Multi-User Task Offloading in UAV-Assisted LEO Satellite Edge Computing: A Game-Theoretic Approach

Ying Chen, *Senior Member, IEEE*, Jie Zhao, Yuan Wu, *Senior Member, IEEE*, Jiwei Huang, *Senior Member, IEEE*, and Xuemin Sherman Shen, *Fellow, IEEE*

***Abstract*—Unmanned Aerial Vehicle (UAV)-assisted Low Earth Orbit (LEO) satellite edge computing (ULSE) networks can address the challenge communications issues in areas with harsh terrain and achieve global wireless coverage to provide services for mobile user devices (MUDs). This paper studies the LEO-UAV task offloading problem where MUDs compete for limited resources in the ULSE networks. We formulate the optimization problem with the goal of minimizing the cost of all MUDs while meeting resource constraint and satellite coverage time constraint. We first theoretically prove that this problem is NP-hard. We then reformulate the problem as a LEO-UAV task offloading game (LUTO-Game), and show that there is at least one Nash equilibrium solution for the LUTO-Game. We propose a joint UAV and LEO satellite task offloading (JULTO) algorithm to obtain the Nash equilibrium offloading strategy, and analyze the performance of the worst-case offloading strategy obtained by the JULTO algorithm. Finally, extensive experiments, including convergence analysis and comparison experiments, are carried out to validate the effectiveness of our JULTO algorithm.**

***Index Terms*—Game model, LEO satellite, Nash equilibrium, task offloading, UAV.**

I. Introduction

With the rapid development of beyond 5G/6G communication systems, the number of mobile user devices (MUDs) such as smartphones and tablets has surged. More and more computation-intensive and latency-sensitive applications (such as natural language processing, virtual reality, face recognition, etc.) are running on MUDs [1]. However, constrained by CPU and battery capacity, most MUDs cannot effectively handle computationally intensive applications all by themselves [2], [3], [4]. The framework of mobile edge computing (MEC) [5], [6], [7], [8] is considered as a viable solution, i.e., edge servers with computing resources are placed on network access points close to MUDs [9]. Then, MUDs can transmit task data to edge servers, thereby utilizing the computing resources of edge servers to process computing tasks [10], [11]. This reduces the application processing latency of MUDs, thereby improving users' quality of experience (QoE) [12], [13].

Traditional MEC usually utilizes ground base stations as network access points, which however may not meet the ubiquitous connection requirements [14], [15]. On the one hand, it is difficult for ground network access points to completely cover some complex terrains, such as oceans, deserts, and remote mountainous areas. On the other hand, ground base stations are vulnerable to damage from natural disasters such as earthquakes, hurricanes, and tsunamis, resulting in communication interruptions. In recent years, Unmanned Aerial Vehicles (UAVs) have become popular because of their flexibility and low cost [16], [17], [18]. Low-altitude UAVs can be regarded as base stations to improve the performance of network systems [19], [20]. In addition, with the development of space communication networks, satellite technology has developed rapidly. The on-ground devices based on Low Earth Orbit (LEO) satellites can obtain better signal strength and lower latency, which has received extensive attention [21], [22], [23]. Therefore, deploying edge servers on UAV and LEO satellites can effectively solve communication and computing problems in areas with complex terrain and harsh environments which has drawn extensive attention from both academia and industry.

However, the problem of task offloading in ULSE network systems faces challenges. First, there are heterogeneous resources in the ULSE network, and the resources of UAVs and LEO satellites are limited. MUDs in the system need to compete for limited system resources to process tasks. Communication and computing resource allocation strategies face the challenges brought by complex and heterogeneous network environments and user competition. Second, in addition to resource constraints, the LEO satellite coverage time constraint should

Received 25 January 2024; revised 25 August 2024; accepted 12 September 2024. Date of publication 24 September 2024; date of current version 4 December 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62472039, in part by Beijing Natural Science Foundation under Grant L232050, in part by the Project of Cultivation for young top-motch Talents of Beijing Municipal Institutions under Grant BPHR202203225, in part by the Young Elite Scientists Sponsorship Program by BAST under Grant BYESS2023031, and in part by the Science and Technology Development Fund of Macau SAR under Grant FDCT 0158/2022/A. Recommended for acceptance by M. Chen. *(Corresponding authors: Yuan Wu; Jiwei Huang.)*

Ying Chen and Jie Zhao are with the Beijing Information Science and Technology University, Beijing 100101, China (e-mail: chenying@bistu.edu.cn; zhaojie99723@bistu.edu.cn).

Yuan Wu is with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau 999078, China (e-mail: yuanwu@um.edu.mo).

Jiwei Huang is with the China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn).

Xuemin Sherman Shen is with the University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TMC.2024.3465591

also be satisfied. For different MUDs, the upper bound of the coverage time with different LEO satellites are different. How to calculate the LEO satellite coverage time with different offloading decisions, and obtain the decisions that satisfy both the coverage time and resource constraints is a challenge. Besides, MUD has individual rationality, that is, it will not sacrifice its own benefits to reduce the cost of other MUDs. Therefore, achieving a balanced offloading strategy for all MUDs while minimizing the cost of all MUDs in the entire system is also a challenging problem. Finally, the solution space size of the problem increases with the network system scale, and the complexity of finding the optimal offloading strategy increases exponentially when the number of MUDs, UAVs, or LEO satellites increases.

This paper studies the multi-user task offloading problem in the ULSE networks. The optimization goal is to minimize the cost of MUDs. The problem is reformulated as a LEO-UAV task offloading game (LUTO-Game) model, and it is theoretically proved that there is at least one Nash equilibrium solution for the LUTO-Game. Then, we propose a joint UAV and LEO satellite task offloading (JULTO) algorithm to obtain the task offloading strategy and realize the balance of multi-MUD decision-making. Finally, we perform both theoretical analysis and experiments to evaluate the performance of the JULTO algorithm. The contributions of this paper are summarized as follows.

- We propose a LEO-UAV task offloading framework which utilizes the edge computing resources of LEO satellites and UAVs. In this framework, satellites and UAVs with edge servers can handle the tasks of MUDs. The offloading decisions of MUDs are optimized with the objective of minimizing the total cost of MUDs. When MUDs offload tasks, they need to compete for limited transmission resources. For UAV edge computing, UAVs transmit energy to MUDs by applying wireless power transmission technology, and the resource constraint is considered. For LEO satellite edge computing, in addition to the restriction of resources, satellite coverage time constraint is also considered. The coverage time model for satellites in different situations is discussed.
- We prove that the formulated task offloading problem is NP-hard. Since MUDs are selfish, the multi-device and multi-server task offloading problem in the ULSE system is reformulated as the LUTO-Game model. The MUDs in the system are game participants, focusing on reducing their own costs. The desirable offloading strategy of the problem is defined as the Nash equilibrium solution of the LUTO-Game. Then, by defining a task offloading rule, the potential function is given, and the LUTO-Game is proved theoretically a potential game. Therefore, it can be determined that LUTO-Game has at least one feasible Nash equilibrium solution.
- The JULTO algorithm is proposed to obtain the Nash equilibrium strategy of the task offloading problem. The JULTO algorithm is implemented in a distributed manner, and the MUDs make offloading decisions in parallel. On the basis of the proposed JULTO algorithm, the price of anarchy (PoA) is defined, which is the ratio of the worst cost obtained by the Nash equilibrium unloading strategy to the cost obtained by the centralized optimal strategy. According to the PoA, we theoretically analyze the performance of the JULTO algorithm.
- Extensive experiments are carried out to evaluate the JULTO algorithm. The experiment results show that the JULTO algorithm can converge after a limited iteration number. When the scale of the problem expands, the problem's solution space experiences exponential growth, and the growth rate of the iteration number required for the game to reach the Nash equilibrium state is lower than the linear speed. In addition, comparison experiments are conducted. The results show that the cost for the JULTO algorithm is lower than that obtained by other algorithms, which verifies the performance superiority of JULTO algorithm.

The subsequent sections of this paper are structured as follows. In Section II, related works are presented. Section III describes the ULSE system model, and formulates the task offloading problem. In Section IV, the problem is reformulated as the LUTO-Game model. Then, a theoretical analysis of the LUTO-Game is conducted. Section V proposes the distributed JULTO algorithm, and analyzes theoretically the JULTO's performance. The experimental evaluation is given in Section VI. Finally, Section VII presents the paper's conclusion.

II. Related Work

As a flexible and efficient mobile aerial platform, UAVs have attracted widespread attention in both civilian and military fields. [24] studied the problem of collecting data in the UAV wireless networks, using the maximum age of information (AoI) to measure information freshness. On this basis, [25] proposed a combined optimization objective of minimizing the maximum AoI. Liu et al. proposed an iteration-based trajectory planning and sensor node association strategy to obtain the optimal AoI solution. With the development of MEC, some related works have studied UAV computing architecture deploying edge servers. In [26], by jointly optimizing UAV position, transmission bandwidth, and CPU frequency, all devices' computation delay was minimized. The joint optimization problem was solved by applying the successive convex approximation technique. In [27], Han et al. adopted optimal transportation theory and classical particle swarm optimization algorithm to conduct joint optimization of UAV deployment and user association, and finally realized the minimization of average delay. [28] jointly optimized resource allocation, equipment scheduling, and UAV trajectories. Han et al. proposed an iterative algorithm based on alternating optimization and convex optimization.

Because of the global coverage characteristics of LEO satellites, they can break through the geographical restrictions of communication and become an important part of future communication systems. For example, in [29], an ultra-dense LEO satellite network was considered. A pricing mechanism was designed based on Stackelberg game to incentivize ground and satellite operators' data reloading. In addition, some studies have considered LEO satellites combined with MEC. In [22], Li et al. studied the MEC deployed on LEO satellites. Service

scheduling and placement were optimized using mixed integer linear programming (MILP). However, they did not consider the coverage of LEO satellites. In our model, we further consider that users need to calculate the corresponding satellite coverage time and perform task offloading on the premise of meeting time constraints. In [30], the joint optimization problem of LEO satellite MEC network was studied. An algorithm based on Lagrangian Dual Decomposition (LDD) was proposed. [31] proposed a satellite MEC architecture based on federated learning and used the blockchain framework to achieve data privacy and security protection. [15] studied the task offloading problem based on LEO satellite MEC, and formulated the optimization problem as a partially observable Markov decision process. A multi-agent algorithm was proposed to achieve resource allocation in a collaborative manner. [32] considered a satellite-ground network using dual edge computing, with the optimization goal of cost minimization. A double-edged computing offloading algorithm was proposed to achieve computing resource allocation. [33] established a system model for satellite edge computing task offloading, and used queuing theory and game theory to optimize utility function.

