_{m_2}^{off}$ should adopt the neighboring ES execution mode. When a task is determined

to employ the neighboring ES execution mode, its specific processing location also needs to be determined. The tasks that are not assigned processing locations in the second level of decision-making will be addressed in the third level. The process for the second level of decision-making is detailed in Algorithm 2.

Algorithm 2: Priority-Based Task Scheduling Algorithm Among ESs

input : $f_{m_1}^{left}, \forall m_1 \in \mathcal{M}^{adequate}$, and $\mathcal{I}_{m_2}^{off}, \forall m_2 \in \mathcal{M}^{inadequate}$.
output: $\mathcal{I}_{m_2}^{ne}, \forall m_2 \in \mathcal{M}^{inadequate}$, and $z_{n_i^{m_2}}, \forall i \in \mathcal{I}_{m_2}^{ne}, \forall m_2 \in \mathcal{M}^{inadequate}$.

Initialize $\mathcal{I}_{m_2}^{ne} = \phi, \forall m_2 \in \mathcal{M}^{inadequate}$;
Calculate the priority of the ESs in $M^{inadequate}$ according to (25), then sort them in descending order to obtain $\mathcal{M}^{inadequate'}$;
for m_2' *in* $\mathcal{M}^{inadequate'}$ **do**
  Sort $m_1 \in \mathcal{M}^{adequate}$ in descending order of R_{m_2',m_1} to yield $\mathcal{M}^{adequate\prime}$;
  for i *in* $\mathcal{I}_{m_2'}^{off}$ **do**
    for m_1' *in* $\mathcal{M}^{adequate'}$ **do**
      if $f_{m_1'}^{n_i^{m_2'}} \leq f_{m_1'}^{left}$ *and* $u_{m_1'}^{n_i^{m_2'}} > 0$ **then**
        $\mathcal{I}_{m_2'}^{ne} = \mathcal{I}_{m_2'}^{ne} \cup \{n_i^{m_2'}\}$, $z_{n_i^{m_2'}} = m_1'$, $f_{m_1'}^{left} = f_{m_1'}^{left} - f_{m_1'}^{n_i^{m_2'}}$;
        break;
return $\mathcal{I}_{m_2}^{ne}, \forall m_2 \in \mathcal{M}^{inadequate}$, and $z_{n_i^{m_2}}, \forall i \in \mathcal{I}_{m_2}^{ne}, \forall m_2 \in \mathcal{M}^{inadequate}$

In Algorithm 2, $f_{m_1}^{left}$ represents the amount of idle resources for ES $m_1 \in \mathcal{M}^{adequate}$ after the first level of decision-making. $I_{m_2}^{ne}$ denotes the set of tasks that are decided to use the neighboring ES execution mode after the second level of decision-making. $z_{n_i^{m_2}}$ represents the specific processing location of task $H_{n_i^{m_2}}$, where $i \in \mathcal{I}_{m_2}^{ne}$ and $m_2 \in \mathcal{M}^{inadequate}$. After the second level of decision-making, all optimization variables for tasks employing the neighboring ES execution mode are determined as $x_{n_i^{m_2}}^* = 1, y_{n_i^{m_2}}^* = ne, z_{n_i^{m_2}}^* = z_{n_i^{m_2}}, \alpha_{n_i^{m_2}}^* = \alpha_{n_i^{m_2}}^1, \forall i \in \mathcal{I}_{m_2}^{off}, \forall m_2 \in \mathcal{M}^{inadequate}$.

D. Third Level of Decision-Making

If there are tasks that remain without a specific processing location after the second level of decision-making, or if all ESs are overloaded after the first level of decision-making, the corresponding cells will initiate the third level of decision-making. In this case, the third level of decision-making is executed in parallel across all relevant cells. The ESs in these cells will delegate certain tasks without processing locations to the ADs for processing, providing a certain reward, as shown in Fig. 3(c). The interaction between the ES and ADs in each cell can be modeled as a double auction. In the cell triggering the third level of decision-making, ADs act as sellers, the ES as the buyer, and the BS as the third-party auctioneer. For clarity, we will use cell m_3 as an example to discuss the specific process of the third-level decision.

At the beginning of the double auction, AD $k_j^{m_3}, \forall k_j^{m_3} \in \mathcal{K}_{m_3}$ reports its available computational resources $F_{k_j^{m_3}}$ to BS m_3 and bids with the lowest acceptable price $a_{k_j^{m_3}}$ per CPU cycle. Assume the set of task indexes delegated by ES m_3 in the third level of decision-making is $\mathcal{I}_{m_3}^{off}$. ES m_3 also reports to BS m_3 the payment fee for each task in $\mathcal{I}_{m_3}^{off}$ determined in the first level of decision-making. Considering the transmission overhead when ES m_3 forwards task $H_{n_i^{m_3}}$ to AD $k_j^{m_3}$, BS m_3 can calculate the maximum acceptable payment fee per CPU cycle for ES m_3 to delegate task $H_{n_i^{m_3}}$ to AD $k_j^{m_3}$, which is formulated as

$$b_{n_i^{m_3}}^{k_j^{m_3}} = \frac{\alpha_{n_i^{m_3}}^1 - cost_{m_3,k_j^{m_3}}^{n_i^{m_3},tran}}{C_{n_i^{m_3}}}. \tag{26}$$

Subsequently, BS m_3 needs to determine the set of tasks using the AD execution mode, their specific processing locations, and the reward ES m_3 needs to pay to each AD.

According to [49] and [50], the desired properties of the double auction design are:

- *Individual rationality:* All auction matching outcomes do not compromise participant interests. If task $H_{n_i^{m_3}}$ is delegated to AD $k_j^{m_3}$ in the third level of decision-making and ES m_3 pays a fee of $\beta_{n_i^{m_3}}^{k_j^{m_3}}$ to AD $k_j^{m_3}$ for this task, it should satisfy $b_{n_i^{m_3}}^{k_j^{m_3}} \geq \beta_{n_i^{m_3}}^{k_j^{m_3}}/C_{n_i^{m_3}}$ and $\beta_{n_i^{m_3}}^{k_j^{m_3}}/C_{n_i^{m_3}} \geq a_{k_j^{m_3}}$.
- *Honesty:* Neither buyers nor sellers have an incentive to alter their bids or offers. All traders submit bids or offers based on their true valuation of the resource. Untruthful offers or bids do not generate additional revenue.
- *Budget Balancing:* After the transaction, the total expenses of all buyers balance with the total income of all sellers, ensuring the auctioneer does not lose money.
- *System Efficiency:* This involves maximizing the number of successful transactions at the end of the auction.

