Computing Resource Pricing for Satellites in the OEC System: A Bilevel Optimization Approach

Xinyu Wang, Qi Wang, Qingshan Wang and Manxia Cao

***Abstract*—As an emerging paradigm, orbital edge computing (OEC) has garnered significant attention due to its advantages in global coverage and seamless connectivity. Most existing research adopts a user-centric approach, focusing on enhancing user satisfaction, but overlooks the economic objectives of satellites. To fill this gap, this paper investigates the collaborative optimization of computing resource pricing, task offloading, and resource allocation, aiming to maximize satellite profits while ensuring fair resource distribution to meet user computing demands. Firstly, a bilevel optimization problem (BOP) is established, considering the coupling between satellite and user strategies. In the upper level, satellites determine unit prices to maximize total profit, while in the lower level, users decide offloading modes and required computing resource amounts based on given prices. Moreover, by deriving the relationship between user resource demands and offloading modes, the mixed variable optimization in the original BOP's lower level is transformed into a discrete variable optimization. To address the transformed BOP, we reduce the search space through server pruning and design a nested optimization algorithm, which utilizes particle swarm optimization (PSO) and matching game theory in upper and lower levels, respectively. Lastly, experiments prove our algorithm surpasses state-of-the-art in satellite total profit and user task completion rate.**

***Index Terms*—Orbital edge computing, computation offloading, bilevel optimization, matching game.**

I. Introduction

The rapid growth of compute-intensive and latency-sensitive applications, such as AR/VR and 3D modeling, poses significant challenges to conventional terrestrial networks. As a key 6G technology, mobile edge computing (MEC) mitigates cloud congestion by offloading tasks to edge nodes closer to users [1], [2]. However, MEC deployment is limited in areas such as oceans and remote rural regions due to insufficient communication and power infrastructure [3]. Moreover, terrestrial networks remain vulnerable to disruptions from natural disasters, reducing their reliability in mission-critical scenarios.

To address these challenges, satellite communication has emerged as a promising solution to extend network coverage [4]. Companies such as Telesat and SpaceX are deploying large-scale low Earth orbit (LEO) satellite constellations to provide global broadband services [5]. In 2019, Bradley et al. introduced the concept of orbital edge computing (OEC), drawing inspiration from terrestrial MEC systems [6]. By integrating edge computing into satellite networks, OEC brings computation closer to users and improves service responsiveness [7]. As a result, satellite–terrestrial integrated networks (STINs) are expected to enhance global connectivity and enable next-generation communication infrastructures [8], [9].

Recent research on OEC spans several key areas, including satellite networking [10], research platforms [11], system architectures [12], [13], as well as resource management and computation offloading [14], [15]. Carlos et al. [16] optimized global flight plans for large satellite constellations using genetic algorithms, while Kssing et al. [17] introduced Hypatia, a simulation tool for LEO constellations. Zhai et al. [18] explored decentralized federated learning, leveraging satellite constellation topology for efficient model aggregation without central servers. These advancements enhance OEC theory and drive the development of space-air-ground integrated networks.

Significant progress has been made in the areas of resource management and computation offloading within OEC systems. However, most existing studies adopt a user-centric perspective, focusing on task offloading and resource allocation to reduce latency or energy consumption [19]–[22]. While these approaches improve user performance, they often overlook the economic goals and resource sustainability of satellite operators. Given the limited onboard computing resources [23]–[25], prioritizing user satisfaction alone may result in excessive resource allocation. To address this, some studies have proposed dynamic pricing mechanisms to enhance operator revenue and improve resource efficiency [26], [27]. Nonetheless, these approaches often treat user offloading and satellite pricing as separate problems, overlooking the strong coupling between them. In practice, pricing directly affects users' offloading decisions based on perceived cost-efficiency. These decisions, in turn, determine computing demand and satellite load, which subsequently influence the provider's pricing strategy. This interaction creates a feedback loop, and ignoring it may result in suboptimal performance and inefficient resource utilization.

To fill the gap, this paper proposes a bilevel optimization framework that captures the inherent coupling among satellite pricing, task offloading, and resource allocation, enabling their collaborative optimization within a unified framework. In this framework, satellite operators act as service providers, setting prices for computing resources to maximize their total profit. Ground users respond by selecting offloading modes and determining resource demands based on cost-efficiency and task requirements. The pricing decisions directly influence users' offloading behaviors, which in turn affect the distri-

Manuscript received September 28, 2024; accepted May 4, 2025. *(Corresponding author: Qi Wang.)*

Xinyu Wang, Qi Wang, Qingshan Wang, and Manxia Cao are with the School of Mathematics, Hefei University of Technology, Hefei, Anhui 230601, China. E-mail: {2022111428, manxiacao}@mail.hfut.edu.cn, {qswang, wangq}@hfut.edu.cn.

bution of resource demands across satellites and feed back into subsequent pricing strategies, forming a coupled decision loop. The specific contributions are delineated as follows:

- We formulate a bilevel optimization problem (BOP) for satellite computing resource pricing, aiming to maximize satellite profits while ensuring fair resource allocation for user offloading. The upper level optimizes satellite pricing to maximize profit, while the lower level determines users' offloading decisions and resource demands under time and resource constraints. This formulation differs from prior studies that separately consider pricing or offloading, and more accurately captures the supply-demand coupling in OEC systems.
- By analyzing the relationship between optimal resource demand and offloading modes, we transform the lower-level MINLP problem into a discrete optimization problem. We then propose a nested optimization algorithm, employing particle swarm optimization (PSO) for satellite profit maximization at the upper level and matching game theory for user profit maximization at the lower level.
- We assess the proposed algorithm's performance in terms of satellite total profit, user task completion rate, and execution time. Experiments demonstrate that our algorithm is effective in reducing the runtime while performing better compared to state-of-the-art algorithms.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III introduces the satellite edge computing scenario and defines the profit functions for users and satellites. Section IV describes the computation offloading process and formulates the bilevel optimization problem. In Section V, we transform the original BOP into a tractable form and propose a nested bilevel optimization algorithm. Section VI presents the experimental results, and Section VII concludes the paper.

II. Related Work

Computation offloading in MEC-augmented satellite-terrestrial networks has gained significant attention [28], [29]. Most existing studies adopt a user-centric perspective, optimizing offloading and resource allocation strategies to enhance user satisfaction. For example, Cui et al. [19] minimized task completion latency in a hybrid OEC system using deep reinforcement learning and convex optimization. Chen et al. [20] jointly allocated computing and communication resources in a mixed-task model, solving an MINLP problem to maximize user-weighted energy efficiency. Similarly, Cao et al. [21] and Huang et al. [22] employed alternating optimization and multi-agent deep reinforcement learning to improve resource utilization and reduce energy consumption. However, by focusing solely on user satisfaction, these approaches often overlook the limited nature of satellite resources. As a result, resource allocation may be excessive, making it difficult to balance user demands with the operator's economic objectives.

Some studies have explored satellite pricing strategies to improve operator revenue and resource utilization. Deng et al. [26] proposed a pricing mechanism for user association, spectrum allocation, and data pricing. Li et al. [27] developed a dynamic game model for joint pricing and power allocation in multibeam satellite systems to balance inter-cell interference and operational profit. While these studies integrate pricing, they typically assume users passively respond to fixed pricing strategies, failing to capture the dynamic interaction between pricing, offloading, and resource allocation. Recent work has addressed this limitation through collaborative optimization frameworks, often leveraging Stackelberg game models. For example, Patrizi et al. [30] proposed a reputation-based UAV tracking and data collection framework using Stackelberg games to optimize offloading decisions and effort-based pricing. Zhang et al. [31] designed a pricing-driven mode selection scheme where operators optimize pricing via a Stackelberg game, while users adopt an evolutionary game for mode selection. However, while Stackelberg-based approaches effectively model hierarchical decision-making, they primarily emphasize strategy stability rather than global optimality, which may limit the maximization of satellite profits.

III. System Model

This section presents the LEO-assisted edge computing scenario, followed by the communication, computation, and energy models, as well as the total profit formulations for users and satellites.

A. Scenario Model

The satellite edge computing scenario is illustrated in Fig. 1. It involves terrestrial users located in remote areas, such as deserts, where traditional base stations cannot provide connectivity. The system consists of a constellation of LEO satellites arranged in efficient configurations to ensure global coverage. Each satellite is equipped with an edge server to provide computational services to users. In the absence of terrestrial connectivity, users can either process tasks locally or offload them to satellites via user data links (UDLs). The communication between users and satellites is achieved through Orthogonal Frequency Division Multiple Access (OFDMA).

Due to satellite and Earth rotations, continuous communication between all satellites and users is not always feasible. The nearest satellite with an active connection to the user area is referred to as the access satellite. Inter-satellite links (ISLs) enable high-speed, low-latency communication between the access satellite and other satellites in the constellation. These links support real-time task scheduling based on task requirements and satellite computing capabilities. Upon task completion, results are returned to users, and the access satellite is reselected according to satellite trajectories, user locations, and coverage conditions.