The above works did not consider the resource limitations of satellite edge computing, i.e., it may not be sufficient to meet the needs of all types of user devices. Therefore, it is an effective method to combine UAVs and LEO satellites to form a Space-Air system. The Space-Air system can serve as a supplement to the ground network, overcoming the impact of harsh environments on communication conditions and providing various computation resources and reliable services for user devices. Therefore, we combine UAV edge computing and LEO satellite network to form a heterogeneous integrated air-ground-space network, aiming to make up for the shortcomings and limitations of a single type of edge server. Some similar works exist. For example, [2] studied the edge computing architecture of the Space-Air-Ground Integrated Network (SAGIN), jointly optimized task scheduling and resource allocation, and used learning-based methods to obtain the optimal flow strategy and UAV flight strategy, thereby minimizing system costs. [34] studied the task offloading problem in SAGIN. The optimization goal was to minimize UAVs' energy consumption and maximize the number of tasks that meet the delay constraints. A solution method based on reinforcement learning was proposed to obtain the optimal offloading solution. Different from [2] and [34], we consider the coverage time constraints and computing resource constraints of different LEO satellites, as well as user competition for resources. A distributed method is proposed to obtain offloading decisions, which achieves multi-user optimization while balancing the performance and computation complexity of the method.

III. System Model and Problem Formulation

A. Network Model

Fig. 1 shows the ULSE framework. There are N MUDs ($\mathbf{D} = \{d_1, \ldots, d_N\}$), M_1 UAVs ($\mathbf{U} = \{u_1, \ldots, u_{M_1}\}$), and M_2 LEO satellites ($\mathbf{S} = \{s_1, \ldots, s_{M_2}\}$) in orbit. Each MUD $d_i \in \mathbf{D}$ has one computation task $H_i = (B_i, C_i)$ that needs to be processed. C_i is the CPU cycles number required to complete H_i. B_i is the size of H_i in bits. Ground base stations may not be available in areas with harsh terrain or disasters. In the ULSE networks, UAVs and LEO satellites are equipped with edge servers (ESs) that can process the tasks of MUDs. There are c^u and c^L wireless channels for UAVs and LEO satellites, respectively. Table I summarizes the paper's main notations. Compared with MUDs, ESs have a larger computing capacity. Therefore, MUDs can transmit computation-intensive task data to ESs of UAVs or LEO satellites for processing. When channel resources are insufficient, MUDs have to complete their tasks locally. MUDs can choose an appropriate offloading decision to process computation tasks according to their own demand.

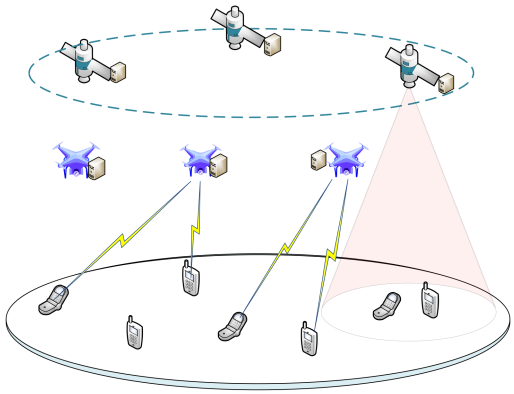

Fig. 1. An ULSE scenario example.

The offloading decision of MUD d_i is represented by $o_i \in \{(0,0,0) \bigcup (a_i, b_i, c_i)\}$. When MUD d_i executes the task H_i locally, its decision is $o_i = (0,0,0)$. $o_i = (a_i, b_i, c_i)$ represents that MUD d_i offloads the task H_i to ESs. a_i represents the offloading way selected by MUD d_i. Specifically, if MUD d_i offloads the task to ESs of UAVs, $a_i = 1$. If MUD d_i offloads the task to ESs of LEO satellites, $a_i = 2$. b_i represents the UAV or LEO satellite that is selected by MUD d_i for task processing. c_i represents the wireless channel that is selected by MUD d_i for task data transmission. When MUD d_i offloads the task to UAVs, i.e., $a_i = 1$, then $b_i \in \{1, \ldots, M_1\}$ and $c_i \in \{1, \ldots, c^u\}$. When MUD d_i offloads the task to LEO satellites, i.e., $a_i = 2$, then $b_i \in \{1, \ldots, M_2\}$ and $c_i \in \{1, \ldots, c^L\}$. In addition, the set of offloading decisions of all MUDs is called the offloading strategy denoted by $o = \{o_1, \ldots, o_N\}$.

In traditional mobile edge computing, terrestrial base stations are used as network access points. When ground base stations are damaged by natural disasters such as tsunamis, hurricanes, and earthquakes, they may not meet the communication requirements of ground devices. Therefore, in this work, in order to cope with the communication and computing challenges in areas with complex terrain and harsh environments, a UAV-assisted LEO satellite edge computing network framework is proposed.

B. Service Coverage Model

1) UAV Service Coverage: In UAV edge computing, each UAV serves a certain range of services, and only MUDs within

TABLE I
KEY NOTATIONS

Notations	Definitions
N	the number of MUDs
d_i	the ith MUD
$\mathbf{D}$	the MUD set
$\mathbf{U}$	the UAV set
H_i	the MUD d_i's task
B_i	the task H_i's size
M_1	the UAV number
c^u	the channel number for UAVs
$\mathbf{S}$	the LEO satellite set
M_2	the LEO satellite number
c^L	the channel number for LEO satellites
f_i^{local}	the computing capability of d_i
o_i	MUD d_i's offloading decision, $o_i \in \{(0,0,0) \bigcup (a_i, b_i, c_i)\}$
L_1	the height of a low-orbit satellite's orbit above the ground
L	the distance from the MUD to the LEO satellite
L_2	the earth's radius
L^{arc}	the maximum arc length of LEO satellite coverage
T_i^L	the longest communication time between MUD and LEO satellite
v_L	the velocity of the LEO satellite
Dr_i	MUD d_i's data rate
W_{b_i,c_i}	the channel bandwidth for MUD d_i
σ^2	the background noise
$g_i^{b_i,c_i}$	the channel gain for MUD d_i
f_{i,b_i}^{LEO}	the computing capability allocated by the ES of LEO satellite s_{b_i} to MUD d_i
$K_{o_{-i}}(o_i)$	the cost of MUD d_i

the range of services can offload the computing task to the UAV edge server for processing. In Fig. 2, (x_j^{uav}, y_j^{uav}) and h_j^{uav} represent the coordinates and flight altitude of the UAV $u_j \in \mathbf{U}$, respectively. R^{uav} represents the service coverage radius of the UAV $u_j \in \mathbf{U}$. (x_i^{MUD}, y_i^{MUD}) represents the coordinates of the MUD $d_i \in \mathbf{D}$. Therefore, if MUD d_i offloads the task H_i to the edge server of UAV u_j for processing, constraint (1) should be satisfied.

$$\sqrt{(x_j^{uav} - x_i^{MUD})^2 + (y_j^{uav} - y_i^{MUD})^2} \leq R^{uav}. \quad (1)$$

2) LEO Satellite Coverage Time: Generally, LEO satellites move continuously in orbit. MUDs and LEO satellites cannot guarantee communication at any time, and data transmission is limited by satellite coverage time. Specifically, in the ULSE networks, the position of the LEO satellite is variable. Therefore, there are constraints in communicating with LEO satellites.

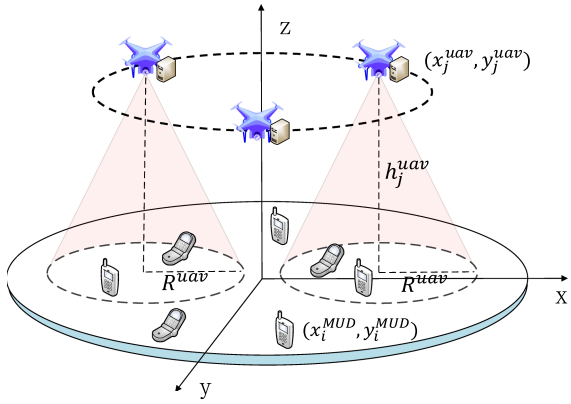

Fig. 2. The spatial configuration of the MUDs and UAVs.

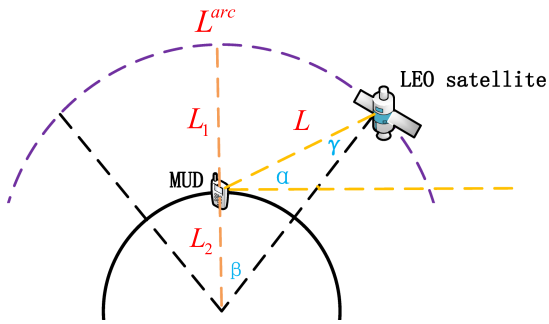

Fig. 3. The spatial configuration of the MUD and LEO satellite.

Generally speaking, MUDs can only transmit data within the coverage area of satellite signals.