To achieve these goals, we design a double auction matching mechanism based on bids, offers, and resource constraints, presented in Algorithm 3.

In Algorithm 3, $\mathcal{I}_{m_3}^{off}$ represents the set of task indexes in cell m_3 participating in the third level of decision-making, and $\mathcal{G}_{m_3}$ denotes the set of matched pairs of tasks and ADs in cell m_3. Similar to the analysis in [49], the designed algorithm satisfies the economic principles of individual rationality, honesty, and budget balancing, which are not expanded here. The third level of decision-making identifies tasks that employ the AD execution mode and their specific execution locations. Additionally, the reward paid by the ESP to AD $k_j^{m_3}$ is calculated as

$$\alpha_{k_j^{m_3}} = \sum_{i \in \mathcal{I}_{m_3}^{off}} \beta_{n_i^{m_3}}^{k_j^{m_3}}. \tag{27}$$

After the third level of decision-making, the optimization variables for tasks determined to use the AD execution mode

Algorithm 3: Double Auction-Based Matching and Pricing Algorithm (Use cell m_3 as an example)

input : $\mathcal{I}_{m_3}^{off}$, $b_{n_i^{m_3}}^{k_j^{m_3}}$, $a_{k_j^{m_3}}$, and $F_{k_j^{m_3}}$, $\forall i \in \mathcal{I}_{m_3}^{off}$, $\forall k_j^{m_3} \in \mathcal{K}_{m_3}$.

output: $\mathcal{G}_{m_3}$, and $\beta_{n_i^{m_3}}^{k_j^{m_3}}$, $\forall i \in \mathcal{I}_{m_3}^{off}$, $\forall k_j^{m_3} \in \mathcal{K}_{m_3}$.

1 Initialize $\mathcal{G}_{m_3} = \phi$, $f_{k_j^{m_3}}^{left} = F_{k_j^{m_3}}$, $\beta_{n_i^{m_3}}^{k_j^{m_3}} = 0$, $\forall i \in \mathcal{I}_{m_3}^{off}$, $\forall k_j^{m_3} \in \mathcal{K}_{m_3}$;
2 Arrange the ADs in ascending order of $a_{k_j^{m_3}}$, yielding $\mathcal{K}'_{m_3}$;
3 **for** i *in* $\mathcal{I}_{m_3}^{off}$ **do**
4 **for** $j = 0, 1, ..., |\mathcal{K}'_{m_3}| - 2$ **do**
5 **if** $b_{n_i^{m_3}}^{\mathcal{K}'_{m_3}[j]} > a_{\mathcal{K}'_{m_3}[j+1]}$ *and* $f_{\mathcal{K}'_{m_3}[j]}^{n_i^{m_3}} \leq f_{\mathcal{K}'_{m_3}[j]}^{left}$ **then**
6 $\mathcal{G}_{m_3} = \mathcal{G}_{m_3} \cup \{(n_i^{m_3}, \mathcal{K}'_{m_3}[j])\}$, $f_{\mathcal{K}'_{m_3}[j]}^{left} = f_{\mathcal{K}'_{m_3}[j]}^{left} - f_{\mathcal{K}'_{m_3}[j]}^{n_i^{m_3}}$, $\beta_{n_i^{m_3}}^{\mathcal{K}'_{m_3}[j]} = a_{\mathcal{K}'_{m_3}[j+1]} * C_{n_i^{m_3}}$;
7 break;
8 **return** $\mathcal{G}_{m_3}$, $\beta_{n_i^{m_3}}^{k_j^{m_3}}$, $\forall i \in \mathcal{I}_{m_3}^{off}$, $\forall k_j^{m_3} \in \mathcal{K}_{m_3}$

are identified. Finally, tasks with undetermined processing locations are designated to employ the local execution mode. At this point, the values of all optimization variables are specified.

E. Multi-Level Decision-Making Framework

In summary, the first level of decision-making identifies tasks using the primary ES execution mode, the second level identifies tasks using the neighboring ES execution mode, the third level identifies tasks using the AD execution mode, and all remaining tasks use the local execution mode. The overall framework is presented in Algorithm 4.

Notably, despite the complexity of the proposed multi-level decision-making framework, only the relevant components need to be implemented on the BSs, ESs and central controller during practical deployment. The components deployed on the BSs handle the first and third-level decisions. The components deployed on the ESs need to provide necessary information for the second-level decision to the central controller component.

At the first level of decision-making, the BS in each cell collects information from all TDs and the ES within the same cell, then calculates the Nash bargaining solutions using the formulas provided in SectionIII-B (line 2). BSs then send these solutions to the corresponding ESs. Afterward, each ES evaluates its load based on these Nash negotiation solutions with all TDs, determines the tasks employing the primary ES execution mode, and sends details of tasks ready for processing elsewhere or information about idle resources to the central controller (lines 3-7).

Based on reports from all ESs, the central controller then decides whether to conclude the decision-making process, skip the second level of decision-making, or proceed with the second level of decision-making (lines 9-13).

If the central controller opts for the second level of decision-making, it determines the tasks employing the neighboring ES mode and their specific processing locations, using task details and idle resources provided by each ES. These decisions are then communicated to the respective ESs (line 14).

Algorithm 4: Multi-Level Decision-Making Algorithm

1 `// The process in lines 2-7 is executed in parallel across all cells. The following process is exemplified by cell` m
2 A parallel bargaining game is played between TD $n_i^m, \forall n_i^m \in \mathcal{N}_m$ and ES m. The initial offloading decision $x_{n_i^m}$ and the payment fee $\alpha_{n_i^m}$ of TD n_i^m are determined according to the formulas in Section III-B;
3 Calculate f_m^1 according to (23);
4 **if** $f_m^1 > F_m$ **then**
5 Filter the tasks to be dispatched to other locations for processing according to Algorithm 1, and identify the tasks employing the primary ES execution mode;
6 **else**
7 All tasks determined to be offloaded in the first level of decision-making employ the primary ES execution mode and calculate the remaining idle resources of ES m;
8 `// The process in lines 9-16 is performed by the central controller`
9 **if** *all ESs are underloaded* **then**
10 Algorithm ends;
11 **if** *all ESs are overloaded* **then**
12 **goto** line 19;
13 **else**
14 Execute Algorithm 2 to determine the tasks employing the neighboring ES mode and their specific processing locations;
15 **if** *All tasks are assigned processing locations* **then**
16 Algorithm ends;
17 **else**
18 `// Line 19 is executed in parallel for cells where tasks with unallocated processing locations exist. The following process is exemplified by cell` m
19 In cell m, execute Algorithm 3 to determine (1) the tasks employing the AD execution mode, (2) their specific processing locations, and (3) the fees the ESP needs to pay to the ADs;
20 Tasks without assigned processing locations are determined to employ the local execution mode.