The model consists of M ground users and N satellites, one of which is the access satellite. The sets of users and satellites are denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$ and $\mathcal{N} = \{1, 2, \ldots, N\}$, respectively. The m-th user and the n-th satellite are denoted by U_m and S_n, with S_1 being the access satellite. There are ISLs between $S_2 \sim S_N$ and S_1. Each user has an indivisible computational task, denoted as $\mathcal{T}_m$, where $m \in \mathcal{M}$. The task $\mathcal{T}_m$ is represented by the 3-tuple $\mathcal{T}_m = \{\mathrm{Q}_m, \mathrm{B}_m, \mathrm{T}_m^{\max}\}$, where Q_m is the input data size (bits), B_m is the task

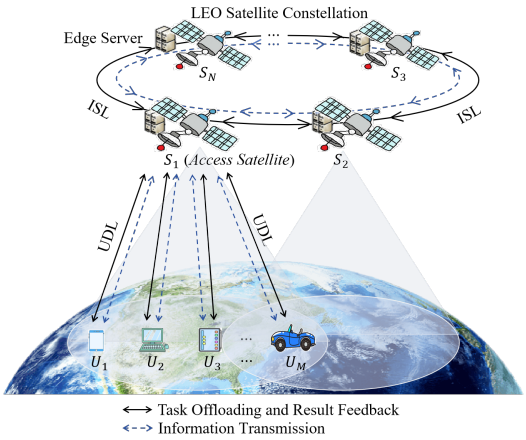

Fig. 1: LEO satellite-assisted OEC system.

processing density (cycles/bit), and $\mathrm{T}_m^{\max}$ is the maximum tolerated delay (s). The matching strategy $\mathbf{\Phi}$ determines where each user performs its task: $\Phi_{mn} = 1$ if $\mathcal{T}_m$ is processed on S_n, otherwise, $\Phi_{mn} = 0$. Similarly, $\Phi_{m0} = 1$ if $\mathcal{T}_m$ is processed locally, otherwise, $\Phi_{mn} = 0$. we define the vector $\mathbf{f} = \{f_1, \ldots, f_m, \ldots, f_M\}$ as users' computing resource demands and $\mathbf{P} = \{P_1, \ldots, P_n, \ldots, P_N\}$ as satellites' computing resource unit prices, where f_m ($m \in \mathcal{M}$) represents the amount of computing resources required by U_m and P_n ($n \in \mathcal{N}$) represents unit price of computing resources for S_n. If U_m chooses to compute locally, i.e., $\Phi_{m0} = 1$, the locally requested computing resources are denoted by f_m^0. If U_m chooses to offload its task to S_n, i.e., $\Phi_{mn} = 1$, the computing resources requested from S_n are denoted by f_m^n. Therefore, the computing resources required by U_m is

$$f_m = \begin{cases} f_m^0, & \text{if } \Phi_{m0} = 1, \\ f_m^n, & \text{if } \Phi_{mn} = 1. \end{cases} \tag{1}$$

B. Communication Model

In this system, users can process tasks locally or offload them to the constellation via the access satellite. The communication links include ground-to-satellite, inter-satellite, and satellite-to-ground.

1) Ground-to-Satellites Communication: In this communication process, users transmit their tasks to S_1 via wireless backhaul links over the Ka-band. The data transmission rate R_m^{gnd} from U_m to S_1 can be given by [32]

$$R_m^{\mathrm{gnd}} = W_{\mathrm{Ka}} \log_2\left(1 + \frac{P_m^{\mathrm{gnd}} H_m^{\mathrm{gnd}}}{N_0}\right), \tag{2}$$

where W_{Ka} is the bandwidth, P_m^{gnd} is the transmission power, H_m^{gnd} is the channel gain, and N_0 denotes the background noise power. Therefore, the transmission delay t_m^{gnd} required by U_m to transmit its task to S_1 is

$$t_m^{\mathrm{gnd}} = \frac{\mathrm{Q}_m}{R_m^{\mathrm{gnd}}}. \tag{3}$$

The propagation delay t^{prop} in the communication process between users and satellites is

$$t^{\mathrm{prop}} = \frac{D}{C}, \tag{4}$$

where D represents the altitude of the satellites from the Earth's surface, C is the speed of light.

2) Inter-Satellites Communication: Due to the access satellite may choose to compute tasks itself or transmit tasks to other satellites, it is necessary to consider the communication process between the access satellite and other satellites. The data transmission rate R_{ISL} between satellites is

$$R_{\mathrm{ISL}} = W_{\mathrm{ISL}} \log_2(1 + R_{\mathrm{SN}}), \tag{5}$$

where W_{ISL} is the channel bandwidth between satellites, R_{SN} is the signal-to-noise ratio for laser communications, which is

$$R_{\mathrm{SN}} = \frac{\gamma^2 (P^{\mathrm{r}})^2}{\sigma^2}, \tag{6}$$

where γ is detector response rate, P^{r} is the received power, and σ^2 is the receiver noise. Therefore, the transmission delay t_m^{ISL} for S_1 to transmit the task $\mathcal{T}_m$ to other satellites is

$$t_m^{\mathrm{ISL}} = \frac{\mathrm{Q}_m}{R_{\mathrm{ISL}}}. \tag{7}$$

3) Satellites-to-Ground Communication: Similar to [21], the communication delay for satellites to transmit processed results back to the ground is neglected due to the small data size.

Thus, when U_m decides to delegate its task to the satellite constellation, if the access satellite provides computing services, the required transmission delay t_{m1}^{trans} is given by

$$t_{m1}^{\mathrm{trans}} = t_m^{\mathrm{gnd}} = \frac{\mathrm{Q}_m}{R_m^{\mathrm{gnd}}}. \tag{8}$$

Otherwise, i.e., other satellites provide computing services, the required transmission delay t_{mn}^{trans} is

$$\begin{aligned} t_{mn}^{\mathrm{trans}} &= t_m^{\mathrm{gnd}} + t_m^{\mathrm{ISL}} \\ &= \frac{\mathrm{Q}_m}{R_m^{\mathrm{gnd}}} + \frac{\mathrm{Q}_m}{R_{\mathrm{ISL}}}, n \neq 1. \end{aligned} \tag{9}$$

C. Computing Model

1) Local Computing: If U_m chooses local computation, the requested computing resources are f_m^0 according to equation (1). Therefore, the local computation delay for U_m is:

$$t_{m0}^{\mathrm{comp}} = \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^0}. \tag{10}$$

2) Satellite Computing: If U_m offloads its task to S_n, the requested computing resources from S_n are f_m^n. The computation delay for S_n to execute $\mathcal{T}_m$ is

$$t_{mn}^{\mathrm{comp}} = \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^n}. \tag{11}$$

Thus, if U_m computes locally, i.e., $\Phi_{m0} = 1$, the total time it takes is

$$t_{m0} = t_{m0}^{\mathrm{comp}} = \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^0}. \tag{12}$$

If U_m offloads its task to the access satellite, i.e., $\Phi_{m1} = 1$, the total time it takes is

$$\begin{aligned} t_{m1} &= t_{m1}^{\mathrm{trans}} + 2t^{\mathrm{prop}} + t_{m1}^{\mathrm{comp}} \\ &= \frac{\mathrm{Q}_m}{R_m^{\mathrm{gnd}}} + \frac{2D}{C} + \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^1}. \end{aligned} \tag{13}$$

Otherwise, i.e., $\Phi_{mn}=1$, $n\neq 0,1$, the total time it takes is

$$\begin{aligned} t_{mn} &= t_{mn}^{\text{trans}} + 2t^{\text{prop}} + t_{mn}^{\text{comp}} \\ &= \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{\mathrm{Q}_m}{R_{\text{ISL}}} + \frac{2D}{C} + \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^n}, n\neq 0,1. \end{aligned} \tag{14}$$

The total time for U_m to complete its computational task is

$$\begin{aligned} t_m^{\text{total}} &= \Phi_{m0} t_{m0} + \Phi_{m1} t_{m1} + \sum_{n=2}^{N} \Phi_{mn} t_{mn} \\ &= \Phi_{m0} \cdot \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^0} + \Phi_{m1}\left(\frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{2D}{C} + \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^1}\right) \\ &+ \sum_{n=2}^{N} \Phi_{mn}\left(\frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{\mathrm{Q}_m}{R_{\text{ISL}}} + \frac{2D}{C} + \frac{\mathrm{Q}_m \mathrm{B}_m}{f_m^n}\right). \end{aligned} \tag{15}$$

D. Energy Consumption Model

1) User Energy Consumption: If U_m computes locally, the computational energy it consumes is [33]

$$E_m^{\text{comp\_user}} = \kappa_m^{\text{user}} (f_m^0)^2 \mathrm{Q}_m \mathrm{B}_m, \tag{16}$$

where $\kappa_m^{\text{user}} > 0$ is the effective capacitance coefficient, which is associated with the CPU of U_m' server. If U_m offloads its task, the transmission energy it needs to consume is

$$E_m^{\text{trans\_user}} = P_m^{\text{gnd}} t_m^{\text{gnd}} = \frac{P_m^{\text{gnd}} \mathrm{Q}_m}{R_m^{\text{gnd}}}. \tag{17}$$

2) Satellite Energy Consumption: If access satellite transmits $\mathcal{T}_m$ to other satellites, the transmission energy consumed is

$$E_n^{\text{trans\_sat}} = P_{\text{ISL}} t_m^{\text{ISL}} = \frac{P_{\text{ISL}} \mathrm{Q}_m}{R_{\text{ISL}}}, \tag{18}$$

where P_{ISL} is the transmission power of the access satellite. The computational energy consumption of S_n is [33]

$$E_n^{\text{comp\_sat}} = \kappa_n^{\text{sat}} (f_m^n)^2 \mathrm{Q}_m \mathrm{B}_m, \tag{19}$$

where $\kappa_n^{\text{sat}} > 0$ is the effective capacitance coefficient, it is related to the CPU of S_n' edge server.