In Fig. 3, L_1 is the LEO satellite's orbit height, L_2 the earth's radius, and α the minimum elevation angle from the MUD to the LEO satellite. β is one-half of the LEO satellite coverage area's geocentric angle. According to the Law of Sines, there is

$$\begin{aligned} &\frac{\sin\gamma}{L_2} = \frac{\sin(\alpha + \frac{\pi}{2})}{L_2 + L_1} \\ &\Rightarrow \frac{\cos(\beta+\alpha)}{L_2} = \frac{\cos\alpha}{L_2 + L_1} \\ &\Rightarrow \cos(\beta+\alpha) = \frac{L_2}{L_2 + L_1}\cos\alpha \\ &\Rightarrow \beta = \arccos\left(\frac{L_2}{L_2 + L_1}\cos\alpha\right) - \alpha. \end{aligned} \quad (2)$$

L is the distance between the MUD and LEO satellites, there exists

$$L = \frac{L_2 + L_1}{\cos\alpha}\sin\beta. \quad (3)$$

The maximum arc length L^{arc} of LEO satellite coverage is

$$L^{arc} = 2\beta(L_1 + L_2). \quad (4)$$

For multiple MUDs and LEO satellites, there may be different elevation angles $\alpha_{i,b_i}, d_i \in \mathbf{D}, s_{b_i} \in \mathbf{S}$. There are two cases, as follows:

Case 1: $\alpha \leq \alpha_{i,b_i} \leq \frac{\pi}{2}$.
According to (2), there exists

$$\beta_{i,b_i} = \arccos\left(\frac{L_2}{L_2+L_1}\cos\alpha_{i,b_i}\right) - \alpha_{i,b_i}. \tag{5}$$

Therefore, the remaining coverage range is

$$L^{re}_{i,b_i} = (\beta + \beta_{i,b_i})(L_1+L_2). \tag{6}$$

Case 2: $\frac{\pi}{2} < \alpha_{i,b_i} < \pi - \alpha$.
Similar to Case 1, there are

$$\beta_{i,b_i} = \arccos\left(\frac{L_2}{L_2+L_1}\cos(\pi-\alpha_{i,b_i})\right) - (\pi - \alpha_{i,b_i}), \tag{7}$$

$$L^{re}_{i,b_i} = (\beta - \beta_{i,b_i})(L_1+L_2). \tag{8}$$

v_L denotes the speed of the LEO satellites. Based on the analysis of Case 1 and Case 2, the maximum communication time of MUD d_i and LEO satellite b_i is

$$T^L_i = \frac{L^{re}_{i,b_i}}{v_L}. \tag{9}$$

C. Communication Model

In the ULSE networks, UAVs and LEO satellites provide multiple available channels for MUDs to transmit task data. Each MUD can only transmit data to an ES through a wireless channel. The MUDs that select the same channel interfere with each other. Similar to the related works [35], [36], [37], the position change of the UAV and the energy consumption associated with UAV flight and hovering are beyond the scope of this paper. We investigate the offloading decision in the current state. In the optimization problem, the goal is to minimize the cost for all MUDs.

When MUDs communicate with UAVs or LEO satellites, the background noise variance is represented by σ^2. Similar to [35], [38], [39], we consider that the Signal-to-Interference-plus-Noise Ratio (SINR) of MUD communicating with a UAV or LEO satellite is

$$\Gamma_i = \frac{p_i g_i^{b_i,c_i}}{\sigma^2 + \sum_{d_l \neq d_i \cap o_l = o_i} p_l g_l^{b_l,c_l}}. \tag{10}$$

p_i is the transmission power of d_i, $g_i^{b_i,c_i}$ is the channel gain between d_i and UAV u_{b_i} or LEO s_{b_i} on channel c_i. The MUD d_i's data rate is

$$Dr_i = W_{b_i,c_i}\log_2(1+\Gamma_i), \tag{11}$$

where W_{b_i,c_i} is the channel c_i's bandwidth. When more MUDs select the same channel, the data rate of MUDs decreases.

The MUD communicates with UAV through line-of-sight links (LoS) and non-line-of-sight links (NLoS). As shown in Fig. 2, (x_i^{MUD}, y_i^{MUD}) is the coordinate position of MUD d_i. (x_j^{uav}, y_j^{uav}) is the horizontal coordinate position of UAV u_j, and h_j^{uav} is the hovering height of UAV u_j. Referring to [40], when the MUD d_i communicates with the UAV u_j, the path loss is as follows:

$$L^{path}_{i,j} = 20\log_{10}\left(\frac{4\pi f_c||P_{d_i} - P_{u_j}||}{c}\right) + LoS(\theta_{i,j})\eta_i^{LoS} + (1 - LoS(\theta_{i,j}))\eta_i^{NLoS},$$

where f_c represents the carrier frequency, v represents the speed of light. η_i^{LoS} and η_i^{NLoS} indicate the path loss of LoS and NLoS, respectively. According to [41], $LoS(\theta_{i,j})$ indicates the LoS probability when the MUD d_i communicates with UAV u_j.

$$LoS(\theta_{i,j}) = \frac{1}{1+\mu_1^{-\mu_2(\theta_{i,j}-\mu_1)}}$$

$$\theta_{i,j} = \frac{180}{\pi}\arctan\left(\frac{h}{||P_{d_i} - P_{u_j}||}\right)$$

$$||P_{d_i} - P_{u_j}|| = \sqrt{(x_i^{MUD}-x_j^{uav})^2+(y_i^{MUD}-y_j^{uav})^2+(h_j^{uav})^2}$$

μ_1 and μ_2 are the environment related parameters. $\theta_{i,j}$ is the elevation Angle of MUD d_i to UAV u_j. $||P_{d_i} - P_{u_j}||$ is the distance between UAV u_j and MUD d_i, where $P_{u_j} = (x_j^{uav}, y_j^{uav}, h_j^{uav})$ and $P_{d_i} = (x_i^{MUD}, y_i^{MUD}, 0)$ represent the position coordinate of UAV u_j and MUD d_i, respectively. Therefore, when MUD d_i selects UAV edge computing, the channel gain is $g_i^{b_i,c_i} = 10^{-0.1L^{path}_{i,j}}$.

When the MUD d_i selects LEO satellite edge computing, the channel gain is $g_i^{b_i,c_i} = G_i^{b_i}G_1^{Fad}G_2^{Fad}(L_{i,b_i})^{C^{path}}$, where $G_i^{b_i}$ is the antenna gain of MUD d_i against satellite s_{b_i}, $G_1^{Fad} \sim \mathcal{CN}(0,1)$ is the complex Gaussian variable representing Rayleigh fading, L_{i,b_i} is the distance between MUD d_i and LEO satellite s_{b_i}, and C^{path} is the path exponent. Similar to [42], $G_2^{Fad} = (\frac{v}{4\pi L_{i,b_i} f_c})^2 A^{cr}_{i,b_i}$ is the fading including shadowing fading, rain, water vapor and other fading, where L_{i,b_i} is the distance between MUD d_i and LEO satellite s_{b_i}, f_c is the carrier frequency, $A^{cr}_{i,b_i} = 10^{([3\chi L_{i,b_i}]/10L_1)}$ is the attenuation due to clouds and rain with L_1 the height of a LEO and χ the attenuation through the cloud and rain in dB/km.

D. Computation Model

The computation task $H_i, i \in [1, N]$ can be processed locally or can be offloaded to UAVs or LEO satellites for remote processing through wireless channels.

1) Local Computing: MUD d_i executes computation task H_i locally. Different MUDs may have different computing capabilities (i.e. CPU cycles per second). The MUD d_i's computing capability is denoted by f_i^{local}. The task H_i's local computing delay is as follows:

$$t_i^{local} = \frac{C_i}{f_i^{local}}. \tag{12}$$

Similar to [35], [43], the computing energy consumption is

$$e_i^{local} = l_i^e C_i, \tag{13}$$

where l_i^e represents the energy consumed by MUD d_i per CPU cycle while processing the task locally. According to (12)

and (13), the cost of local computing can be obtained as

$$K_i^{local} = \lambda_i^t t_i^{local} + \lambda_i^e e_i^{local}, \tag{14}$$

where λ_i^t and λ_i^e are weighted parameters of delay and energy consumption respectively. There are $\lambda_i^t, \lambda_i^e \in [0,1]$ and $\lambda_i^t + \lambda_i^e = 1$. In practice, MUDs can appropriately adjust the weighted parameters. When MUD d_i pays more attention to energy consumption, it can set $\lambda_i^t < \lambda_i^e$. When MUD d_i pays more attention to latency, it can set $\lambda_i^t > \lambda_i^e$.

2) Offloading to the UAVs: MUDs can offload tasks to the ESs of UAVs for processing. UAVs can provide MUDs with computing resources, and can also transfer energy to MUDs by applying wireless power transfer technology. When MUD d_i offloads the task to UAV u_{b_i}, the task H_i's data transmission delay is

$$t_i^{tr} = \frac{B_i}{Dr_i}. \tag{15}$$

The task computing delay is

$$t_i^{UAV} = \frac{C_i}{f_{i,b_i}^{UAV}}, \tag{16}$$

where f_{i,b_i}^{UAV} denotes the computing capability (CPU cycles/second) obtained by MUD d_i on the edge server of UAV u_{b_i}. $f_{b_i}^{UAV}$ is the computing capability of the UAV u_{b_i}, there is $\sum_{d_l \in \mathbf{D} \cap b_l = b_i} f_{l,b_i}^{UAV} \le f_{b_i}^{UAV}$. According to (15) and (16), the total delay of MUD d_i offloading task H_i to UAV u_{b_i} for processing is

$$T_i^{UAV} = t_i^{tr} + t_i^{UAV}. \tag{17}$$

For UAV edge computing, the energy consumption of the MUD offloading task and the subsequent energy harvesting (EH) performed by the MUD as a receiver are considered. The energy consumption of MUD d_i is

$$e_i^{tr} = p_i t_i^{tr}. \tag{18}$$

MUDs can obtain energy from the UAV through wireless power transfer technology after completing the task offloading. The EH of MUD d_i is

$$e_i^{eh} = \eta p^{uav} g_i^{b_i, c_i}, \tag{19}$$

where η is the energy transmission efficiency, and p^{uav} is the power of the UAV WPT. Therefore, the cost of MUD d_i offloading task to the UAV u_{b_i} is

$$K_i^{UAV} = \lambda_i^t T_i^{UAV} + \lambda_i^e (e_i^{tr} - e_i^{eh}). \tag{20}$$

3) Offloading to the LEO Satellites: When MUD d_i offloads the task H_i to the LEO satellite s_{b_i}, the computing delay is

$$t_i^{LEO} = \frac{C_i}{f_{i,b_i}^{LEO}}, \tag{21}$$

where f_{i,b_i}^{LEO} denotes the computing capability obtained by MUD d_i on the edge server of LEO satellite s_{b_i}. $f_{b_i}^{LEO}$ is the LEO satellite s_{b_i}'s computing capability, there is $\sum_{d_l \in \mathbf{D} \cap b_l = b_i} f_{l,b_i}^{LEO} \le f_{b_i}^{LEO}$. In addition, the distance between MUD and LEO satellites is relatively large. Therefore, MUDs suffer from propagation delays when communicating with LEO satellites. The propagation delay is

$$t_i^p = \frac{L_{i,b_i}}{v}, \tag{22}$$

where L_{i,b_i} is the distance between MUD d_i and LEO satellite s_{b_i}. Therefore, the delay for MUD d_i to offload task H_i to LEO satellite s_{b_i} is

$$T_i^{LEO} = t_i^p + t_i^{tr} + t_i^{LEO}. \tag{23}$$

The cost of MUD d_i offloading task to the LEO satellite s_{b_i} is

$$K_i^{LEO} = \lambda_i^t T_i^{LEO} + \lambda_i^e e_i^{tr}. \tag{24}$$

E. Problem Formulation

For each MUD $d_i \in \mathbf{D}$, the cost function is as follows.