Next, the central controller checks if processing locations have been assigned for all tasks. If not, it instructs the ESs with tasks that have unallocated processing locations to start the third level of decision-making, while notifying the others to skip this step. If all tasks have been assigned, the process ends, and all ESs are informed to skip the third level (lines 15-17).

Each ES involved in the third-level of decision-making determines the tasks employing the AD execution mode and their specific processing locations through a double auction organized by the corresponding BS, acting as the auctioneer. Then, tasks without assigned processing locations are determined to employ the local execution mode (lines 19-20). Finally, all ESs transmit task processing decisions to the respective TDs in their cells. Relevant fees will be collected from the respective TDs and paid to the corresponding ADs upon completion of task processing, based on the decision results.

IV. Theoretical Analysis

In this section, we theoretically analyze the performance of the proposed scheme. First, we analyze the complexity of the proposed scheme. Next, we discuss the equivalence between the decomposed subproblems and the original problem, followed by an analysis of the scheme's optimality.

A. Computational Complexity

The proposed framework employs hierarchical decision-making and parallel execution to reduce computational complexity. The first level of decision-making is executed in parallel for individual cells. Its computational complexity is $\mathcal{O}(1)$ since the Nash bargaining solution is obtained directly via formula calculation. For task screening of overloaded ESs, Algorithm 1 has a computational complexity of $\mathcal{O}(NlogN)$, where $N = max(N_1, \ldots, N_M)$. Algorithm 2 has a computational complexity of $\mathcal{O}(MlogM + MNlogN)$. Algorithm 3 is executed in parallel for certain cells; its computational complexity is $\mathcal{O}(NKlogK)$, where $K = max(K_1, \ldots, K_M)$. Combining the above analyses, the overall computational complexity of the proposed framework is $\mathcal{O}(MlogM + MNlogN + NKlogK)$.

B. Equivalence Analysis

The system utility can be expressed as the sum of the utilities obtained by TDs, ADs, and the ESP for each task. As stated in Section II-C, the fee paid to the ESP when a TD offloads its task is independent of the processing location. Therefore, when calculating the utility for each TD, we need to consider only two scenarios: offloading the task or processing it locally. As mentioned in Section II-D, the ESP and ADs receive varying utility for each offloaded task depending on the execution mode. Therefore, the utility of the ESP and ADs needs to be calculated separately for each execution mode. Based on (11) and (13), the objective function of Problem $\mathcal{P}_1$ can be reformulated as follows:

$$\begin{aligned}
&\sum_{m\in\mathcal{M}}\sum_{n_i^m\in\mathcal{N}_m}\mathbb{I}(x_{n_i^m}=1)\Bigg\{\\
&\quad\mathbb{I}(y_{n_i^m}=pe)u_{esp}^{n_i^m,m}+\\
&\quad\mathbb{I}(y_{n_i^m}=ne)\Bigg\{\sum_{m'\in\mathcal{M}\setminus\{m\}}\mathbb{I}(z_{n_i^m}=m')u_{esp}^{n_i^m,m'}\Bigg\}+\\
&\quad\mathbb{I}(y_{n_i^m}=ad)\Bigg\{\sum_{k_j^m\in\mathcal{K}_m}\mathbb{I}(z_{n_i^m}=k_j^m)(u_{esp}^{n_i^m,k_j^m}+u_{k_j^m}^{n_i^m})\Bigg\}+\\
&\quad u_{n_i^m}^{off}\\
&\Bigg\}+\sum_{m\in\mathcal{M}}\sum_{n_i^m\in\mathcal{N}_m}\mathbb{I}(x_{n_i^m}=0)u_{n_i^m}^{loc},
\end{aligned}\tag{28}$$

where $u_{k_j^m}^{n_i^m}$ represents the utility that AD k_j^m gains by assisting ES m in processing task $H_{n_i^m}$. In our proposed multi-level decision-making framework (Section III-E), the first level of decision-making identifies tasks using the primary ES execution mode, their processing locations, and the fee for offloading each task. The second level determines tasks that use the neighboring ES execution mode and their processing locations. The third level identifies tasks processed using the AD execution mode, their processing locations, and the reward per CPU cycle paid by the ESP to ADs for assistance. Lastly, tasks processed locally are determined. Therefore, the subproblems solved at each level correspond to subterms in (28), and the final solution includes the optimization variables of the original problem. Moreover, the solutions of all subproblems meet the constraints of problem $\mathcal{P}_1$. Based on the above analysis, the decomposed subproblems are jointly equivalent to the original problem.

C. Optimality Analysis

Based on the previous subsection's analysis, the decomposed subproblems in our multi-level decision-making framework are jointly equivalent to the original problem. Thus, to analyze optimality, we focus on each level of decision-making individually. In the first level, a parallel bargaining game is employed to determine the initial offloading decision and payment fee for each TD. Decisions are made by solving the Nash bargaining solution, whose optimality is supported by four core Nash axioms: pareto efficiency, symmetry, invariance to affine transformations, and independence of irrelevant alternatives. More details can be found in [51]. In the second level, a priority-based scheduling algorithm is applied, where priority is determined by the utility per unit of resource. This is essentially a greedy algorithm. Although the greedy algorithm may not achieve a global optimum, it provides an approximate optimal solution with strong performance guarantees. The third level employs a double auction mechanism. Similar to the analysis in [49], we can show that the algorithm is budget-balanced, individually profitable, system-efficient, and truthful. From the above analyses, it is evident that the proposed framework can achieve an approximate optimal solution to

the original problem with strong performance guarantees. We prioritize efficiency in practice, accepting some deviation from the global optimum as it remains within acceptable limits.

TABLE III
MAIN SIMULATION PARAMETERS

Parameters	Value
M	5
$N_m, \forall m \in \mathcal{M}$	[200, 800]
$K_m, \forall m \in \mathcal{M}$	[20, 40]
N_0	-100 dBm
$k_{n_i^m}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	5×10^{-27}
W	20 MHz
$F_{n_i^m}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	[0.05, 1] GHz
$F_m, \forall m \in \mathcal{M}$	10 GHz
$L_{n_i^m}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	[200, 500] Kbit
$C_{n_i^m}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	[50, 500] Mega Cycles
$t_{n_i^m}^{max}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	[0.05, 2] s
γ	1 \$
$V_{n_i^m}, \forall n_i^m \in \mathcal{N}_m, \forall m \in \mathcal{M}$	[5, 10] \$

V. PERFORMANCE EVALUATION

A. Simulation Setting

In the simulation, we consider the scenario depicted in Fig. 1. Similar to [43], the transmit power of each IoT device is set to 2 W, the background noise power is -100 dBm, and the energy efficiency coefficient is 5×10^{-27}. According to [35], the unit energy cost is set to 1 \$ and the task value ranges from 5 \$ to 10 \$. The main simulation parameters are summarized in Table III. Unless otherwise specified, parameter values are set according to Table III. All simulations were executed on a Python platform with an Intel Core i7-13650HX 4.9 GHz CPU and 16 GB of RAM.