E. Profit Model

1) Profit of Users: If U_m performs local computation, i.e., $\Phi_{m0}=1$, it consumes computational energy and its profit is

$$\begin{aligned} U_{m0}^{\text{user}} &= \lambda \mathrm{Q}_m - \eta_1 E_m^{\text{comp\_user}} \\ &= \lambda \mathrm{Q}_m - \eta_1 \kappa_m^{\text{user}} (f_m^0)^2 \mathrm{Q}_m \mathrm{B}_m, \end{aligned} \tag{20}$$

where $\lambda \mathrm{Q}_m$ represents the reward and λ denotes the reward coefficient. $\eta_1 > 0$ is the conversion factor of the computational energy for users. If U_m offloads its task to S_n, i.e., $\Phi_{mn}=1$, it consumes transmission energy and obtains computing resources from S_n. Hence, its profit is

$$\begin{aligned} U_{mn}^{\text{user}} &= \lambda \mathrm{Q}_m - P_n f_m^n - \xi_1 E_m^{\text{trans\_user}} \\ &= \lambda \mathrm{Q}_m - P_n f_m^n - \xi_1 \frac{P_m^{\text{gnd}} \mathrm{Q}_m}{R_m^{\text{gnd}}}, \end{aligned} \tag{21}$$

where P_n is the unit price for computing resources of S_n, and $\xi_1 > 0$ is the conversion factor of the transmission energy for users. Therefore, the profit of U_m is given by

$$\begin{aligned} U_m^{\text{user}} &= \Phi_{m0} U_{m0}^{\text{user}} + \sum_{n=1}^{N} \Phi_{mn} U_{mn}^{\text{user}} \\ &= \Phi_{m0}[\lambda \mathrm{Q}_m - \eta_1 \kappa_m^{\text{user}} (f_m^0)^2 \mathrm{Q}_m \mathrm{B}_m] \\ &+ \sum_{n=1}^{N} \Phi_{mn}(\lambda \mathrm{Q}_m - P_n f_m^n - \xi_1 \frac{P_m^{\text{gnd}} \mathrm{Q}_m}{R_m^{\text{gnd}}}). \end{aligned} \tag{22}$$

The total profit for all ground users is

$$U^{\text{user}}(\mathbf{P}, \mathbf{\Phi}, \mathbf{f}) = \sum_{m=1}^{M} U_m^{\text{user}}. \tag{23}$$

2) Profit of Satellites: If S_1 executes task $\mathcal{T}_m$ locally, it needs to consume computational energy, the profit that S_1 can earn from completing $\mathcal{T}_m$ is

$$\begin{aligned} U_{m1}^{\text{satellite}} &= P_1 f_m^1 - \eta_2 E_1^{\text{comp\_sat}} \\ &= P_1 f_m^1 - \eta_2 \kappa_1^{\text{sat}} (f_m^1)^2 \mathrm{Q}_m \mathrm{B}_m, \end{aligned} \tag{24}$$

where $\eta_2 > 0$ is the conversion factor of the computational energy for satellites. Hence, the profit of S_1 is

$$\begin{aligned} U_1^{\text{satellite}} &= \sum_{m=1}^{M} \Phi_{m1} U_{m1}^{\text{satellite}} \\ &= \sum_{m=1}^{M} \Phi_{m1}[P_1 f_m^1 - \eta_2 \kappa_1^{\text{sat}} (f_m^1)^2 \mathrm{Q}_m \mathrm{B}_m]. \end{aligned} \tag{25}$$

If $S_n (n\neq 1)$ completes the task $\mathcal{T}_m$, it needs to consume transmission and computational energy for revenue. Therefore, the profit that S_n can earn from completing $\mathcal{T}_m$ is

$$\begin{aligned} U_{mn}^{\text{satellite}} &= P_n f_m^n - \eta_2 E_n^{\text{comp\_sat}} - \xi_2 E_n^{\text{trans\_sat}} \\ &= P_n f_m^n - \eta_2 \kappa_n^{\text{sat}} (f_m^n)^2 \mathrm{Q}_m \mathrm{B}_m \\ &- \xi_2 \frac{P_{\text{ISL}} \mathrm{Q}_m}{R_{\text{ISL}}}, n\neq 1, \end{aligned} \tag{26}$$

where $\xi_2 > 0$ is the conversion factor of the transmission energy for satellites. Thus, the profit of S_n is given by

$$\begin{aligned} U_n^{\text{satellite}} &= \sum_{m=1}^{M} \Phi_{mn} U_{mn}^{\text{satellite}} \\ &= \sum_{m=1}^{M} \Phi_{mn}[P_n f_m^n - \eta_2 \kappa_n^{\text{sat}} (f_m^n)^2 \mathrm{Q}_m \mathrm{B}_m \\ &- \xi_2 \frac{P_{\text{ISL}} \mathrm{Q}_m}{R_{\text{ISL}}}], n\neq 1. \end{aligned} \tag{27}$$

The total profit for all satellites is

$$U^{\text{satellite}}(\mathbf{P}, \mathbf{\Phi}, \mathbf{f}) = \sum_{n=1}^{N} U_n^{\text{satellite}}. \tag{28}$$

IV. Offloading Process And Problem Formulation

In this section, the specific process of computation offloading is described, and the BOP studied in this paper is proposed.

A. Offloading Process

The computation offloading process illustrated in Fig. 1 consists of three stages: information collection, algorithm execution and task offloading.

- **Collecting Information**. Users transmit task details (e.g., computation volume, maximum tolerance delay) to S_1, and other satellites share their computing capacity with S_1 ("Information Transmission" in Fig. 1).
- **Executing Algorithm**. S_1 processes the received information using the proposed algorithm (PSO-MG in Section V) to determine offloading modes, resource demands, and pricing. The decisions are then sent to users and satellites ("Information Transmission" in Fig. 1).
- **Offloading Computation**. Based on the received decisions, users either compute locally or offload tasks, while satellites process and return results ("Task offloading and Result Feedback" in Fig. 1).

B. Problem Formulation

As shown in Fig. 2, the interaction process between users and satellites is as follows:

1) Satellites set initial computing resource unit price $\mathbf{P}$.
2) Given $\mathbf{P}$, the users determine the optimal decision $\mathbf{\Phi}^*$ and $\mathbf{f}^*$ to maximize their total profit.
3) Given $\mathbf{\Phi}^*$ and $\mathbf{f}^*$, the satellites update $\mathbf{P}$ to maximize their total profit.
4) Return to 2) until the convergence condition is reached.

Satellites act as leaders in this interaction, while users serve as followers who make decisions in response to the satellites' strategies. To dynamically adjust satellite resource pricing, this paper formulates a bilevel optimization problem, where maximizing users' total profit serves as a constraint to maximize satellite profit. The lower level determines users' offloading decisions $\mathbf{\Phi}$ and resource demands $\mathbf{f}$ in response to pricing, whereas the upper level optimizes the pricing strategy vector $\mathbf{P}$ for satellites. The BOP is

$$
\begin{aligned}
\mathcal{P}_0: \max_{\mathbf{P},\mathbf{\Phi},\mathbf{f}} \quad & U^{\text{satellite}} \\
\text{s.t. } & \mathcal{C}_1: P_n^{\min} \le P_n \le P_n^{\max}, \quad \forall n \in \mathcal{N} \\
& \{\mathbf{\Phi}, \mathbf{f}\} = \arg\max_{\mathbf{\Phi},\mathbf{f}} \ U^{\text{user}} \\
& \mathcal{C}_2: \Phi_{mn} \in \{0,1\}, \quad \forall m \in \mathcal{M}, \forall n \in \{0\} \cup \mathcal{N} \\
& \mathcal{C}_3: \sum_{n=0}^{N} \Phi_{mn} \le 1, \quad \forall m \in \mathcal{M} \\
& \mathcal{C}_4: \Phi_{m0} f_m \le f_m^{\text{max\_user}}, \quad \forall m \in \mathcal{M} \\
& \mathcal{C}_5: \sum_{m=1}^{M} \Phi_{mn} f_m \le f_n^{\text{max\_sat}}, \quad \forall n \in \mathcal{N} \\
& \mathcal{C}_6: t_m^{\text{total}} \le \mathrm{T}_m^{\max}, \quad \forall m \in \mathcal{M} \\
& \mathcal{C}_7: f_m \ge 0, \quad \forall m \in \mathcal{M}
\end{aligned}
$$

the argmax term in the lower level problem represents the users' best response strategy, where users optimize the offloading decision $\mathbf{\Phi}$ and resource demand $\mathbf{f}$ to maximize their utility. Where $\mathcal{C}_1$ defines the valid range for resource pricing; $\mathcal{C}_2$ ensures each user completes its task either locally or via offloading; $\mathcal{C}_3$ enforces task indivisibility and single-server processing. $f_m^{\text{max\_user}}$ denotes the maximum computing resource for U_m, $\mathcal{C}_4$ ensures local computation demand does not exceed the threshold. $f_n^{\text{max\_sat}}$ denotes the maximum computing resource for S_n, $\mathcal{C}_5$ limits the total computing resources that S_n can allocate to users. $\mathcal{C}_6$ ensures each user's task completion time does not exceed the maximum allowed delay; and $\mathcal{C}_7$ guarantees the feasibility of users' computing resource demands.