$$K_{o_{-i}}(o_i) = \begin{cases} K_i^{local}, & o_i = (0,0,0) \\ K_i^{UAV}, & o_i = (1, b_i, c_i) \\ K_i^{LEO}, & o_i = (2, b_i, c_i) \end{cases}, \tag{25}$$

where o_{-i} represents the set of offloading decisions of all MUDs except MUD d_i. The MUDs are subject to some constraints in making decisions. The detailed problem formulation is as follows:

$$\begin{aligned} \min \ & \sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i) \\ \text{s.t. } & C_1: T_i^{LEO} \le T_i^L, \text{if } a_i = 2 \\ & C_2: K_{o_{-i}}(o_i) \le K_i^{local}, \forall d_i \in \mathbf{D} \\ & C_3: \sum_{d_l \in \mathbf{D} \cap b_l = j_1} f_{l,j_1}^{UAV} \le f_{j_1}^{UAV}, \forall u_{j_1} \in \mathbf{U} \\ & C_4: \sum_{d_l \in \mathbf{D} \cap b_l = j_2} f_{l,j_2}^{LEO} \le f_{j_2}^{LEO}, \forall s_{j_2} \in \mathbf{S}. \end{aligned} \tag{26}$$

C_1 is the coverage time constraint when the MUDs offload tasks to the LEO satellites. C_2 means that when the MUDs offload the tasks, the cost obtained should be lower than the cost of local computing; otherwise, the MUDs will not offload the tasks. C_3 and C_4 are the computing resource constraints of UAVs and LEO satellites, respectively.

Theorem 1: The problem (26) is NP-hard.

Proof: The problem can be proven to be NP-hard with the multiple Knapsack (MK) problem. In the MK problem, there are n items $\mathbb{I} = \{im_1, \ldots, im_n\}$ and m backpacks $\mathbb{B} = \{bk_1, \ldots, bk_m\}$. The capacity of each knapsack $bk_j \in bk$ is c_j. The income and weight of each item $im_i \in im$ are ie_i and w_i, respectively. $q_i = \{q_{i,1}, \ldots, q_{i,m}\}$ indicates the decision to the item im_i. $q_{i,j} = 1$ indicates that item im_i is packed into the knapsack bk_j, and $q_i = 0$ indicates that it is not packed into any backpack. $I\{\mathcal{P}\}$ is a condition function. If $\mathcal{P}$ is true, $I\{\mathcal{P}\} = 1$. Otherwise, $I\{\mathcal{P}\} = 0$. The goal of the MK problem is to maximize the overall income.

$$\max \sum_{im_i \in \mathbf{Im}} ie_i I\{q_i \ne 0\}$$

$$\text{s.t.} \sum_{im_i \in \mathbf{Im}} w_i I\{q_{i,j}=1\} \le c_j, \forall bk_j \in \mathbb{B}.$$

In the problem (26), there is a certain resource requirement for each MUD and a certain user capacity for each wireless channel. Thus, they can be seen as items and backpacks in the MK problem, respectively. The goal of problem (26) can be seen as a transformation of the goal of problem MK, and constraints C_3 and C_4 are equivalent to $\sum_{im_i \in \mathbf{Im}} w_i I\{q_{i,j}=1\} \le c_j, \forall bk_j \in \mathbb{B}$. Other constraints of problem (26) can be projected as weights into the MK problem. Therefore, the problem (26) can be transformed from the MK problem and is NP-hard. □

According to Theorem 1, it is difficult to obtain the optimal solution of problem (26) in polynomial time. The time complexity required to obtain the optimal solution through the centralized method is significant. Moreover, in problem (26), each MUD focuses on its own benefit and competes for the limited system resources. In other words, each MUD does not aim at reducing the overall cost at the expense of increasing its own cost. How to balance the benefits of the individual MUD and the benefits of overall MUDs is a challenge. Therefore, we propose a distributed approach based on game theory to solve this problem. With the distributed approach, different MUDs are allowed to make their own decisions and finally reach to an equilibrium state, i.e., the Nash equilibrium state. In other words, the balanced benefits of the individual MUD and the overall MUDs are achieved.

IV. LEO-UAV Task Offloading Game

In this section, the LEO-UAV task offloading Game (LUTO-Game) model is established, and the property of the game is analyzed theoretically.

A. LUTO-Game Formulation

To solve this task offloading problem, the idea of game theory is used, each MUD has a certain degree of autonomy. We reformulate the problem as the LUTO-Game $G = (\mathbf{D}, \{O_i\}_{d_i \in \mathbf{D}}, \{K_{o_{-i}}(o_i)\}_{d_i \in \mathbf{D}})$. $\mathbf{D}$ is a players set, and MUD $d_i \in \mathbf{D}$ makes the offloading decision $o_i \in \{(0,0,0) \bigcup (a_i, b_i, c_i)\}$. O_i is the set of available offloading decisions, and $K_{o_{-i}}(o_i)$ is the cost of MUD d_i. All players compete for limited resources and minimize their own costs. Definition 1 shows the detailed definition for the Nash Equilibrium (NE) solution of the LUTO-Game G.

Definition 1: If no MUD can change its decision to decrease cost, $o^* = (o_1^*, o_2^*, \ldots, o_N^*)$ can reach a Nash Equilibrium (NE) for the LUTO-Game $G = (\mathbf{D}, \{O_i\}_{d_i \in \mathbf{D}}, \{K_{o_{-i}}(o_i)\}_{d_i \in \mathbf{D}})$, i.e.,

$$K_{o_{-i}^*}(o_i^*) \le K_{o_{-i}^*}(o_i), \forall d_i \in \mathbf{D}, \forall o_i \in O_i. \tag{27}$$

Then, for a set of NE decisions, each participant's decision is the best response decision to the other participants, described in detail in Property 1.

Property 1: For the offloading strategy $o^* = (o_1^*, o_2^*, \ldots, o_N^*)$ of the LUTO-Game G, MUD d_i's offloading decision $o_i^* \in O_i$ is the best response to the other MUDs' decisions o_{-i}^*.

Proof: For MUD d_i, if $o_i^* \in O_i$ is not the best response decision, there must exist a decision $o_i \in O_i$ that can decrease its cost, i.e., $K_{o_{-i}^*}(o_i^*) > K_{o_{-i}^*}(o_i)$. This hypothesis conflicts with (27). Therefore, o_i^* is the MUD d_i's best response decision. □

According to Property 1, the LUTO-Game allows each MUD to make the offloading decision. Therefore, the offloading strategy can be obtained in a distributed manner. This method can reduce complexity and improve efficiency.

B. Analysis of LUTO-Game Solution

In order to solve the problem (26), a task offloading rule is proposed to optimize the total utility of MUDs in LUTO-Game. The offloading rule is designed based on the principle of overall cost reduction, i.e., the decision update of any MUD can reduce its own cost and the system cost. Suppose the current decision of MUD $d_i \in \mathbf{D}$ is o_i and it wants to change to decision o_i' to reduce cost. MUD d_i changing decisions will affect its own cost and the cost of other MUDs. The offloading rules are defined in Definition 2.

Definition 2: (Task Offloading Rule) After MUD d_i changes the decision, the amount of cost reduction is $\triangle K_i = K_{o_{-i}}(o_i) - K_{o_{-i}}(o_i')$, and the impact on other MUDs is $\triangle K_{-i} = \sum_{d_l \ne d_i} K_{o'_{-l}}(o_l) - \sum_{d_l \ne d_i} K_{o_{-l}}(o_l)$. d_i can change the decision if it reduces the overall cost after changing the decision, i.e.,

$$\triangle K_i > \triangle K_{-i}. \tag{28}$$

It guarantees that MUD reduces its own cost while reducing the overall cost. Then, we analyze the existence of NE in the LUTO-Game by proving that the game is a potential game. The definition of a potential game is as follows.

Definition 3: For a game problem, if there is a function $Y(o_i, o_{-i})$ that satisfies (29), then the game is a potential game.

$$K_{o_{-i}}(o_i') \le K_{o_{-i}}(o_i) \Rightarrow Y(o_i', o_{-i}) \le Y(o_i, o_{-i}), \tag{29}$$

where $d_i \in \mathbf{D}$ and $o_i, o_i' \in O_i$.

Theorem 2: The LUTO-Game is a potential game, and (30) gives the potential function.

$$Y(o_i, o_{-i}) = \sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i) \tag{30}$$

Proof: We assume that MUD $d_i \in \mathbf{D}$ has two decisions o_i and o_i', and $K_{o_{-i}}(o_i) > K_{o_{-i}}(o_i')$. According to Definition 2, there is

$$\begin{aligned} &Y(o_i, o_{-i}) - Y(o_i', o_{-i}) \\ &= \sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i) - \sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i') \\ &= K_{o_{-i}}(o_i) + \sum_{d_l \ne d_i} K_{o_{-l}}(o_l) \\ &\quad - K_{o_{-i}}(o_i') - \sum_{d_l \ne d_i} K_{o'_{-l}}(o_l) \\ &= K_{o_{-i}}(o_i) - K_{o_{-i}}(o_i') \end{aligned}$$

$$-\left(\sum_{d_l \neq d_i} K_{o'_{-l}}(o_l) - \sum_{d_l \neq d_i} K_{o_{-l}}(o_l)\right)$$
$$= \triangle K_i - \triangle K_{-i} > 0 \tag{31}$$

Therefore,

$$K_{o_{-i}}(o_i) > K_{o_{-i}}(o'_i) \Rightarrow Y(o_i, o_{-i}) > Y(o'_i, o_{-i})$$

and Theorem 2 holds. □

V. Joint UAV and LEO Satellite Task Offloading Algorithm

This section proposes the Joint UAV and LEO Satellite Task Offloading (JULTO) algorithm to solve the offloading problem for the ULSE network. The theoretical analysis for the JULTO algorithm is also given.