We compare the performance of the proposed framework with the following benchmark schemes.

1) *Collaborative Edge-End Computing Strategy* [52]: Auxiliary IoT devices serve as collaborators of edge servers for task processing but do not involve multiple ESs collaborating with each other.
2) *Collaborative Edge Computing Strategy* [53]: ESs collaborate to process tasks without assistance from ADs.
3) *Double Auction-Based Strategy* [49]: All ESs and ADs act as sellers, TDs act as buyers, and a trusted third-party platform acts as the auctioneer to organize the double auction.
4) *Conventional Edge Computing Strategy*: ESs execute the arrived tasks directly without collaboration among ESs or assistance from ADs.

Notably, the double auction-based strategy involves new parameters (bids and offers of ESs and TDs) and is not comparable with the other schemes in terms of the objective function. Therefore, this scheme is only used to compare computational complexity, as shown in Fig. 5.

In addition to comparing different frameworks, we also compared the algorithms used at each level of the proposed framework with the following baseline algorithms.

1) *Stackelberg Game*: A Stackelberg game is used in the first level of decision-making to determine the initial offloading decision and the payment fee for each TD.
2) *Round-Robin Scheduling*: Round-robin scheduling is employed in the second level of decision-making to allocate tasks among ESs.
3) *Vickrey Auction*: A Vickrey auction is employed in the third level of decision-making to incentivize ADs to assist ESs in processing tasks.

Notably, we modify only the algorithm of one level for comparison, while the algorithms in the other levels remain consistent with those in our proposed framework.

B. Simulation Results

1) System Utility: The system utility is defined as the average utility of all participants in each cell. Fig. 4(a) shows the average system utility under various edge computing strategies with different numbers of cells. At $M = 1$, the system utility is the same for both the proposed scheme and the collaborative edge-end computing strategy, as well as for the collaborative edge computing strategy and the conventional edge computing strategy, because there is only one cell. As the number of cells increases, the system utility of the collaborative edge-end computing strategy and the conventional edge computing strategy

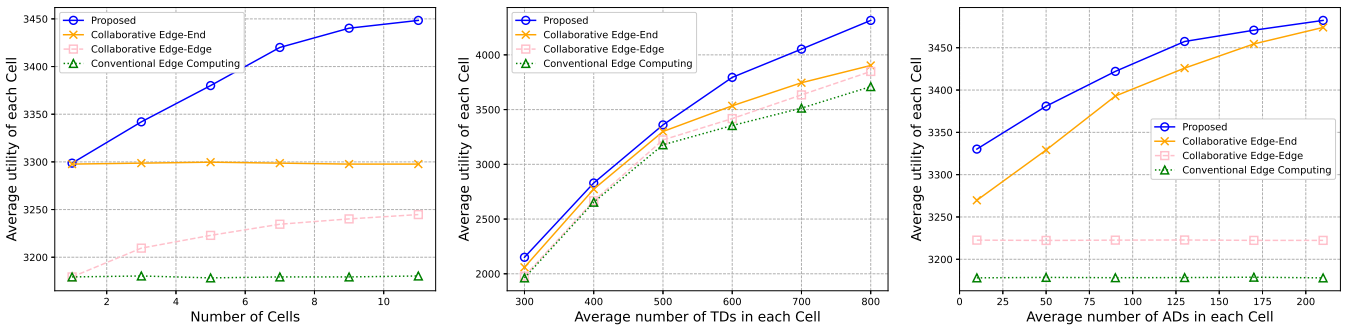


Fig. 4. System utility of different frameworks. (a) Comparison of system utility with different numbers of cells. (b) Comparison of system utility with different numbers of TDs. (c) Comparison of system utility with different numbers of ADs.

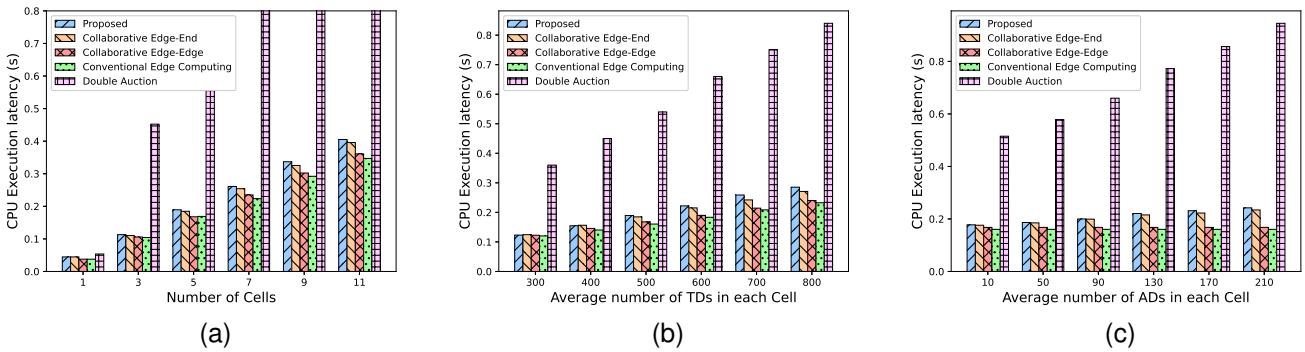


Fig. 5. Average algorithm running time of different frameworks. (a) Comparison of algorithm running time with different numbers of cells. (b) Comparison of algorithm running time with different numbers of TDs. (c) Comparison of algorithm running time with different numbers of ADs.

remain nearly constant, while that of the proposed scheme and the collaborative edge computing strategy gradually increases. This is because, as the number of cells increases, the utility gain from inter-cell collaboration becomes more prominent.

Fig. 4(b) shows the variation in average system utility for different edge computing strategies with the mean number of TDs in each cell. In this experiment, $N_m, \forall m \in \mathcal{M}$ is uniformly distributed within a range with an interval length of 600. As the number of TDs increases, the system utility for all strategies increases because TDs can gain utility from processing their tasks locally. As the number of TDs increases, the number of tasks generated also increases, thus increasing the utility of processing these tasks. The proposed scheme achieves the highest system utility because it allows more TDs to offload their tasks. According to Section III-B, TDs offload their tasks if and only if the utility gained from offloading is greater than from processing locally. Therefore, more TDs offloading their tasks results in higher system utility.