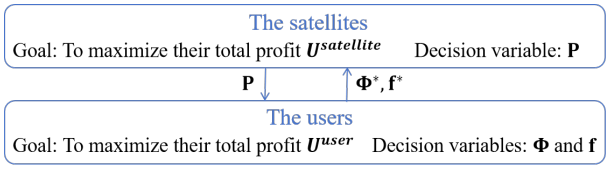

Fig. 2: Interaction between satellites and users.

V. Proposed Approach

Since $\mathcal{P}_0$ is a nested bilevel optimization problem, we design a nested bilevel optimization algorithm (PSO-MG), as shown in Fig. 3. The algorithm consists of three main components: the upper level pricing optimization, the lower level offloading decision, and the preprocessing module. To reduce computational complexity, the preprocessing module (Alg. 1) first generates a candidate server set for each user. In each iteration, the upper level applies a PSO-based optimization to explore satellite pricing strategies $\mathbf{P}$. For each candidate solution in PSO, the lower level invokes a multi-round matching game (Alg. 2) to determine the corresponding optimal offloading decisions $\mathbf{\Phi}$. These decisions are then fed back to the upper level to evaluate and update the pricing vector $\mathbf{P}$.

A. Problem Transformation

If U_m computes locally, $\mathcal{C}_6$ should be satisfied, i.e., $t_{m0} \le \mathrm{T}_m^{\max}$. According to (12), f_m^0 at least satisfies

$$f_m^0 \ge \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max}}. \tag{29}$$

Based on (20), the smaller of f_m^0, the higher of U_m' profit. The optimal local computing resource demand is

$$(f_m^0)^* = \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max}}. \tag{30}$$

According to (13), only if $t_{m1}^{\text{trans}} + 2t^{\text{prop}} \le \mathrm{T}_m^{\max}$ holds, i.e., $\frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{2D}{C} \le \mathrm{T}_m^{\max}$, will U_m offload its task to the access satellite. Additionally, $\mathcal{C}_6$ should be satisfied, i.e., $t_{m1} \le \mathrm{T}_m^{\max}$. Therefore, f_m^1 at least satisfies

$$f_m^1 \ge \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{2D}{C}}. \tag{31}$$

Given the fixed P_1, U_m can achieve higher profit by requesting fewer computing resources, as shown in (21). The optimal amount of resources requested from S_1 is

$$(f_m^1)^* = \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{2D}{C}}. \tag{32}$$

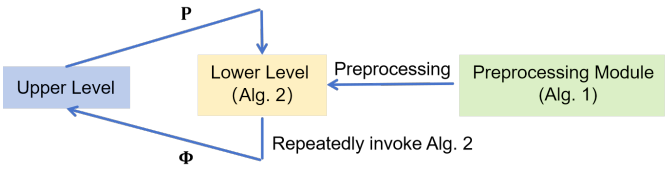

Fig. 3: Architecture of the algorithm PSO-MG.

According to (14), only if $t_{mn}^{\text{trans}} + 2t^{\text{prop}} \leq \mathrm{T}_m^{\max}$ holds, i.e., $\frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{\mathrm{Q}_m}{R_{\text{ISL}}} + \frac{2D}{C} \leq \mathrm{T}_m^{\max}$, will U_m offload its task to $S_n (n \neq 1)$. Similarly, the optimal amount of computing resources requested from $S_n (n \neq 1)$ is

$$(f_m^n)^* = \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{\mathrm{Q}_m}{R_{\text{ISL}}} - \frac{2D}{C}}. \tag{33}$$

Based on (1), the optimal quantity of computing resources required by U_m is

$$f_m^* = \begin{cases} \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max}}, & \text{if } \Phi_{m0} = 1, \\ \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{2D}{C}}, & \text{if } \Phi_{m1} = 1, \\ \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{\mathrm{Q}_m}{R_{\text{ISL}}} - \frac{2D}{C}}, & \text{if } \Phi_{mn} = 1, n \neq 0, 1, \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

It is obvious that $\mathcal{C}_7$ is satisfied if (34) is hold. Hence, as long as $\mathbf{\Phi}$ is known, the computing resources demanded by users can be determined. Therefore $\mathcal{P}_0$ can be transformed as

$$\begin{aligned} \mathcal{P}_1 : \max_{\mathbf{P}, \mathbf{\Phi}} \quad & U^{\text{satellite}} \\ \text{s.t. } & \mathcal{C}_1 : P_n^{\min} \leq P_n \leq P_n^{\max}, \quad \forall n \in \mathcal{N} \\ & \mathbf{\Phi} = \arg\max_{\mathbf{\Phi}} \ U^{\text{user}} \\ & \mathcal{C}_2 : \Phi_{mn} \in \{0, 1\}, \quad \forall m \in \mathcal{M}, \forall n \in \{0\} \cup \mathcal{N} \\ & \mathcal{C}_3 : \sum_{n=0}^{N} \Phi_{mn} \leq 1, \quad \forall m \in \mathcal{M} \\ & \mathcal{C}_8 : \Phi_{m0} f_m^* \leq f_m^{\text{max\_user}}, \quad \forall m \in \mathcal{M} \\ & \mathcal{C}_9 : \sum_{m=1}^{M} \Phi_{mn} f_m^* \leq f_n^{\text{max\_sat}}, \forall n \in \mathcal{N} \end{aligned}$$

The original MINLP problem at the lower level of $\mathcal{P}_0$ is transformed into a discrete optimization problem in $\mathcal{P}_1$, making the optimization issue more tractable.

B. Generation of Candidate Servers Set

Each user has $N + 1$ server options, which makes $\mathcal{P}_1$ computationally complex. However, not all servers satisfy the task constraints. To reduce the computational burden, a candidate server set is defined for each user.

Given limited local computing resources, $\mathcal{C}_8$ must be met for U_m to process tasks locally. Thus, the condition to enable local computation based on (34) is

$$f_m^{\text{max\_user}} \geq \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max}}. \tag{35}$$

The computing resources of satellites are limited, based on $\mathcal{C}_9$ and (34), the necessary condition for U_m to be allowed to offload its task to S_1 is

$$f_1^{\text{max\_sat}} \geq \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{2D}{C}}. \tag{36}$$

According to (13), another precondition for U_m to offload its task to S_1 is given as

$$\mathrm{T}_m^{\max} \geq \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{2D}{C}. \tag{37}$$

Similarly, the necessary condition for U_m to be allowed to offload its task to $S_n (n \neq 1)$ is

$$f_n^{\text{max\_sat}} \geq \frac{\mathrm{Q}_m \mathrm{B}_m}{\mathrm{T}_m^{\max} - \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} - \frac{\mathrm{Q}_m}{R_{\text{ISL}}} - \frac{2D}{C}}, n \neq 1. \tag{38}$$

In addition, according to (14), another precondition is

$$\mathrm{T}_m^{\max} \geq \frac{\mathrm{Q}_m}{R_m^{\text{gnd}}} + \frac{\mathrm{Q}_m}{R_{\text{ISL}}} + \frac{2D}{C}. \tag{39}$$

In summary, for each user $U_m (m \in \mathcal{M})$, a candidate server set τ_m is generated. As shown in Alg. 1, the process sequentially checks each server: if inequality (35) holds, the local server (with index 0) is added to τ_m; if inequalities (36) and (37) are satisfied, the access satellite's edge server (with index 1) is included; and if inequalities (38) and (39) hold, the edge server of $S_n (n \neq 1)$ (with index n) is added. This approach reduces the search space of the lower level problem and accelerates the solution process, with the optimal server for U_m selected from τ_m.

Algorithm 1 Generation of Candidate Servers Set

```
for m = 1 to M do
    τ_m = ∅;
    for n = 0 to N do
        if n = 0 and (35) is satisfied then
            τ_m ← τ_m ∪ {0};
        else if n = 1 and (36), (37) are satisfied then
            τ_m ← τ_m ∪ {1};
        else if n ≠ 0, 1 and (38), (39) are satisfied then
            τ_m ← τ_m ∪ {n};
        end if
    end for
end for
return τ_1, τ_2, ..., τ_M
```

C. Lower Level Optimization

The lower level optimization problem of $\mathcal{P}_1$ is

$$\begin{aligned} \mathcal{P}_2 : \max_{\mathbf{\Phi}} \quad & U^{\text{user}} \\ \text{s.t. } & \mathcal{C}_2 : \Phi_{mn} \in \{0, 1\}, \quad \forall m \in \mathcal{M}, \forall n \in \{0\} \cup \mathcal{N} \\ & \mathcal{C}_3 : \sum_{n=0}^{N} \Phi_{mn} \leq 1, \quad \forall m \in \mathcal{M} \\ & \mathcal{C}_8 : \Phi_{m0} f_m^* \leq f_m^{\text{max\_user}}, \quad \forall m \in \mathcal{M} \\ & \mathcal{C}_9 : \sum_{m=1}^{M} \Phi_{mn} f_m^* \leq f_n^{\text{max\_sat}}, \forall n \in \mathcal{N} \end{aligned}$$

$\mathcal{P}_2$ is a discrete variable optimization problem that maximizes the users' total profit, solved using the matching game approach. Firstly, the fitness evaluation functions are defined, then the concept of user priority is presented, and the specific process of matching game is introduced through Alg. 2.