A. Algorithm Design

Based on Theorem 2, the game has a limited improvement property (FIP [44]). As a result, the NE offloading strategy can be obtained by a limited iteration number. To find the LUTO-Game's NE solutions, the JULTO algorithm is designed. The JULTO algorithm operates in an iterative manner, where each MUD makes offloading decisions independently. In each iteration, each MUD searches for the best decision and then competes with other MUDs for the chance to update the decision. Then, the winner of the participant competition gets an update opportunity and can update its decision. The algorithm continues to iterate until no MUD wants to change its decision further, at which point the algorithm terminates. Overall, the JULTO algorithm provides an efficient method for obtaining the NE solution based on FIP. By allowing MUDs to make offloading decisions independently and compete for update opportunities, the game can find solutions that satisfy the NE conditions. The details are illustrated in Algorithm 1.

Algorithm 1: Joint UAV and LEO Satellite Task Offloading (JULTO) Algorithm.

Input: $\mathbf{D} = \{d_1, \ldots, d_N\}$, $\mathbf{U} = \{u_1, \ldots, u_{M_1}\}$, $\mathbf{S} = \{s_1, \ldots, s_{M_2}\}$ and other parameters
Output: the task offloading strategy

1 **Initialization:**
2 $o = \{o_1, o_2, \ldots, o_N\}$, the decision of MUD $d_i \in \mathbf{D}$ is $o_i = (a_i, b_i, c_i) = (0, 0, 0)$
3 **End Initialization**
4 **repeat**
5 **for** *each MUD* $d_i \in \boldsymbol{D}$ **do**
6 Calculate the delay T_i and cost $K_{o_{-i}}(o_i)$
7 **if** $a_i \neq 0$ *and* $K_{o_{-i}}(o_i) > K_i^{local}$ **then**
8 $o_i = (0, 0, 0)$
9 **if** $a_i = 2$ *and* $T_i > T_i^L$ **then**
10 $o_i = (0, 0, 0)$
11 Calculate $\sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i)$
12 **for** *each MUD* $d_i \in \boldsymbol{D}$ **do**
13 **for** *each UAV* $u_{j_1} \in \boldsymbol{U}$ *and LEO satellite* $s_{j_2} \in \boldsymbol{S}$ *that MUD* d_i *can select* **do**
14 **for** *each channel* $c_{j_1}^u$ *of UAV and* $c_{j_2}^L$ *of LEO satellite* **do**
15 Calculate the cost when d_i's task data is offloaded by channel $c_{j_1}^u$ or $c_{j_2}^L$
16 Find a new decision o'_i that can reach the minimum $K_{o_{-i}}(o'_i)$
17 **if** $\sum_{d_i \in \boldsymbol{D}} K_{o_{-i}}(o'_i) < \sum_{d_i \in \boldsymbol{D}} K_{o_{-i}}(o_i)$ *and* $o'_i \neq o_i$ **then**
18 Send o'_i to the set of competing decisions
19 **if** d_i *is the winner* **then**
20 Update d_i's decision to o'_i
21 **until** *no MUD wants to update the decision for cost reduction.*;
22 **return** $o = \{o_1, \ldots, o_N\}$

First, the algorithm needs to set the necessary parameters. In the initial state, no MUD makes an offloading decision, and the decisions of all MUDs are initialized to (0,0,0). Next, each MUD updates its decision by iteration. The FIP guarantees the convergence of the JULTO algorithm, and finally reaches the Nash equilibrium state, that is, no MUD can change its offloading decision.

In each iteration, each MUD calculates the current delay and cost (Line 6). If the MUD $d_i \in \mathbf{D}$ does not choose local computing, it is necessary to judge whether the utility at this time is better than that of local computing (Lines 7-8). In addition, if the MUD $d_i \in \mathbf{D}$ chooses to offload tasks to the satellite, it needs to check whether the satellite coverage time constraint is met at this time (Lines 9-10). Next, the total cost $\sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i)$ of all MUDs is calculated (Line 11). After that, each MUD $d_i \in \mathbf{D}$ finds a new decision o'_i that can reach the minimum $K_{o_{-i}}(o'_i)$ in parallel (Lines 12-16). The new decision o'_i of MUD d_i needs to be beneficial for both MUD d_i's cost and the overall system cost. Therefore, the new decision needs to be guaranteed to reduce the overall cost, i.e., $\sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o'_i) < \sum_{d_i \in \mathbf{D}} K_{o_{-i}}(o_i)$. The MUD that wants to update the decision sends the new decision o'_i to a set of competing decisions. Then, these MUDs compete for update opportunities (lines 17-18). In this paper, the competition determines the winner in a non-deterministic manner such as a random method. If MUD d_i becomes the winner, it will get an update opportunity and its decision o_i will be updated to o'_i (Lines 19-20).

Finally, when no MUDs can change their decisions, the JULTO algorithm ends (Line 21). At this point, the system reaches an equilibrium state and the task offloading decisions of all MUDs constitute the NE solution of the problem. During the process, MUDs make their own task-offloading decisions in parallel. Therefore, the JULTO algorithm is a distributed algorithm.

B. Convergence Analysis

After a finite number of iterations, the LUTO-Game will finally reach a Nash Equilibrium offloading strategy because of

the FIP property. Next, we prove the upper bound of the number of iterations, as in Theorem 3.

Theorem 3: There is an upper limit on the number of iterations, which satisfies

$$R^{FIP} \leq N(\max\{t_{max}^{local}, e_{max}^{local}\} - \min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\}),$$

where $t_{max}^{local} = \frac{C_{max}}{f_{min}^{local}}$, $e_{max}^{local} = l_{max}^{e} C_{max}$, $t_{min}^{tr} = \frac{B_{min}}{W_{max}\log_2(1+\frac{p_{max}g_{max}}{\sigma^2})}$, $t_{min}^{UAV} = \frac{C_{min}}{f_{max}^{UAV}}$, $e_{min}^{tr} = \frac{p_{min}B_{min}}{W_{max}\log_2(1+\frac{p_{max}g_{max}}{\sigma^2})}$, $e_{max}^{eh} = \eta p_{max} g_{max}$, $C_{min} = \min\{C_1, \ldots, C_N\}$, $l_{max}^{e} = \max\{l_1^e, \ldots, l_N^e\}$, $f_{max}^{local} = \max\{f_1^{local}, \ldots, f_N^{local}\}$, $B_{min} = \min\{B_1, \ldots, B_N\}$, $p_{min} = \min\{p_1, \ldots, p_N\}$, $p_{max} = \max\{p_1, \ldots, p_N\}$, $W_{max} = \max\{W_{j,k}, j \in \{1, \ldots, M_1\}, k \in \{1, \ldots, c_j^u\}\}$, $g_{max} = \max\{g_i^{j,k}, d_i \in \mathbf{D}, j \in \{1, \ldots, M_1\}, k \in \{1, \ldots, c_j^u\}\}$, $f_{max}^{UAV} = \max\{f_{i,j}^{UAV}, d_i \in \mathbf{D}, j \in \{1, \ldots, M_1\}\}$.

Proof: There are maximum cost and minimum cost for any MUD $d_i \in \mathbf{D}$. Based on the model of the problem, MUD d_i achieves the maximum cost when it chooses local processing. There is $K(o_i, o_{-i}) \leq \max\{K_i^{local}\} = \max\{\lambda_i^t t_i^{local} + \lambda_i^e e_i^{local}\}$. According to $\lambda_i^t + \lambda_i^e = 1$, we can obtain

$$K(o_i, o_{-i}) \leq \max\{t_{max}^{local}, e_{max}^{local}\} \leq \max\left\{\frac{C_{max}}{f_{min}^{local}}, l_{max}^{e} C_{max}\right\}.$$

The MUD d_i may obtain the minimum cost when it offloads the task to the UAV for processing. There exists $K(o_i, o_{-i}) \geq \min\{K_i^{UAV}\} = \min\{\lambda_i^t T_i^{UAV} + \lambda_i^e e_i^{tr}\}$. We can obtain (32) shown at the bottom of the this page.

If MUD d_i updates its decision from o_i to o_i', d_i's cost decreases, $K_{o_{-i}}(o_i') \leq K_{o_{-i}}(o_i)$. According to Definition 3, the potential function $\phi_{-a}(a_i)$ meets

$$Y(o_i, o_{-i}) - Y(o_i', o_{-i}) \geq 0.$$

Therefore, the reduction of the potential function before and after each iteration is at least 1. Therefore, Theorem 3 is proved. □

C. Price of Anarchy in Total Cost

Generally, there are many different Nash equilibrium states in the LUTO-Game. The Nash equilibrium solution obtained by the JULTO algorithm may not be the global optimal solution. For the problem (26), to evaluate the gap between the NE solution and the global optimal solution, we consider an important attribute: price of anarchy (PoA).

The PoA is a metric used in game theory to evaluate the efficiency of Nash equilibrium solutions. It measures the ratio between the worst outcome achievable by adopting a Nash equilibrium strategy and the outcome achievable by the optimal solution to the problem. PoA is the ratio between the utility of the least effective Nash equilibrium solution and the utility of the globally optimal solution, used to evaluate and measure the efficiency of LUTO-Game's NE solution. In LUTO-Game, $\mathbb{A}$ represents the set of all Nash equilibrium solutions, $\hat{o} = \{\hat{o}_1, \ldots, \hat{o}_N\}$ represents a centralized optimal solution. The PoA in the total cost is

$$POA_{cost} = \frac{\max_{o^* \in \mathbb{A}}\{\sum_{d_i \in \mathbf{D}} K_{o_{-i}^*}(o_i^*)\}}{\sum_{d_i \in \mathbf{D}} K_{\hat{o}_{-i}}(\hat{o}_i)}. \quad (33)$$

In the LUTO-Game, the PoA in total cost obtained by the JULTO algorithm is analyzed, and its upper and lower bounds can be given. Therefore, Theorem 4 can be obtained.

Theorem 4: The PoA of the LUTO-Game in total cost satisfies

$$1 \leq POA_{cost} \leq \frac{\max\{t_{max}^{local}, e_{max}^{local}\}}{\min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\}}.$$

Proof: For any offloading strategy $o^* \in \mathbb{A}$ and optimal offloading strategy $\hat{o}$, there exists $\sum_{d_i \in \mathbf{D}} K(o_i^*, o_{-i}^*) \geq \sum_{d_i \in \mathbf{D}} K(\hat{o}_i, \hat{o}_{-i})$. Therefore, $\frac{\sum_{d_i \in \mathbf{D}} K(o_i^*, o_{-i}^*)}{\sum_{d_i \in \mathbf{D}} K(\hat{o}_i, \hat{o}_{-i})} \geq 1$.