Fig. 4(c) shows the variation in average system utility for different edge computing strategies with the mean number of ADs in each cell. Similarly, $K_m, \forall m \in \mathcal{M}$ is uniformly distributed within a range, with an interval length of 20. As the number of ADs increases, the system utility for the collaborative edge computing strategy and the conventional edge computing strategy remains nearly constant, while the system utility for the proposed scheme and the collaborative edge-end computing strategy gradually increases. This is because, as the number of ADs increases, the utility gain from ADs becomes more prominent, while the collaborative edge computing strategy and the conventional edge computing strategy do not utilize ADs.

2) Algorithm Running Time: Algorithm running time is defined as the average time taken for algorithm execution. Notably, in the simulation, we execute sub-algorithms serially that would be executed in parallel in a real scenario, so the algorithm's running time should be shorter in practice. Additionally, the parameter settings are consistent with the previous experiments. Fig. 5(a) shows the average algorithm running times for different strategies versus the number of cells, indicating that the running times of all strategies increase with the number of cells. In real scenarios, due to parallel execution, the algorithm running time of the proposed strategy does not significantly increase with the number of cells. Fig. 5(b) shows the variation in algorithm running times for different strategies with the mean number of TDs in each cell. As the mean number of TDs in each cell increases, the algorithm running times for all strategies grow. Fig. 5(c) shows the variation in algorithm running times for different strategies with the mean number of ADs in each cell. As the number of ADs grows, the algorithm running times for the collaborative edge computing strategy and the conventional edge computing strategy remain nearly constant, while those of the proposed scheme and the collaborative edge-end computing strategy slowly increase.

From the three figures, it can be seen that the proposed strategy significantly reduces the algorithm execution latency compared to the double auction-based algorithm, mainly due to its hierarchical decision-making approach. Additionally, the algorithm running time of the proposed strategy is not sensitive to the number of TDs, ADs, or cells, making it suitable for online execution in large-scale IoT networks. Notably, the algorithm running time of the proposed scheme is longer than that of the collaborative edge-end strategy, collaborative edge computing strategy, and conventional edge computing strategy. This is because the latter three strategies only consider a subset of participants from the proposed scheme, effectively skipping one or two levels of decision-making, which leads to shorter running time. As illustrated in Fig. 4, the proposed scheme achieves significant performance gains at the expense of only a slight increase in algorithm running time.

3) Average Task Offloading Ratio: The task offloading ratio is defined as the ratio of the number of tasks eventually offloaded to the number of tasks decided to be offloaded in the first level of decision-making. Fig. 6 shows the variation in task offloading ratio with different parameters for various strategies, with the experimental settings remaining the same. Fig. 6(a) shows the task offloading ratio versus the number of cells for different strategies. At $M = 1$, the task offloading ratio is the same for both the proposed scheme and the collaborative edge-end computing strategy, as well as for the collaborative edge computing strategy and the conventional edge computing strategy. As the number of cells increases,

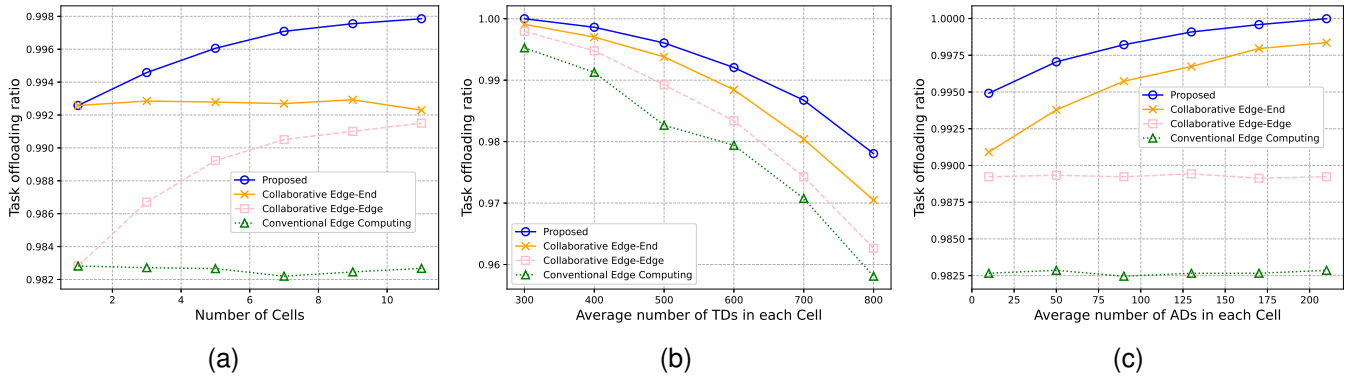


Fig. 6. Average task offloading ratio of different frameworks. (a) Comparison of task offloading ratio with different numbers of cells. (b) Comparison of task offloading ratio with different numbers of TDs. (c) Comparison of task offloading ratio with different numbers of ADs.

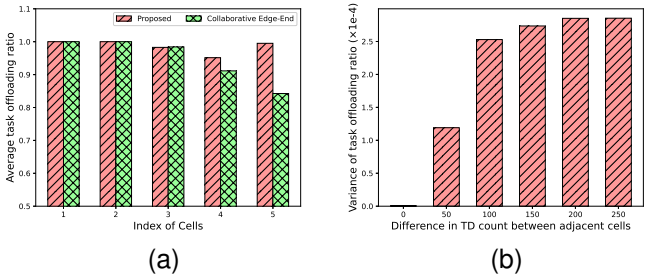


Fig. 7. (a) The task offloading ratio in each cell with different frameworks. (b) The variance of task offloading ratio with different levels of task distribution non-uniformity for the proposed framework.

the task offloading ratio for the conventional edge computing strategy and the collaborative edge-end computing strategy remains nearly constant, while that for the proposed scheme and the collaborative edge computing strategy gradually increases. This is because inter-cell collaboration allows the edge computing system to accommodate more offloaded tasks as the number of cells increases.

Fig. 6(b) shows the task offloading ratio versus the mean number of TDs in each cell for different strategies. It can be seen that the task offloading ratio decreases with the increasing number of TDs for all strategies, with the proposed strategy maintaining the highest ratio. Fig. 6(c) shows the task offloading ratio versus the mean number of ADs in each cell for different strategies. As the number of ADs increases, the task offloading ratio for the collaborative edge computing strategy and the conventional edge computing strategy remains nearly constant, while that for the proposed scheme and the collaborative edge-end computing strategy gradually increases. This is because, as the number of ADs increases, ADs can accommodate more offloaded tasks.