1) Fitness Evaluation Functions: To evaluate the offloading decision $\mathbf{\Phi}$, two fitness evaluation functions are introduced. In scenarios with a large number of users, limited computing resources may prevent full task completion, making the task completion rate a key priority. Therefore, the optimization process first prioritizes maximizing task completion and then aims to maximize the total profit for users. The two fitness functions are

$$F_1(\mathbf{\Phi}) = \sum_{m=1}^{M} \sum_{n=0}^{N} \Phi_{mn}. \tag{40}$$

$$F_2(\mathbf{P}, \mathbf{\Phi}) = U^{\text{user}} = \sum_{m=1}^{M} U_m^{\text{user}}. \tag{41}$$

When comparing two offloading decisions, the one with the higher F_1 is preferred. If both yield the same F_1, the decision with the higher F_2 is selected.

2) User Priorities: Before determining $\mathbf{\Phi}$, user priorities must be established, as random prioritization may hinder the maximization of task completion rates. As shown in Fig. 4, U_1 has candidate servers 1 and 2 ($\tau_1 = 1, 2$), whereas U_2 has only server 1 ($\tau_2 = 1$). In Fig. 4 (a), if U_1 is matched with S_1 first, U_2 may fail to complete its task due to the limited resources on S_1. In contrast, Fig. 4(b) shows that if U_2 is given higher priority and matched with S_1 first, U_1 can still complete its task by offloading to S_2. Accordingly, user priorities are assigned based on the size of their candidate server sets: the fewer servers in τ_m, the higher the priority for U_m.

3) Matching Game: The matching game is a type of game theory that has relatively wide applications in the MEC field [34]–[36]. In determining user offloading decisions, it is not only important to select the optimal computation mode (local processing or satellite computation), but also to choose the best cooperation partner, i.e., the specific satellite for task offloading. This requires considering the cooperation willingness between users and satellites. Moreover, as the number of users grows, the exponential increase in decision combinations makes traditional heuristic algorithms inefficient due to high computational costs and poor scalability. To address this, a many-to-many matching game is formulated, in which each user selects either a local server or a satellite edge server, while satellites determine whether to accept user requests based on their preferences and resource constraints. This matching game is denoted as $\mathfrak{G}(\mathcal{M}, \{0\} \cup \mathcal{N}, \Gamma, \succ_m, \succ_n)$, where $\mathcal{M}$ and $\{0\} \cup \mathcal{N}$ represent the participants in the matching game, Γ is the matching function, $\succ_m$ and $\succ_n$ respectively represent the preference relationship of users and satellites.

Definition 1 (Players in the Game): The matching game $\mathfrak{G}$ is a process in which two groups of game participants $(\mathcal{M}, \{0\} \cup \mathcal{N})$ order each other to achieve an optimal match, where $\mathcal{M} = \{1, 2, \ldots, M\}$ is the set of users and $\{0\} \cup \mathcal{N} = \{0, 1, 2, \ldots, N\}$ is the set of servers.

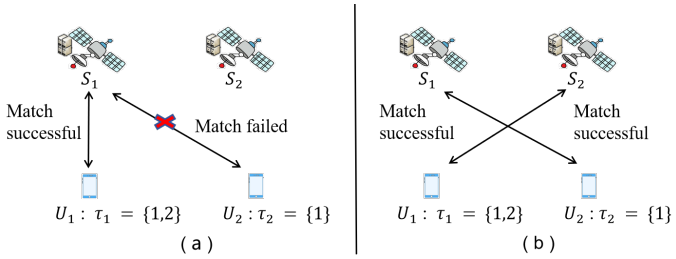

Fig. 4: A demonstration of the impact of whether or not to set user priorities on the final match.

Definition 2 (Matching Function): The matching function $\Gamma : \mathcal{M} \to \{0\} \cup \mathcal{N}$ is a mapping relation between these two sets. Γ needs to satisfy the following conditions:

$$\Gamma(m) \in \{0\}, |\Gamma(m)| = 1, \Phi_{m0} = 1, \tag{42a}$$

$$\Gamma(m) \in \mathcal{N}, |\Gamma(m)| = 1, \Phi_{mn} = 1, \tag{42b}$$

(42a) and (42b) indicate that $U_m (m \in \mathcal{M})$ can only select its local server or one of the edge servers in the LEO constellation, not multiple servers.

Definition 3 (User's Preference Relationship): In the matching process, $U_m (m \in \mathcal{M})$ can select an optimal server from τ_m that can maximize its profit. Therefore, the user's preference relationship is

$$\begin{aligned} &n \succ_m n' \Leftrightarrow U_{mn}^{\text{user}} > U_{mn'}^{\text{user}}, \\ &\forall m \in \mathcal{M}, \forall n, n' \in \tau_m, \end{aligned} \tag{43}$$

where n, n' are both candidate servers for U_m.

Definition 4 (Satellite's Preference Relationship): Since each satellite may receive multiple offloading requests while operating under limited computing resources, it must decide which to accept. To maximize task completion, S_n selects users in ascending order of their computing resource demands until reaching the resource limit. In other words, satellites prioritize users with lower resource requirements. Therefore, the satellite's preference relation is defined as follows

$$\begin{aligned} &m \succ_n m' \Leftrightarrow (f_m^n)^* < (f_{m'}^n)^*, \\ &\forall n \in \mathcal{N}, n \in \tau_m, \tau_{m'}, \end{aligned} \tag{44}$$

where n is a candidate server for m, m'.

The matching game process is outlined in Alg. 2. Users are prioritized based on the ascending number of candidate servers and matched sequentially, with users of the same priority matching simultaneously (lines 1-5). Each user evaluates profits from its candidate servers according to the pricing vector $\mathbf{P}$, ranks them in descending order, and sends a request to the top choice. Satellites prioritize users with lower resource demands, accepting requests sequentially until reaching their resource limit and rejecting the rest. Rejected users proceed to the next server in their preference lists and repeat the process until a match is found or all options are exhausted, forming the initial matching (lines 6-11). Subsequently, swap matching reallocates servers between matched pairs, generating a new offloading matrix $\mathbf{\Phi}'$. If it yields a fitness improvement, the current matrix $\mathbf{\Phi}$ is updated. This process repeats until no further optimization is possible, resulting in the final solution (lines 12-20).

Algorithm 2 Multi-Round Matching Game Algorithm

Input: $\mathcal{M}$, $\mathcal{N}$, $\mathbf{Q}$, $\mathbf{B}$, $\mathbf{T}^{\max}$, and $\mathbf{P}$
Output: $\mathbf{\Phi}$

/*Initial solution construction*/
Initialize $\mathbf{\Phi} = \mathbf{0}$;
Generate user priorities: $\mathbf{L} = \{l_1, \ldots, l_M\}$; // l_m is U_m's priority, $l_m^{\max}$ is the maximum value in $\mathbf{L}$.
for $l = 1$ **to** $l_m^{\max}$ **do**
  $\{U_m : l_m = l, \forall m \in \mathcal{M}\} \leftarrow l$.
  U_m constructs preference list from τ_m based on $\mathbf{P}$.
  Users either compute locally or send requests to the first satellite in their list.
  S_n accepts users based on the preference relationship until its resource limit $f_n^{\max\_sat}$ is reached.
  Rejected users send requests to the next server until matched or all options are exhausted.
end for
Obtain the initial solution $\mathbf{\Phi}$.
/* Swap-matching */
repeat
  Select matched pairs (i, j) and (i', j') from $\mathbf{\Phi}$, ensuring $j, j' \in \tau_i, \tau_{i'}$.
  Swap servers between pairs (i, j') and (i', j) to form new matching matrix $\mathbf{\Phi}'$.
  if $F_2(\mathbf{P}, \mathbf{\Phi}') > F_2(\mathbf{P}, \mathbf{\Phi})$ **then**
    Update $\mathbf{\Phi} \leftarrow \mathbf{\Phi}'$
  end if
until No further improvement: $F_2(\mathbf{P}, \mathbf{\Phi}') \leq F_2(\mathbf{P}, \mathbf{\Phi})$
return $\mathbf{\Phi}$

According to standard matching theory [37], if the proposed Alg. 2 yields a two-sided exchange-stable (2ES) matching, as defined in Def. 5, this indicates that the matching game has reached a Nash equilibrium.