There are maximum cost and minimum cost for any MUD $d_i \in \mathbf{D}$. We can obtain

$$K(o_i, o_{-i}) \leq \max\left\{\frac{C_{max}}{f_{min}^{local}}, l_{max}^{e} C_{max}\right\}$$

$$K(o_i, o_{-i}) \geq \min\left\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\right\}.$$

For offloading strategy $o^* \in \mathbb{A}$, there are

$$\sum_{d_i \in \mathbf{D}} K(o_i^*, o_{-i}^*) \leq N \max\left\{\frac{C_{max}}{f_{min}^{local}}, l_{max}^{e} C_{max}\right\}.$$

The centralized optimal solution $\hat{o}$ satisfies

$$\sum_{d_i \in \mathbf{D}} K(\hat{o}_i, \hat{o}_{-i}) \geq N \min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\}.$$

Therefore, Theorem 4 can be proved. □

D. Complexity Analysis

In each iteration, first, each MUD calculates its delay and cost (lines 5-11), with only a few basic mathematical operations, the time complexity of which can be viewed as $\mathcal{O}(1)$. Each MUD then searches for its own optimal decision in parallel (lines 13-16) with a time complexity of $\mathcal{O}(M_1 c^u + M_2 c^L)$. Therefore, the time complexity of each iteration is $\mathcal{O}(1) + \mathcal{O}(M_1 c^u + M_2 c^L) = \mathcal{O}(M_1 c^u + M_2 c^L)$. Further, Theorem 3 proves that the upper bound of the number of iterations, that is, $R^{FIP} \leq N(\max\{t_{max}^{local}, e_{max}^{local}\} - \min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\})$. Therefore, the time complexity of

$$K(o_i, o_{-i}) \geq \min\{T_{min}^{UAV}, e_{min}^{tr}\} = \min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\}$$

$$\Rightarrow K(o_i, o_{-i}) \geq \min\left\{\frac{B_{min}}{W_{max}\log_2(1+\frac{p_{max}g_{max}}{\sigma^2})} + \frac{C_{min}}{f_{max}^{UAV}}, \frac{p_{min}B_{min}}{W_{max}\log_2(1+\frac{p_{max}g_{max}}{\sigma^2})} - \eta p_{max} g_{max}\right\}. \quad (32)$$

TABLE II
EXPERIMENT SETTINGS (1)

Parameter	Value
The height of the LEO satellites	784 km
The bandwidth of Channel	5 MHz
Data size B_i of task	3 MB∼5 MB
The radius of the earth	6371 km
The computing capability allocated to MUD by UAV	10 GHz
The computing capability of MUD	2 GHz
The computing capability allocated to MUD by LEO satellite	10 GHz

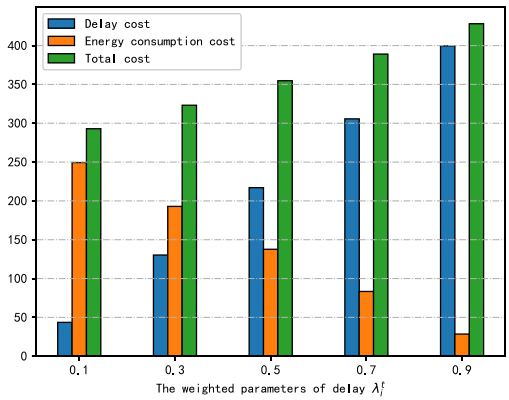

Fig. 4. The total cost of MUDs with different weighted parameters.

the algorithm is $\mathcal{O}(N(M_1c^u + M_2c^L)(\max\{t_{max}^{local}, e_{max}^{local}\} - \min\{t_{min}^{tr} + t_{min}^{UAV}, e_{min}^{tr} - e_{max}^{eh}\}))$.

VI. PERFORMANCE EVALUATION

A. Parameter Configuration

We consider that multiple MUDs are randomly distributed in a certain area, and UAVs and LEO satellites carrying edge servers can provide MUDs with computing services. Each MUD generates a computing task that needs to be processed (task offloading or local computing). The task H_i's data size B_i is randomly set between 3 MB and 5 MB. The CPU cycle C_i required by task H_i can be obtained according to $C_i = B_i\nu$, where $\nu = 1000$ cycle/bit. The Table II shows the main parameter configuration. For MUD $d_i \in \mathbf{D}$, the transmission power $p_i = 1000$ mWatts, and the computing capability is 2 GHz. The wireless channel bandwidth is 5 MHz [4]. For UAV $u_{j_1} \in \mathbf{U}$, the computing capability f_{i,j_1}^{UAV} allocated to the MUD d_i is 10 GHz. For LEO satellite $s_{j_2} \in \mathbf{S}$, the satellite's height is 784 km. The computing capability f_{i,j_2}^{LEO} allocated by the satellite s_{j_2} to the MUD d_i is 10 GHz. The radius of the earth is 6371 km [39].

B. Analysis of Parameter

Fig. 4 shows the MUDs' total cost obtained by the JULTO algorithms with different weighted parameters of delay λ_i^t and energy consumption $(1 - \lambda_i^t)$. We set $N = 50$, $M_1 = 5$, $M_2 = 3$, $c^u = 3$, and $c^L = 3$. It can be found that the total cost of MUDs increases with the increase in delay weight. Therefore, in the cost function, the delay cost plays a more important part in the cost.

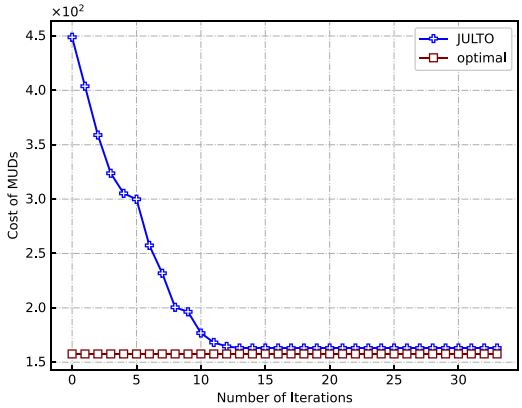

Fig. 5. The JULTO algorithm's convergence analysis.

TABLE III
EXECUTION TIME WITH DIFFERENT NUMBERS OF MUDS

UAV number (M_1) = 2, LEO satellites number (M_2) = 1, channel number of UAV (c^u) = 1, channel number of satellite (c^L) = 1		
Number of MUDs (N)	Centralized method's execution time (ms)	JULTO method's execution time (ms)
5	60.84	0.33
6	386.97	0.56
7	2402.57	0.75
8	14560.04	1.09
9	87044.14	1.29
10	512096.68	1.78

Fig. 5 shows the JULTO algorithm's convergence performance. The optimal algorithm is a centralized method and can obtain a globally optimal solution. The MUD number (N) is 10, the UAV number (M_1) is 3, the LEO satellite number (M_2) is 1, and the wireless channel numbers of UAVs and LEO satellites (c^u and c^L) are 1. In Fig. 5, before the 12th iteration, the total cost of MUD obtained by the JULTO algorithm gradually decreases. After 12 iterations, the total cost of MUD reaches a stable state. This verifies the JULTO algorithm is capable of reaching convergence. In addition, the total cost obtained by the JULTO algorithm is close to the optimal algorithm. This shows that the JULTO algorithm can achieve good performance after iteration.

Table III shows the execution time of the centralized method and our proposed JULTO method with different MUD numbers. The numbers of UAVs, LEO satellites, and wireless channels are 3, 1, and 1 respectively. As the number of MUDs increases, the execution time of centralized method and JULTO method increases. The execution time of the JULTO method is much shorter compared to the centralized method.

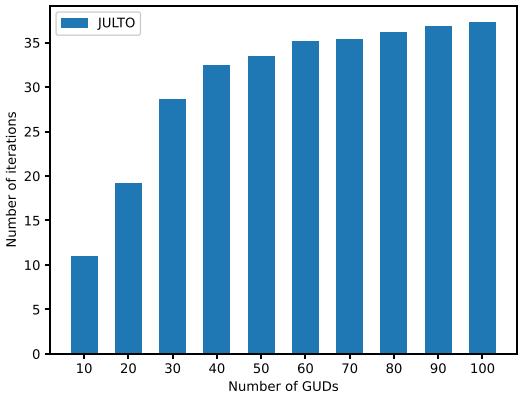

Fig. 6. Iteration number versus MUD number.

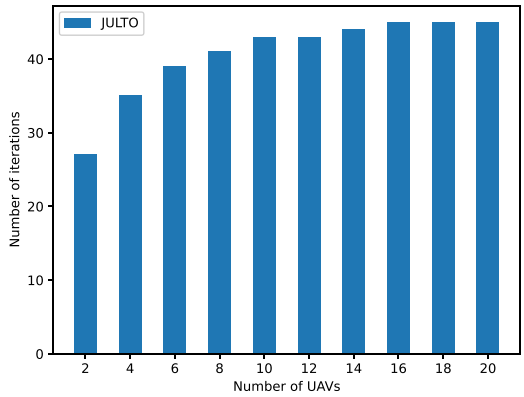

Fig. 7. Iteration number versus UAV number.

Generally, the different hardware of the experiment machine may lead to different convergence times of the algorithm. Therefore, convergence time can be measured based on the number of iterations. Specifically, in order to evaluate the iteration number required for the JULTO method, we vary the number of MUDs, wireless channels, UAVs, and LEO satellites.

Fig. 6 shows the iteration numbers for the JULTO algorithm versus varying MUDs' numbers. N is increasing from 10 to 100. M_1 is 5, M_2 is 3, c^u is 3, and c^L is 3. In Fig. 6, the number of iterations increases as the number of MUDs increases. In addition, the growth rate of the number of iterations gradually slows down. This shows that as N increases, the JULTO algorithm's convergence time grows slower than linearly. This is because of the participant competition mechanism in JULTO's algorithm. In particular, when N is small, there are enough resources to service MUDs. As the number of MUDs increases, the competition of MUDs for resources intensifies, so the JULTO algorithm needs more iterative processes to reach the equilibrium state. As N increases to a certain number, system resources are insufficient. At this point, the number of MUDs allocated to the edge server has been saturated, and if the number of MUDs continues to increase further, MUDs can only process tasks locally, without competing with other MUDs. Therefore, the algorithm does not need more iterations to reach the equilibrium state.