4) Handle Non-Uniformly Distributed Tasks: To evaluate the performance of the proposed framework with non-uniformly distributed tasks, we first compare it with the collaborative edge-end computing strategy, using the default values for the number of cells and the distribution of ADs from Table III. Fig. 7(a) shows the task offloading ratio in each cell, with the number of TDs in the five cells set to 200, 350, 500, 650, and 800, respectively. It can be seen that the task offloading ratio does not differ much among cells under our proposed scheme. However, in the collaborative edge-end computing strategy, the task offloading ratio significantly decreases in the cells with more TDs. Fig. 7(b) shows the performance of the proposed scheme under different levels of task distribution non-uniformity. The horizontal axis represents the difference in the number of TDs between adjacent cells, with an average of 500 TDs across the five cells. It can be observed that although the variance of the task offloading ratio gradually increases as task distribution becomes more uneven, the variance remains at a relatively low level overall. In summary, the proposed scheme demonstrates good load-balancing capability.

5) Effect of Different Parameters on the Framework's Performance: Fig. 8 illustrates the effect of various system parameters on the utility gain achieved by the proposed scheme. Here, utility gain refers to the increase in system utility relative to the scenario where all TDs adopt local execution mode. Fig. 8(a) and Fig. 8(b) depict how the system utility gain changes with an increasing number of cells, considering different average numbers of TDs and ADs per cell. As the number of cells increases, the utility gain from inter-cell collaboration grows, although the rate of growth progressively slows. Additionally, as the average number of TDs ($\overline{N_m}$) and ADs ($\overline{K_m}$) per cell increases, the system utility also rises, but the growth rate diminishes for different reasons. With an increase in the number of TDs, the available computing resources in the device-assisted edge computing network become inadequate to accommodate all TDs. When the number of ADs increases, the limited task processing demands of TDs cause more ADs to remain idle. Fig. 8(c) shows how the system utility gain changes with varying unit energy costs as the number of cells increases. As γ increases, the system utility gain decreases gradually, reaching zero when $\gamma \geq 4$. This occurs because as γ increases, based on the expressions for $\alpha_{n_i^m}^{min}$ and $\alpha_{n_i^m}^{max}$ in (20), $\alpha_{n_i^m}^{min}$ rises while $\alpha_{n_i^m}^{max}$ declines. When $\alpha_{n_i^m}^{min} \geq \alpha_{n_i^m}^{max}$, TD n_i^m will execute the task locally, resulting in no utility gain for that task.

6) Algorithm Comparison: Fig. 9 compares the algorithms used in the proposed framework with the baseline algorithms,

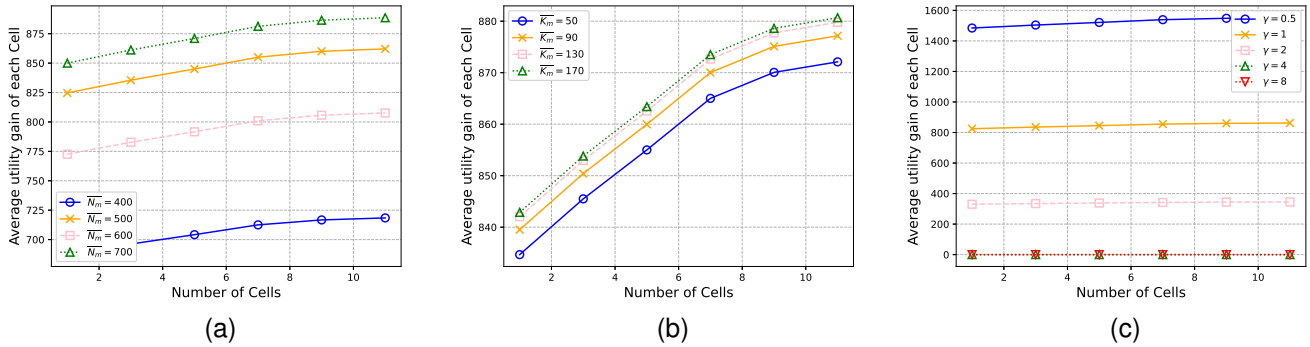


Fig. 8. Impact of various system parameters on the utility gain achieved by the proposed scheme. (a) Comparison of utility gain with different average numbers of TDs per cell. (b) Comparison of utility gain with different average numbers of ADs per cell. (c) Comparison of utility gain with different unit energy costs.

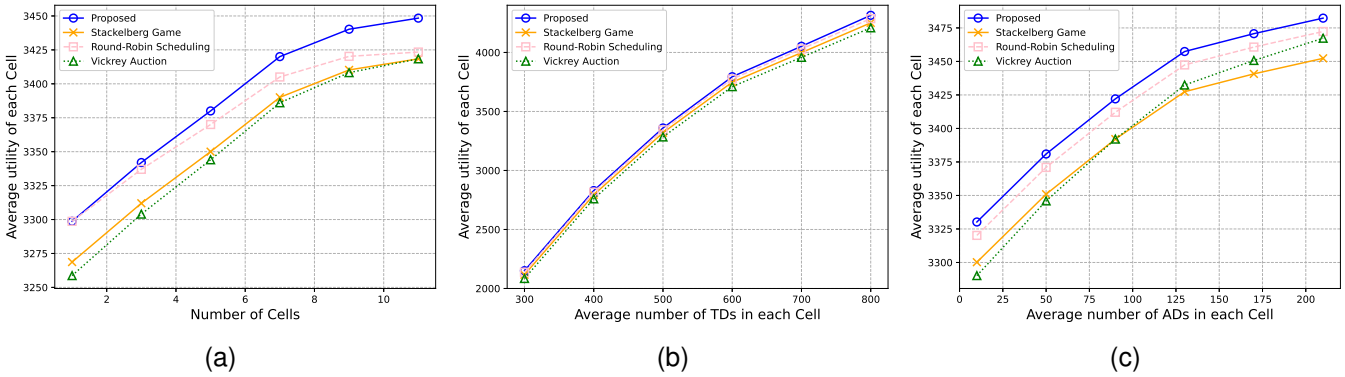


Fig. 9. System utility of different algorithms. (a) Comparison of system utility with different numbers of cells. (b) Comparison of system utility with different numbers of TDs. (c) Comparison of system utility with different numbers of ADs.