Definition 5: Assuming the current match is $\Gamma(m) = n$, then Γ is a 2ES matching if there does not exist a blocking pair (m', n') that satisfies: 1) $U_{mn'}^{\text{user}} > U_{mn}^{\text{user}}$, i.e., $n' \succ_m n$; 2) $U_{m'n}^{\text{user}} > U_{m'n'}^{\text{user}}$, i.e., $n \succ_{m'} n'$; 3) $(f_m^n)^* > (f_{m'}^n)^*$, i.e., $m' \succ_n m$; 4) $(f_{m'}^{n'})^* > (f_m^{n'})^*$, i.e., $m \succ_{n'} m'$.

Next, we prove the absence of blocking pairs using a proof by contradiction.

Theorem 1: The matching game in Alg. 2 can reach a Nash equilibrium.

Proof 1: The current match is $\Gamma(m) = n$, and we assume that there exists a blocking pair (m', n') that satisfies conditions 1) to 4) of Def. 5, then it follows from the assumption that $n' \succ_m n$ and $m \succ_{n'} m'$. There are three possible situations for the existence of this blocking pair as follows:

- Server n' is not in the candidate set of user m. This situation contradicts the condition $n' \succ_m n$.
- Server n' is in the candidate set of user m while n' is unmatched. User m will send its matching request to server n' rather than n due to the higher preference. Thus, user m should be matched with server n', which is against the assumption.
- Server n' is in the candidate set of user m while n' is matched and its computing resources have been depleted. This implies that there exists a user m' that makes $m' \succ_{n'} m$ hold, which is violate the assumption.

Therefore, no blocking pairs exist, which confirms that the matching game reaches a Nash equilibrium. Theorem 1 is proved.

D. Upper Level Optimization

To evaluate the effectiveness of the pricing decision $\mathbf{P}$, a fitness function is defined. Since satellites aim to maximize their total profit, the fitness function is expressed as

$$F_3(\mathbf{P}, \mathbf{\Phi}) = U^{\text{satellite}} = \sum_{n=1}^{N} U_n^{\text{satellite}}. \tag{45}$$

Given the pricing strategy $\mathbf{P}$, the lower level optimization of $\mathcal{P}_1$ is solved using Alg. 2, while the upper level optimization is handled by the PSO algorithm, which is known for its efficiency and fast convergence [38]. The effectiveness of PSO has been validated in [39], [40]. As $\mathcal{P}_1$ is a nested bilevel optimization problem, a nested bilevel algorithm is developed to dynamically adjust the pricing strategy.

In the PSO algorithm, each particle l_i has a position vector $\mathbf{x}_i$, which represents a potential satellite pricing strategy, and a velocity vector $\mathbf{v}_i$ that guides the update direction. During each iteration, the personal best position $\mathbf{y}_i^{\text{best}}$ is tracked for each particle, while the global best $\mathbf{g}^{\text{best}}$ is selected based on overall fitness. The velocity and position of l_i are updated as follows

$$\begin{aligned} \mathbf{v}_i(t+1) &= w(t)\mathbf{v}_i(t) + c_1(t)\mathbf{z}_1[\mathbf{y}_i^{\text{best}}(t) - \mathbf{x}_i(t)] \\ &\quad + c_2(t)\mathbf{z}_2[\mathbf{g}^{\text{best}}(t) - \mathbf{x}_i(t)], \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \end{aligned} \tag{46}$$

where t is the iteration number, $w \in [0, 1]$ is the inertia weight, c_1 and c_2 represent the cognitive and social learning factors, respectively, and $\mathbf{z}_1, \mathbf{z}_2$ are independent random variables in the range $(0, 1)$. Based on the traditional PSO algorithm, a linear decreasing weight (LDW) strategy and adaptive learning factors are adopted to balance convergence speed and solution accuracy. The inertia weight and learning factors are updated by

$$\begin{aligned} w(t) &= w^{\max} - (w^{\max} - w^{\min}) \cdot \frac{t}{t^{\max}}, \\ c_1(t) &= c_1^{\max} - (c_1^{\max} - c_1^{\min}) \cdot \frac{t}{t^{\max}}, \\ c_2(t) &= c_2^{\min} + (c_1^{\max} - c_1^{\min}) \cdot \frac{t}{t^{\max}}, \end{aligned} \tag{47}$$

where $w^{\max}$ and $w^{\min}$ are the maximum and minimum inertia weights, respectively, and $c_1^{\max}$, $c_1^{\min}$, $c_2^{\max}$, and $c_2^{\min}$ are the maximum and minimum values of the learning factors.

During the particle movement, Alg. 2 calculates the optimal offloading decision matrix $\mathbf{\Phi}_i$ corresponding to the current position $\mathbf{x}_i$, leading to the satellite total profit $F_3(\mathbf{x}_i, \mathbf{\Phi}_i)$. Subsequently, the personal best position $\mathbf{y}_i^{\text{best}}$ is updated as

$$\mathbf{y}_i^{\text{best}}(t+1) = \begin{cases} \mathbf{y}_i^{\text{best}}(t), & \text{if } F_3(\mathbf{x}_i(t+1), \mathbf{\Phi}_i(t+1)) \leq F_3(\mathbf{x}_i(t), \mathbf{\Phi}_i(t)), \\ \mathbf{x}_i(t+1), & \text{otherwise.} \end{cases} \tag{48}$$

Algorithm 3 PSO-MG

$t = 0$;
Randomly initialize particle swarm $\mathbf{L} = \{l_1, l_2, \ldots, l_E\}$.
while $t \leq t^{\max}$ **do**
  Calculate inertia weight and learning factors by (47);
  for each particle $i = 1, 2, \ldots, E$ **do**
    Update position $\mathbf{x}_i(t+1)$ and velocity $\mathbf{v}_i(t+1)$ by (46);
    if $\mathbf{x}_i(t+1)$ exceeds boundary **then**
      $\mathbf{x}_i(t+1) = \mathbf{x}_i^{\text{bound}}(t+1)$
    end if
    Perform Alg. 2 to get offloading decision matrix $\mathbf{\Phi}_i(t+1)$;
    Calculate satellite total profit $F_3(\mathbf{x}_i(t+1), \mathbf{\Phi}_i(t+1))$ by (45);
    Update personal best position $\mathbf{y}_i^{\text{best}}(t+1)$ by (48);
  end for
  Update global best position $\mathbf{g}^{\text{best}}(t+1)$ by (49);
  $t = t + 1$
end while
Set final global best position as $\mathbf{P}$.
return $\mathbf{P}$

The global best position $\mathbf{g}^{\text{best}}$ is updated as

$$\mathbf{g}^{\text{best}}(t+1) = \arg\max_{\mathbf{y}_i^{\text{best}}} F_3(\mathbf{y}_i^{\text{best}}(t+1), \mathbf{\Phi}_{\mathbf{y}_i^{\text{best}}}(t+1)). \quad (49)$$

Since satellite pricing must satisfy the $\mathcal{C}_1$ constraint, boundary handling is required for particle positions. If a particle's updated position exceeds the feasible range, it is adjusted to the nearest boundary, i.e., $\mathbf{x}_i(t+1) = \mathbf{x}_i^{\text{bound}}(t+1)$.

The PSO-MG algorithm is described in Alg. 3. Initially, the positions and velocities of the particle swarm are randomly initialized within the feasible range. In each iteration, the inertia weight and learning factors are dynamically adjusted using the LDW strategy to update particle velocities and positions, while boundary constraints ensure feasibility. Then, Alg. 2 is used to compute the optimal user offloading decision for each particle, and satellite profit is evaluated using equation (45). Personal and global best positions are updated based on profit values. After all iterations, the final global best position yields the optimal pricing strategy.

E. Complexity Analysis

The complexity of Alg. 2 consists of three main phases: initial solution construction, matching request submission, and swap-matching. In the first phase, each user sorts its candidate server list, leading to a worst-case complexity of $\mathcal{O}(M(N+1)\log(N+1))$. In the second phase, each user submits requests to servers, which sort and process them, resulting in a complexity of $\mathcal{O}(NM\log(M))$. The swap-matching phase iteratively checks and swaps matched pairs, requiring $\mathcal{O}(KM^2)$, where K is the maximum number of swap iterations. Since K is a constant, the overall complexity of Alg. 2 is $\mathcal{O}(MN\log(N) + NM\log(M) + M^2)$. In Alg. 3, each particle in the swarm executes Alg. 2 once per iteration. Given that the PSO algorithm runs for t^{max} iterations with E particles, the total complexity of Alg. 3 is $t^{max}E\mathcal{O}(MN\log(N) + NM\log(M) + M^2)$.