Figs. 7 and 8 respectively show the iteration number required by the JULTO algorithm versus M_1 and M_2. In Fig. 7, the iteration number keeps increasing as M_1 increases from 2 to 10. When M_1 increases from 12 to 20, the iteration number changes little. Similarly, in Fig. 8, as M_2 increases from 1 to 5, the iteration number keeps increasing. When M_2 increases from 6 to 10, the iteration number does not increase significantly. This is because when M_1 or M_2 is small, MUD has more optional decisions as M_1 or M_1 increases. The JULTO algorithm needs more iterations to get the equilibrium solution. However, for a fixed number of MUDs, system resources become sufficient when UAVs or LEO satellites reach a certain number. Therefore, the iteration number does not increase significantly.

Fig. 9 shows the iteration number with different numbers of channels. N is 50, M_1 is 5, M_2 is 3, and c^u and c^L are increasing from 1 to 10. In Fig. 9, the iteration number increases as c^u and c^L increase. More wireless channels to choose from bring more strategies to MUDs. However, while the solution space's size grows exponentially with the wireless channel number, the iteration number of the JULTO algorithm grows shorter than linearly.

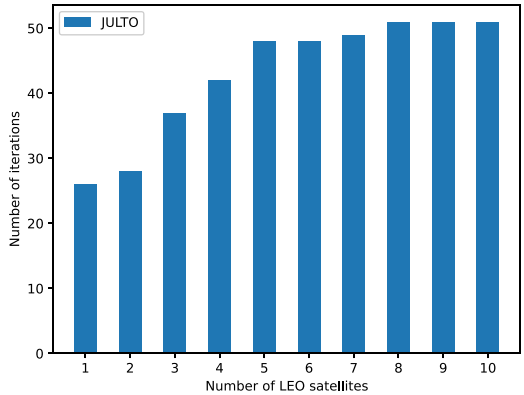

Fig. 8. Iteration number versus LEO satellite number.

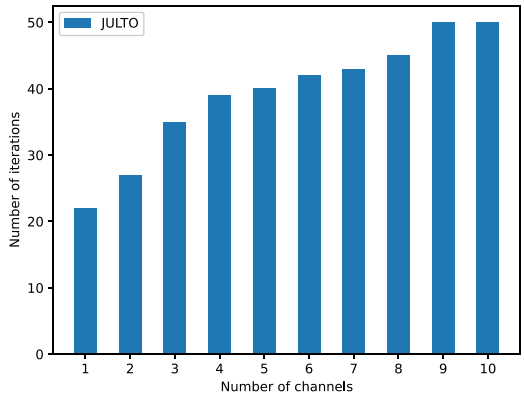

Fig. 9. Iteration number versus channel number.

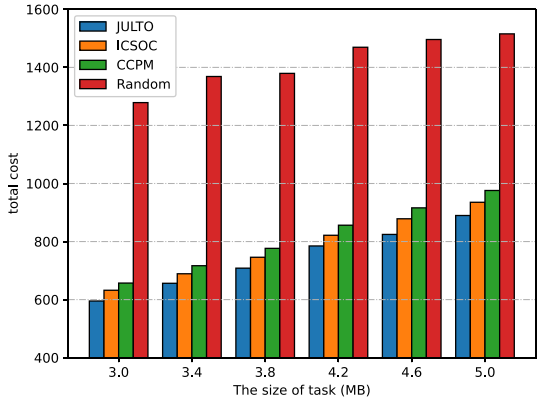

Fig. 10. Total cost versus size of task.

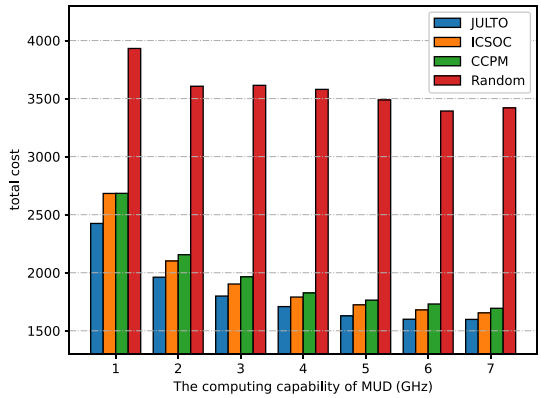

Fig. 11. Total cost versus computing capability of MUD.

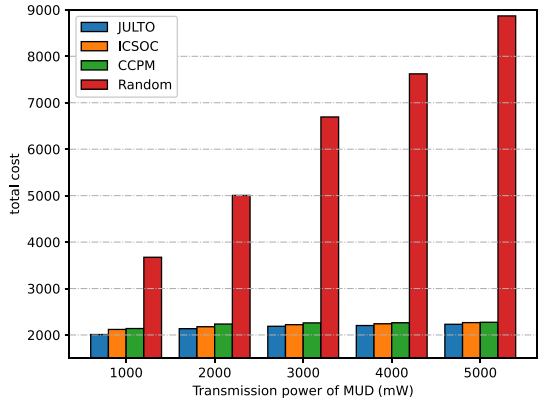

Fig. 12. Total cost versus transmission power.

C. Comparison Experiments

We compare the JULTO algorithm with 3 other comparison methods to evaluate its performance. The comparison algorithms are shown below.

- *ICSOC:* This method is an extension of the method of [45] to solve the offloading problem. Particularly, each MUD selfishly seeks to obtain the more resources to minimize its cost while satisfying the constraints.
- *CCPM:* This method is extended from [46]. In this method, MUDs are ranked according to channel transmission conditions. Then, according to the sequence of the sorted MUDs, the decisions are updated to find the optimal decision.
- *Random:* In this method, each MUD makes a decision randomly. When the LEO satellite coverage time constraints are met, the MUD randomly selects a method (local computing, offloading tasks to UAVs or LEO satellites for processing). Otherwise, the MUD randomly chooses one of two ways to process the task (local computing or offloading the task to the drone).

Fig. 10 shows the MUDs' total cost obtained by the four algorithms with different task sizes. N is 50, M_1 is 5, M_2 is 3, c^u is 3, and c^L is 3. The larger the task size, the larger the total cost of MUDs. This is because the task's size becomes larger, and the MUDs' cost required to process the task data increases. In addition, the proposed JULTO algorithm has a lower cost compared with the other comparison algorithms.

The Random algorithm randomly selects the offloading decision for each MUD, and ultimately cannot guarantee the lowest total cost. For the ICSOC method, MUDs greedily choose the decision that can give themselves the lowest cost, which may lead to conflicts between multiple MUDs and increase the total cost. In the CCPM method, MUDs are sorted according to the channel transmission conditions, and then appropriate decisions are selected according to the order. When all MUDs go through a round of decision selection, the CCPM method cannot further optimize the total cost. At this point, there may be some MUDs that can achieve lower costs by updating their decisions. For the JULTO method, through multiple rounds of iteration and competition among MUDs, the total cost will be relatively low until no MUD changes the decision. Therefore, for the utility comparison of these four methods, the proposed JULTO method can outperform other methods and is close to the optimal total cost.

Fig. 11 shows the total MUDs' cost obtained by the four algorithms for different computing capabilities of MUDs. The total cost of MUDs decreases as the computing capability of MUDs increases. This is because the latency of task processing locally decreases as the processing power increases, causing the total cost of MUDs to increase. Moreover, in Fig. 11, it can be found that the utility achieved by the JULTO algorithm is better than the other three algorithms.

Fig. 12 shows the total cost of the MUDs obtained by the four algorithms when the transmission power is different. We can find that the higher the transmission power of the MUD, the higher the total cost. This is because increasing the transmit power results in increased energy consumption for MUDs offloading tasks to edge servers. Therefore, the total cost of all MUDs increases. Furthermore, in Fig. 12, the performance of the JULTO algorithm is better than the other three comparison algorithms, which reflects the performance superiority of JULTO.

In addition, to analyze the JULTO algorithm's performance with different problem sizes, we establish three different experiment settings, which are listed in Table IV.

In Fig. 13, the four algorithms' average cost with different MUD numbers are shown. N is 10 to 100, M_1 is 5, M_2 is 3, c^u

TABLE IV
EXPERIMENT SETTINGS (2)

the MUD number (N)	the UAV number (M_1)	the satellite number (M_2)	the channel numbers (c^u and c^L)
$10 \sim 100$	5	3	3
50	$2 \sim 20$	3	3
50	5	$1 \sim 10$	3
50	5	3	$1 \sim 10$

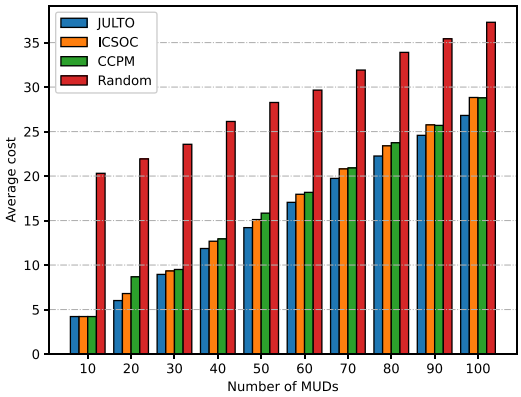

Fig. 13. Average cost versus number of MUDs.

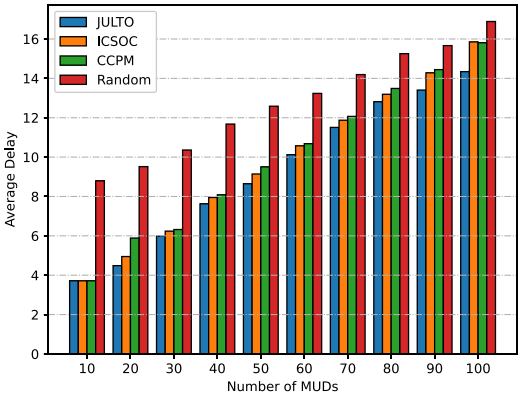

Fig. 14. The average delay of MUDs with different numbers of MUDs.

is 3, and c^L is 3. All four algorithms' average cost increases with N. In the experiment's set, the edge server number and wireless channel number are fixed. More MUDs are added, leading to the exhaustion of communication and computing resources on the system. Therefore, the number of MUDs processing tasks locally increases, and the average cost gradually increases. In addition, In Fig. 13, compared with other comparison algorithms, the MUDs' average cost implemented by the JULTO algorithm is lower.