using an experimental setup consistent with Fig. 4. The three sub-figures demonstrate that the algorithms used in the proposed framework are more aligned with the defined objective function, achieving higher system utility. This occurs because the Stackelberg game maximizes the utility of the leader (ESP) while neglecting the utility of the followers (TDs). Round-robin scheduling ensures fair resource allocation but does not aim to maximize utility. The Vickrey auction encourages buyers to bid honestly but ignores the lowest acceptable price for ESs' auction tasks, potentially reducing the utility of the ESP. Based on the experimental results and analyses, round-robin scheduling has the least effect on system utility. As shown in Fig. 9(a), the performance gap between round-robin scheduling and the proposed framework widens as the number of cells increases, while the gap between Vickrey auction and the proposed framework narrows. This trend is likely due to the increasing influence of the second decision-making level as the number of cells grows, which reduces the role of the third level in task-related decisions. Fig. 9(c) shows that as the average number of ADs per cell increases, the Vickrey auction curve surpasses the Stackelberg game, likely due to higher transaction prices for tasks in the third-level of decision-making as the average number of ADs per cell rises.

VI. Conclusion

In this paper, we consider a multi-cell device-assisted edge computing system and propose an incentive-driven multi-level task scheduling framework that enables horizontal collaboration among edge servers and vertical collaboration between edge servers and auxiliary IoT devices. At the first level of decision-making, a bargaining game is used to model the interaction between TDs and ESs, ensuring fair and Pareto-optimal outcomes. At the second level of decision-making, we design a priority-based task scheduling algorithm to allocate tasks among ESs. At the third level of decision-making, the double auction mechanism is employed to incentivize ADs to assist ESs in processing tasks. Simulation results show that our proposed algorithm outperforms the benchmark strategies in terms of system utility, task offloading ratio, load balancing capability, and algorithm execution time. Future work involves edge servers from different edge service providers. This includes collaboration among edge servers controlled by the same provider and developing a reasonable profit-sharing mechanism to incentivize collaboration among different providers.

References

[1] L. Zhang and J. Xu, "Differential security game in heterogeneous device-to-device offloading network under epidemic risks," *IEEE Trans.*

Network Sci. Eng., vol. 7, no. 3, pp. 1852–1861, Nov. 2019.

[2] Y. Li, B. Lei, Z. Li, Z. Qu, X. Zhang, and W. Wang, "Task offloading with multi-cluster collaboration for computing and network convergence," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–3.

[3] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166 079–166 108, Nov. 2019.

[4] Y. Li, X. Ge, B. Lei, X. Zhang, and W. Wang, "Joint task partitioning and parallel scheduling in device-assisted mobile edge networks," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14 058–14 075, Apr. 2024.

[5] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Mar. 2020.

[6] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-enabled mobile-edge computation offloading for multiuser IoT network," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12 490–12 504, Mar. 2021.

[7] X. Dai, Z. Xiao, H. Jiang, M. Alazab, J. C. Lui, S. Dustdar, and J. Liu, "Task co-offloading for D2D-assisted mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Inf.*, vol. 19, no. 1, pp. 480–490, Mar. 2022.

[8] Z. Xiao, J. Shu, H. Jiang, J. C. Lui, G. Min, J. Liu, and S. Dustdar, "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023.

[9] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.

[10] Z. Chen, W. Yi, A. S. Alam, and A. Nallanathan, "Dynamic task software caching-assisted computation offloading for multi-access edge computing," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6950–6965, Oct. 2022.

[11] T. N. Dang, K. Kim, L. U. Khan, S. A. Kazmi, Z. Han, and C. S. Hong, "On-device computational caching-enabled augmented reality for 5G and beyond: A contract-theory-based incentive mechanism," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17 382–17 394, Dec. 2021.

[12] Z. Xiao, J. Shu, H. Jiang, J. C. Lui, G. Min, J. Liu, and S. Dustdar, "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023.

[13] Y. Zhao, A. Xiao, S. Wu, C. Jiang, L. Kuang, and Y. Shi, "Adaptive partitioning and placement for two-layer collaborative caching in mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 8, pp. 8215–8231, Jan. 2024.

[14] H. Zhou, Z. Wang, G. Min, and H. Zhang, "UAV-aided computation offloading in mobile-edge computing networks: A Stackelberg game approach," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6622–6633, Apr. 2023.

[15] W. Fan, X. Li, B. Tang, Y. Su, and Y. Liu, "MEC network slicing: Stackelberg-game-based slice pricing and resource allocation with QoS guarantee," *IEEE Trans. Netw. Serv. Manage.*, vol. 21, no. 4, pp. 4494–4509, Jun. 2024.

[16] 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. 15 679–15 689, Sep. 2023.

[17] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 356–367, Apr. 2020.

[18] P. Wang, W. Ma, H. Zhang, W. Sun, and L. Xu, "A dynamic contribution measurement and incentive mechanism for energy-efficient federated learning in 6G," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Seoul, Korea, 2022, pp. 1–6.

[19] 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. 16 510–16 520, Sep. 2022.

[20] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 138–149, Feb. 2022.

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

[22] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint resource allocation and incentive design for blockchain-based mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 6050–6064, Sep. 2020.

[23] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, and M. Huang, "TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things," *IEEE Trans. Mob. Comput.*, vol. 21, no. 11, pp. 4125–4138, Nov. 2022.

[24] M. Chen, H. Wang, D. Han, and X. Chu, "Signaling-based incentive mechanism for D2D computation offloading," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4639–4649, Mar. 2022.

[25] C. Chen, S. Gong, W. Zhang, Y. Zheng, and Y. C. Kiat, "DRL-based contract incentive for wireless-powered and UAV-assisted backscattering MEC system," *IEEE Trans. Cloud Comput.*, vol. 12, no. 1, pp. 264–276, Jan. 2024.

[26] T. N. Dang, K. Kim, L. U. Khan, S. A. Kazmi, Z. Han, and C. S. Hong, "On-device computational caching-enabled augmented reality for 5G and beyond: A contract-theory-based incentive mechanism," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17 382–17 394, Dec. 2021.

[27] T. N. Dang, A. Manzoor, Y. K. Tun, S. A. Kazmi, Z. Han, and C. S. Hong, "A contract theory-based incentive mechanism for UAV-enabled VR-based services in 5G and beyond," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16 465–16 479, Sep. 2023.

[28] A. Adhikari and D. B. Rawat, "Leveraging smart contracts for priority based caching in D2D networks for 5G and beyond," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Madrid, Spain, 2021, pp. 1–6.