TABLE I: SYSTEM PARAMETERS SETTINGS

Parameter	Value	Parameter	Value
N	6	B_m	[1000, 1100] cycle/bit
Q_m	[1, 3] Mbit	$\mathrm{T}_m^{\max}$	1 s
$f_m^{\text{max\_user}}$	1.5 GHz	$f_n^{\text{max\_sat}}$	[10, 50] GHz
R_m^{gnd}	200 Mbps	R_{ISL}	10 Gbps
P_m^{gnd}	4 J/s	P_{ISL}	0.08 J/s
κ_m^{user}	1e-27	κ_n^{sat}	1e-28
η_1	1 per J	η_2	0.1 per J
ξ_1	1 per J	ξ_2	0.1 per J
t^{prop}	0.015 s	λ	5e-5
$P_n^{\min}$	1 per Gcycles	$P_n^{\max}$	50 per Gcycles

VI. Numerical Results

A. Scenario Settings

1) Parameter Settings: In the simulation, ground users are uniformly distributed within a 1000m radius circular region, outside the base station coverage area. The access satellite S_1 is connected to five satellites $S_2 \sim S_6$ via ISLs. The main system parameters are derived from [29] and [41], and summarized in Table I. Specifically, each user's task size is uniformly distributed between [1M, 3M] bits, and the task processing density is uniformly distributed between [1000, 1100] cycles/bit. Each satellite's maximum computing resource ranges uniformly from [10G, 50G] Hz. All simulations were conducted using MATLAB.

2) Competitors: To evaluate the proposed PSO-MG algorithm, we compare it with six algorithms, including the method proposed in [41], three nested optimization algorithms, and two benchmark algorithms.

- DE-VNS: It is a nested bilevel optimization algorithm, where differential evolution (DE) is employed in the upper level and variable neighborhood search (VNS) is applied in the lower level. Moreover, a greedy method is developed to construct a good initial solution for VNS.
- GA-MG: This algorithm employs a genetic algorithm (GA) in the upper level to determine pricing strategies for satellites' computing resources, and utilizes Alg. 2 in the lower level to make offloading decisions for users.
- Simplex-MG: A nested optimization algorithm that employs the Nelder-Mead simplex method in the upper level and utilizes Alg. 2 in the lower level.
- PSO-ACS: It is composed of two heuristic algorithms which uses PSO algorithm at the upper level and ant colony system (ACS) algorithm at the lower level.
- Random: Users randomly choose local computation or send offloading requests to the access satellite. The access satellite randomly sets prices and accepts requests until its maximum resource threshold is reached. Users rejected by the access satellite can opt for local computation again if their candidate server sets include local servers.
- ISL-assisted Random: Each user selects a server from its candidate server set and sends an offloading request. Each

satellite accepts requests at random until its maximum computing resource threshold is reached, rejecting the remaining users. Rejected users send requests randomly to the next server in their candidate sets until a successful match is made or all servers have been attempted.

To evaluate the algorithms' performance, three key metrics are considered: satellite total profit, user task completion rate, and algorithm runtime.

B. Performance Evaluation

Three sets of comparative experiments are conducted to evaluate the convergence of the algorithms and to examine the impact of varying user and satellite numbers on key performance metrics.

1) Convergence of Algorithms: The convergence performance of the five algorithms is illustrated in Fig. 5. All algorithms converge to stable values. Specifically, GA-MG achieves convergence after 18 iterations, while DE-VNS, PSO-MG, PSO-ACS, and Simplex-MG converge after 24, 32, 61, and 75 iterations, respectively. Upon convergence, PSO-MG outperforms the other four algorithms, with performance improvements of 5.5%, 16.4%, 21.7%, and 55.9%, respectively.

2) Impact of Number of Users: As depicted in Fig. 6, we compare the total satellite profits achieved by seven algorithms under a fixed number of satellites ($N = 6$). Each algorithm is executed 30 times per instance, and the average results are reported. The satellite profits obtained by all algorithms initially increase and then plateau as M increases. This is because, with N fixed, a growing number of users provides more opportunities for satellites to offer computing services. However, once the maximum resource threshold is reached ($M \geq 110$), profits no longer improve. The proposed PSO-MG algorithm yields the highest total profit, benefiting from Alg. 2, which constructs preference strategies based on fitness functions. In contrast, the heuristic ACS algorithm starts with a random solution and struggles to explore the expanded search space as M increases. In bilevel optimization, ACS's lower level solution may deviate significantly from the optimal, affecting upper level performance accuracy.

The user task completion rate is defined as the ratio of users who successfully complete their tasks to the total number of users. Fig. 7 illustrates how this rate varies with the number of users. Most algorithm curves remain stable at first but drop sharply as the user count increases, with the exception of the Random algorithm. The proposed PSO-MG algorithm consistently achieves the highest completion rate. When satellite resource thresholds are reached ($M \geq 110$), the completion rate of the ISL-assisted Random algorithm falls below that of the proposed algorithm, indicating that user prioritization based on sorting strategies effectively enhances task completion.

The average running times of the five nested algorithms for each instance are shown in Fig. 8. All algorithms exhibit increasing runtimes as the number of users M grows. Among them, Simplex-MG achieves the shortest execution time because each iteration involves lightweight computations with minimal complexity. The DE-VNS, GA-MG, and PSO-MG algorithms show comparable runtimes, whereas PSO-ACS exhibits exponential growth. For instance, when $M = 100$, the execution times of Simplex-MG, DE-VNS, GA-MG, and PSO-MG are 13, 31, 44, and 56 minutes, respectively, while PSO-ACS requires up to 14 hours.

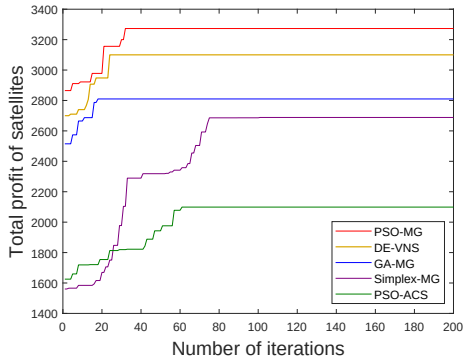

Fig. 5: Iteration comparison of satellites' profit for M = 30.

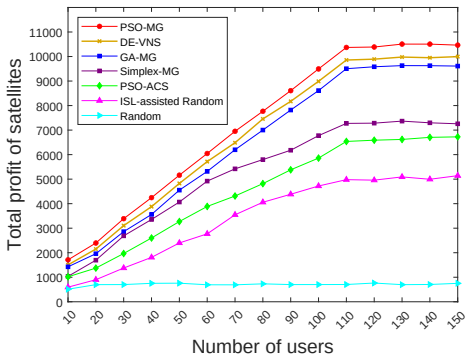

Fig. 6: Comparison of satellites' profit with different numbers of users.

3) Impact of Number of Satellites: In Fig. 9, we examine the impact of the number of satellites on total profit under scenarios where user task data volumes are relatively large and satellite computing resource thresholds are low, assuming a fixed number of users ($M = 50$). Our algorithm outperforms the others in six instances. All algorithm curves initially increase and then plateau, except for the Random algorithm. When the number of satellites is small ($N < 9$), increasing N improves task completion and thus enhances satellite profits. Once all user tasks are completed, the total profit stabilizes. Fig. 10 further shows that the user task completion rate increases with N, with our algorithm consistently achieving the highest completion rate, ultimately reaching 1.

VII. CONCLUSION

This paper proposes a bilevel optimization framework to determine the pricing of satellite computing resources while ensuring fair resource allocation to meet user task requirements. The upper level optimizes satellite profit by adjusting resource prices, while the lower level manages user task offloading and resource demands based on these prices. To address this NP-hard problem, we transform it by leveraging the relationship between offloading modes and resource demands, pruning servers to narrow the search space, and

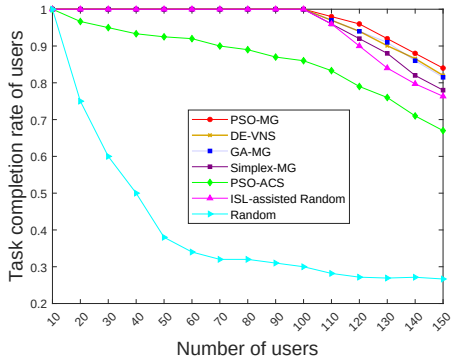

Fig. 7: Comparison of users' task completion rate with different numbers of users.

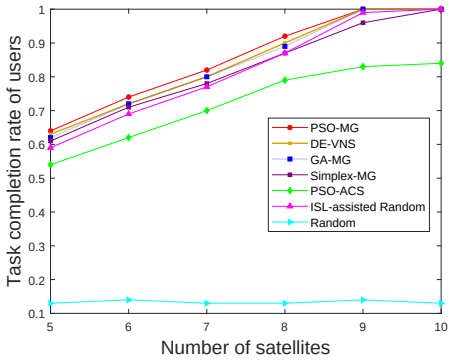

Fig. 10: Comparison of users' task completion rate with different numbers of satellites.

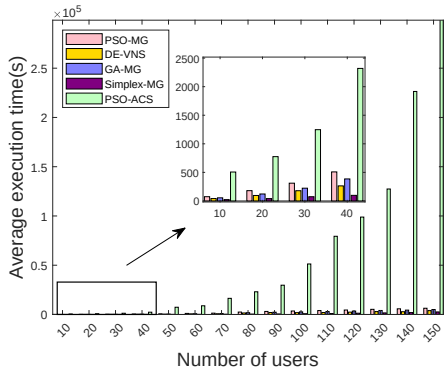

Fig. 8: Comparison of average execution time with different numbers of users.

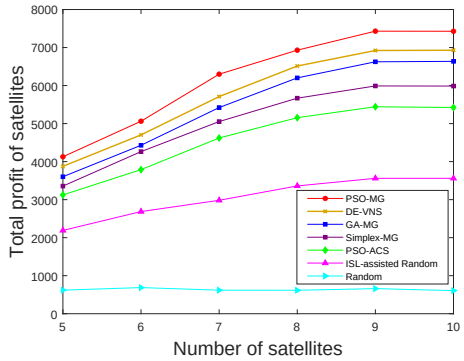

Fig. 9: Comparison of satellites' profit with different numbers of satellites.

devising a nested bilevel optimization algorithm (PSO-MG). Experimental results demonstrate that PSO-MG outperforms both nested heuristic and baseline algorithms in terms of satellite profit and task completion rate. However, the current model does not fully consider complex orbital dynamics, such as satellite movement and dynamic connectivity. Future work will focus on developing a real-time adaptive pricing algorithm to handle satellite mobility and resource demand variations.

REFERENCES

[1] L. P. Qian, Y. Wu, N. Yu, D. Wang, F. Jiang, and W. Jia, "Energy-efficient multi-access mobile edge computing with secrecy provisioning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 237–252, 2023.

[2] M. Cao, Q. Wang, and Q. Wang, "Federated learning in smart home: A dynamic contract-based incentive approach with task preferences," *Computer Networks*, 2024.

[3] T. de Cola and I. Bisio, "Qos optimisation of embb services in converged 5g-satellite networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 098–12 110, 2020.

[4] Z. Lin, M. Lin, T. de Cola, J.-B. Wang, W.-P. Zhu, and J. Cheng, "Supporting iot with rate-splitting multiple access in satellite and aerial-integrated networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 123–11 134, 2021.

[5] Z. Zhang, Y. Li, C. Huang, Q. Guo, L. Liu, C. Yuen, and Y. L. Guan, "User activity detection and channel estimation for grant-free random access in leo satellite-enabled internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8811–8825, 2020.

[6] B. Denby and B. Lucia, "Orbital edge computing: Machine inference in space," *IEEE Computer Architecture Letters*, vol. 18, no. 1, pp. 59–62, 2019.

[7] H.-C. Chao, D. E. Comer, and O. Kao, "Space and terrestrial integrated networks: Emerging research advances, prospects, and challenges," *IEEE Network*, vol. 33, no. 1, pp. 6–7, 2019.

[8] M. Giordani and M. Zorzi, "Non-terrestrial networks in the 6g era: Challenges and opportunities," *IEEE Network*, vol. 35, no. 2, pp. 244–251, 2021.

[9] J. Cui, S. X. Ng, D. Liu, J. Zhang, A. Nallanathan, and L. Hanzo, "Multiobjective optimization for integrated ground-air-space networks: Current research and future challenges," *IEEE Vehicular Technology Magazine*, vol. 16, no. 3, pp. 88–98, 2021.

[10] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.

[11] S. Wang, Q. Li, M. Xu, X. Ma, A. Zhou, and Q. Sun, "Tiansuan constellation: An open research platform," in *2021 IEEE International Conference on Edge Computing (EDGE)*, 2021, pp. 94–101.

[12] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi, "Broadband leo satellite communications: Architectures and key technologies," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 55–61, 2019.

[13] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Network*, vol. 34, no. 3, pp. 224–231, 2020.

[14] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (sagin)," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 276–289, 2022.

[15] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and G. Y. Li, "Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1362–1377, 2022.

[16] C. E. Gonzalez, A. Bergel, and M. A. Diaz, "Nanosatellite constellation control framework using evolutionary contact plan design," in *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2021, pp. 85–92.

[17] S. Kassing, D. Bhattacherjee, A. B. guas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," in *IMC '20: ACM Internet Measurement Conference*, 2020.

[18] Z. Zhai, Q. Wu, S. Yu, R. Li, F. Zhang, and X. Chen, "Fedleo: An offloading-assisted decentralized federated learning framework for low earth orbit satellite networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5260–5279, 2024.
[19] G. Cui, P. Duan, L. Xu, and W. Wang, "Latency optimization for hybrid geo–leo satellite-assisted iot networks," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6286–6297, 2023.
[20] X. Chen, Y. Zhou, L. Yang, and L. Lv, "User satisfaction oriented resource allocation for fog computing: A mixed-task paradigm," *IEEE Transactions on Communications*, vol. 68, no. 10, pp. 6470–6482, 2020.
[21] X. Cao, B. Yang, Y. Shen, C. Yuen, Y. Zhang, Z. Han, H. V. Poor, and L. Hanzo, "Edge-assisted multi-layer offloading optimization of leo satellite-terrestrial integrated networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 381–398, 2023.
[22] C. Huang, G. Chen, P. Xiao, Y. Xiao, Z. Han, and J. A. Chambers, "Joint offloading and resource allocation for hybrid cloud and edge computing in sagins: A decision assisted hybrid action space deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2024.
[23] S. Wang and Q. Li, "Satellite computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 514–22 529, 2023.
[24] Q. Li, S. Wang, X. Ma, A. Zhou, and F. Yang, "Towards sustainable satellite edge computing," in *2021 IEEE International Conference on Edge Computing (EDGE)*, 2021, pp. 1–8.
[25] NASA. (2023) Thermal control. Accessed: 2023-07-30. [Online]. Available: https://www.nasa.gov/smallsat-institute/sst-soa/thermal-control/
[26] R. Deng, B. Di, S. Chen, S. Sun, and L. Song, "Ultra-dense leo satellite offloading for terrestrial networks: How much to pay the satellite operator?" *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6240–6254, 2020.
[27] F. Li, K.-Y. Lam, X. Liu, J. Wang, K. Zhao, and L. Wang, "Joint pricing and power allocation for multibeam satellite systems with dynamic game model," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2398–2408, 2018.
[28] J. Zhou, Q. Yang, L. Zhao, H. Dai, and F. Xiao, "Mobility-aware computation offloading in satellite edge computing networks," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2024.
[29] Y. Zhang, C. Chen, L. Liu, D. Lan, H. Jiang, and S. Wan, "Aerial edge computing on orbit: A task offloading and allocation scheme," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 275–285, 2023.
[30] N. Patrizi, G. Fragkos, K. Ortiz, M. Oishi, and E. E. Tsiropoulou, "A uav-enabled dynamic multi-target tracking and sensing framework," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.
[31] X. Zhang, X. Qin, B. Qian, T. Ma, and H. Zhou, "Joint mode selection and dynamic pricing in ultra dense leo integrated satellite-terrestrial networks," in *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, 2022, pp. 1090–1094.
[32] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.
[33] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.
[34] T. Fang, D. Wu, J. Chen, and D. Liu, "Cooperative task offloading and content delivery for heterogeneous demands: A matching game-theoretic approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 1092–1103, 2022.
[35] Y. Yang, Y. Hui, N. Cheng, R. Sun, M. Tian, and C. Li, "Vehicle digital twins in space-air-ground integrated networks: A game-based migration scheme," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, 2023, pp. 1–6.
[36] J. Xia, G. Cheng, D. Guo, and X. Zhou, "A qoe-aware service-enhancement strategy for edge artificial intelligence applications," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9494–9506, 2020.
[37] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer effects and stability in matching markets," *arXiv e-prints*, 2011.
[38] M. Xie, Y. Bai, M. Huang, Y. Deng, and Z. Hu, "Energy- and time-aware data acquisition for mobile robots using mixed cognition particle swarm optimization," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7734–7750, 2020.
[39] M. Huang, V. C. M. Leung, A. Liu, and N. N. Xiong, "Tma-dpso: Towards efficient multi-task allocation with time constraints for next generation multiple access," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1652–1666, 2022.
[40] H. D. Chantre and N. L. S. da Fonseca, "Multi-objective optimization for edge device placement and reliable broadcasting in 5g nfv-based small cell networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2304–2317, 2018.
[41] P.-Q. Huang, Y. Wang, and K. Wang, "A divide-and-conquer bilevel optimization algorithm for jointly pricing computing resources and energy in wireless powered mec," *IEEE Transactions on Cybernetics*, vol. 52, no. 11, pp. 12 099–12 111, 2022.

Xinyu Wang is currently pursuing the M.S. degree with the Laboratory of Big-data, School of Mathematics, Hefei University of Technology, Hefei. Her research interests include edge computing and satellite-terrestrial integrated networks.

Qi Wang received the Ph.D. degree in Computer Science from Hefei University of Technology, Hefei, China, in 2010. She was a visiting scholar at Temple University between 2014 and 2015. She is an associate professor in the Department of Mathematics at Hefei University of Technology. Her research interests include edge computing, delay-tolerant networks, and network coding.

Qingshan Wang received his Ph.D. degree in Computer Science from the University of Science and Technology of China (USTC), Hefei, China, in 2007. He was a visiting scholar at Cornell University between 2009 and 2010. He is a professor in the Department of Mathematics at Hefei University of Technology. His research interests include human action recognition, delay-tolerant networks, ad hoc network protocol design, and network coding.

Manxia Cao received the M.S. degree from the School of Mathematics, Hefei University of Technology, Hefei, China, in 2021. She is currently pursuing the Ph.D. degree with the Laboratory of Big-data, School of Mathematics, Hefei University of Technology, Hefei. Her research interests include edge computing and federated learning.