[29] X. Tan, S. Li, S. Wang, Y. Liu, Q. Zheng, and J. Yang, "Cooperative bargaining game based adaptive video multicast over mobile edge networks," *IEEE Trans. Multimedia*, vol. 26, pp. 2380–2394, 2024.

[30] G. Chen, Y. Chen, Z. Mai, C. Hao, M. Yang, and L. Du, "Incentive-based distributed resource allocation for task offloading and collaborative computing in MEC-enabled networks," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 9077–9091, May. 2023.

[31] Y.-Y. Shih, C.-Y. Wang, and A.-C. Pang, "Fog computing service provision using bargaining solutions," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 1765–1780, Dec. 2021.

[32] E. Meskar and B. Liang, "Fair multi-resource allocation in heterogeneous servers with an external resource type," *IEEE/ACM Trans. Networking*, vol. 31, no. 3, pp. 1244–1262, Jun. 2023.

[33] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[34] Y. Zhang, X. Hou, H. Du, L. Zhang, J. Du, and W. Men, "Joint trajectory and resource optimization for UAV and D2D-enabled heterogeneous edge computing networks," *IEEE Trans. Veh. Technol.*, Early Access. 2024.

[35] W. Hou, H. Wen, N. Zhang, J. Wu, W. Lei, and R. Zhao, "Incentive-driven task allocation for collaborative edge computing in Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 706–718, Jan. 2022.

[36] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[37] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Sep. 2021.

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

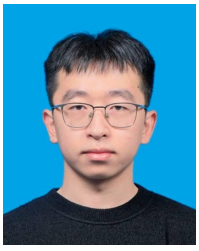
[39] C. Feng, P. Han, X. Zhang, Q. Zhang, Y. Zong, Y. Liu, and L. Guo, "Cost-minimized computation offloading of online multifunction services in collaborative edge-cloud networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 20, no. 1, pp. 292–304, Mar. 2023.

[40] W. Hua, P. Liu, and L. Huang, "Energy-efficient resource allocation for heterogeneous edge-cloud computing," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2808–2818, Jan. 2024.

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

[42] Z. Chen, W. Yi, A. S. Alam, and A. Nallanathan, "Dynamic task software caching-assisted computation offloading for multi-access edge computing," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6950–6965, Oct. 2022.

[43] Y. Li, X. Zhang, B. Lei, Q. Zhao, M. Wei, Z. Qu, and W. Wang, "Computation rate maximization for wireless powered edge computing with multi-user cooperation," *IEEE Open J. Commun. Soc.*, 2024.
[44] Y. Li, X. Zhang, Y. Sun, J. Liu, B. Lei, and W. Wang, "Joint offloading and resource allocation with partial information for multi-user edge computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Rio de Janeiro, Brazil, 2022, pp. 1736–1741.
[45] D. Pisinger, "Where are the hard knapsack problems?" *Comput. Oper. Res.*, vol. 32, no. 9, pp. 2271–2284, Sep. 2005.
[46] L. Gao, G. Iosifidis, J. Huang, L. Tassiulas, and D. Li, "Bargaining-based mobile data offloading," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1114–1125, Jun. 2014.
[47] M. M. Pillutla and J. K. Murnighan, "Fairness in bargaining," *Social Justice Research*, vol. 16, pp. 241–262, 2003.
[48] R. Forsythe, J. L. Horowitz, N. E. Savin, and M. Sefton, "Fairness in simple bargaining experiments," *Games and Economic behavior*, vol. 6, no. 3, pp. 347–369, 1994.
[49] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial internet of things," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4692–4701, Oct. 2018.
[50] Y. Yue, W. Sun, and J. Liu, "Multi-task cross-server double auction for resource allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–6.
[51] E. Van Damme, "The Nash bargaining solution is optimal," *Journal of Economic Theory*, vol. 38, no. 1, pp. 78–100, 1986.
[52] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.
[53] R. Lin, T. Xie, S. Luo, X. Zhang, Y. Xiao, B. Moran, and M. Zukerman, "Energy-efficient computation offloading in collaborative edge computing," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21 305–21 322, Nov. 2022.



Yang Li received the B.S. degree in communication engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2022. He is currently pursuing the Ph.D. degree with the Key Laboratory of Universal Wireless Communication, School of Information and Communication Engineering, BUPT. His research interests include device-assisted mobile edge networks, computing offloading and resource allocation.

Xing Zhang (M'10-SM'14) is currently a full professor with the School of Information and Communications Engineering, Beijing University of Posts and Telecommunications, China. His research interests are mainly in 5G/6G networks, satellite communications, edge intelligence, and Internet of Things. He has authored or coauthored five technical books and over 300 papers in top journals and international conferences and holds over 80 patents. He has received six Best Paper Awards in international conferences. He is a Senior Member of the IEEE and a member of CCF. He has served as a General Co-Chair of the third IEEE International Conference on Smart Data (SmartData-2017), as a TPC Co-Chair/TPC Member for a number of major international conferences.

Bo Lei is now the deputy director of the Network Technology Research Institute of a director of future network research center of China telecom research institute. Bo Lei received his Master's degree in Telecommunication Engineering from Beijing University of Posts and Telecommunications, Beijing, P. R. China, in 2006. His currently research interests include future network architecture, new network technology, computing power network and 5G application verification. Bo Lei now leads the future network research center focusing on future network. He is the first author of two technical books and has published more than 30 papers in top journals and international conferences, and filed more than 30 patents.

Qianying Zhao is now an engineer of future network research center of China telecom research institute. Qianying Zhao received her Master's degree in Sant'Anna School of Advanced Studies, Pisa, Italy, in 2018. Her currently research interests include computing power network, edge computing. She is the author of two technical books and has published 7 papers in journals.

Min Wei is now an engineer of future network research center of China telecom research institute. She received her Master's degree in Beijing University of Posts and Telecommunications, Beijing, P. R. China, in 2015. Her currently research interest is computing power network.

Zheyan Qu received his B.S. degree in information engineering at the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2022. He is currently pursuing the M.S. degree with the Key Laboratory of Universal Wireless Communication, School of Information and Communication Engineering, BUPT. His research interests include edge computing, natural language processing, and large language models.

Wenbo Wang received the B.S., M.S., and Ph.D. degrees from BUPT in 1986, 1989, and 1992, respectively. He is currently a Professor with the School of Information and Communications Engineering, and the Executive Vice Dean of the Graduate School, Beijing University of Posts and Telecommunications. He is currently the Assistant Director with the Key Laboratory of Universal Wireless Communication, Ministry of Education. He has authored over 200 journal and international conference papers, and six books. His current research interests include radio transmission technology, wireless network theory, and software radio technology.