Figs. 14 and 15 respectively give the average delay and energy consumption of the four algorithms under different MUD numbers. We vary the values of N from 10 to 100, and set $M_1 = 5$, $M_2 = 3$, $c^u = 3$, $c^L = 3$. The average delay and energy consumption of all four algorithms increase when the value of N

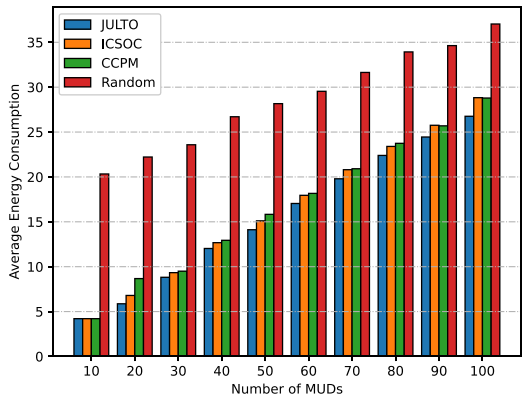

Fig. 15. The average energy consumption of MUDs with different numbers of MUDs.

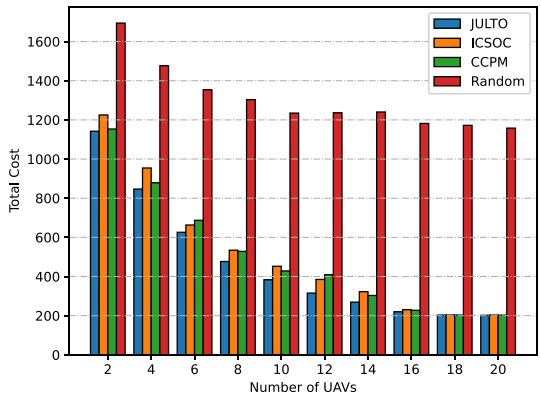

Fig. 16. Total cost versus number of UAVs.

increases. This is because with more MUDs the communication and computing resources on the system tend to be insufficient. When the communication resources are insufficient, the interference of the MUD's task data transmission increases, and the transmission delay and energy consumption of the MUD for task offloading increases. At the same time, since the resources are insufficient to support more MUDs to offload tasks, the number of MUDs for local task processing increases, and the average delay and energy consumption gradually increase. Nevertheless, compared with other comparison algorithms, the average delay and energy consumption of MUDs implemented by the JULTO algorithm are the lowest.

Fig. 16 shows all MUDs' total cost for the four algorithms when M_1 is 2 to 20. N is 50, M_2 is 3, c^u is 3, and c^L is 3. The total cost obtained by the four algorithms decreases with the increase in M_1. The reason is that as M_1 increases, so do transmission resources and computing resources increase. More MUDs' tasks can be offloaded to UAVs, and the total cost of all MUDs is reduced. However, the cost does not improve much when M_1 exceeds 16. This is because increasing the number of UAVs does not improve MUDs' utility when resources are sufficient. In addition, in Fig. 16, it is easy to see that the JULTO algorithm achieves a lower MUDs' total cost than other comparison algorithms.

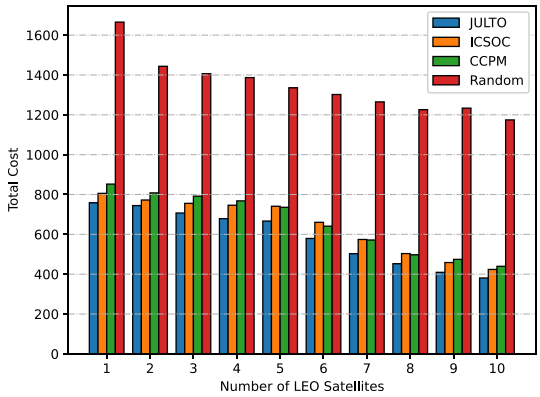

Fig. 17. Total cost versus number of LEO satellites.

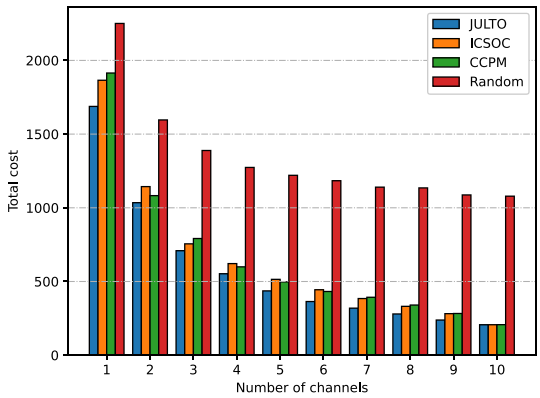

Fig. 18. Total cost versus number of channels.

Fig. 17 shows the total cost for the different algorithms when M_2 is 1 to 10. N is 50, M_1 is 5, c^u is 3, and c^L is 3. As M_2 increases, more transmission resources and computing resources are provided, and more MUDs can offload tasks to LEO satellites. Therefore, we can see from Fig. 17 that the total cost decreases with increasing in M_2. In addition, it is easy to find that the JULTO algorithm achieves better performance than the other three comparison algorithms.

Fig. 18 shows the change in the total cost obtained by the four algorithms as c^u and c^L increase from 1 to 10. The total cost obtained by the four algorithms decreases as c^u and c^L increase. This is because c^u and c^L increase, leading to an increase in transmission resources and an increase in MUDs who choose task offloading increases. Thus, the total cost for all MUDs is reduced. When c^u and c^L are less than 9, the total cost obtained by the JULTO algorithm is lower than comparison algorithms. However, when c^u and c^L exceed 9, the costs of the three algorithms (JULTO, ICSOC and CCPM) do not have a large gap. This is because when resources are already sufficient, further increases of c^u and c^L do not affect the performance of all algorithms.

VII. Conclusion

In this paper, the problem of task offloading in UAV-assisted LEO satellite edge computing networks is investigated. We formulate the task offloading problem with the goal of minimizing the total cost of MUDs while satisfying the satellite coverage constraints. We prove that the formulated task offloading problem is NP-hard. Then, we redefine it as the LUTO-Game model, propose the potential function, and prove that the LUTO-Game is a potential game. Next, a distributed JULTO algorithm is proposed to obtain the Nash equilibrium offloading strategy. We then perform the theoretical analysis for the utility of our JULTO algorithm in the worst-case. Finally, we conduct both convergence analysis and comparison experiments to verify the performance of the JULTO algorithm.

An important future research direction is to model the position changes, flight, and hovering of UAVs, taking into account the energy consumption of UAVs and satellites in optimization problems.

References

[1] H. Zhang, Y. Yang, B. Shang, and P. Zhang, "Joint resource allocation and multi-part collaborative task offloading in MEC systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8877–8890, Aug. 2022.
[2] N. Cheng et al., "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[3] J. Mei, L. Dai, Z. Tong, X. Deng, and K. Li, "Throughput-aware dynamic task offloading under resource constant for MEC with energy harvesting devices," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3460–3473, Sep. 2023.
[4] D.-G. Zhang et al., "New computing tasks offloading method for MEC based on prospect theory framework," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 1, pp. 770–781, Feb. 2024.
[5] C. W. Zaw, N. H. Tran, Z. Han, and C. S. Hong, "Radio and computing resource allocation in co-located edge computing: A generalized nash equilibrium model," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2340–2352, Apr. 2023.
[6] Y. Chen, J. Xu, Y. Wu, J. Gao, and L. Zhao, "Dynamic task offloading and resource allocation for NOMA-aided mobile edge computing: An energy efficient design," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1492–1503, Jul./Aug. 2024.
[7] W. Chu, X. Jia, Z. Yu, J. C. Lui, and Y. Lin, "Joint service caching, resource allocation and task offloading for MEC-based networks: A multi-layer optimization approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2958–2975, Apr. 2023.
[8] X. Gao, Y. Sun, H. Chen, X. Xu, and S. Cui, "Joint computing, pushing, and caching optimization for mobile edge computing networks via soft actor-critic learning," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9269–9281, Mar. 2024.
[9] D. Wu, D. Zhang, M. Zhang, R. Zhang, F. Wang, and S. Cui, "ILCAS: Imitation learning-based configuration-adaptive streaming for live video analytics with cross-camera collaboration," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 6743–6757, Jun. 2024.
[10] S. Duan et al., "MOTO: Mobility-aware online task offloading with adaptive load balancing in small-cell MEC," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 645–659, Jan. 2024.
[11] J. Huang, B. Ma, Y. Wu, Y. Chen, and X. Shen, "A hierarchical incentive mechanism for federated learning," *IEEE Trans. Mobile Comput.*, early access, Jul. 04, 2024, doi: 10.1109/TMC.2024.3423399.
[12] J. Huang, F. Liu, and J. Zhang, "Multi-dimensional QoS evaluation and optimization of mobile edge computing for IoT: A survey," *Chin. J. Electron.*, vol. 33, no. 5, pp. 1–16, 2024.
[13] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.
[14] Y. Chen, K. Li, Y. Wu, J. Huang, and L. Zhao, "Energy efficient task offloading and resource allocation in air-ground integrated MEC systems: A distributed online approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 8, pp. 8129–8142, Aug. 2023.

[15] Y. Lyu, Z. Liu, R. Fan, C. Zhan, H. Hu, and J. An, "Optimal computation offloading in collaborative LEO-IoT enabled MEC: A multi-agent deep reinforcement learning approach," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 996–1011, Jun. 2023.
[16] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.
[17] M. Dai, Y. Wu, L. Qian, Z. Su, B. Lin, and N. Chen, "UAV-assisted multi-access computation offloading via hybrid NOMA and FDMA in marine networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 113–127, Jan./Feb. 2023.
[18] Y. Zhang et al., "Packet-level throughput analysis and energy efficiency optimization for UAV-assisted IAB heterogeneous cellular networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 9511–9526, Jul. 2023.
[19] J. Huang, M. Zhang, J. Wan, Y. Chen, and N. Zhang, "Joint data caching and computation offloading in UAV-assisted Internet of Vehicles via federated deep reinforcement learning," *IEEE Trans. Veh. Technol.*, early access, Jul. 18, 2024, doi: 10.1109/TVT.2024.3429507.
[20] J. Ji, K. Zhu, and L. Cai, "Trajectory and communication design for cache- enabled UAVs in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6190–6204, Oct. 2023.
[21] D. Han et al., "Two-timescale learning-based task offloading for remote IoT in integrated satellite–terrestrial networks," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10131–10145, Jun. 2023.
[22] C. Li, Y. Zhang, X. Hao, and T. Huang, "Jointly optimized request dispatching and service placement for MEC in LEO network," *China Commun.*, vol. 17, no. 8, pp. 199–208, 2020.
[23] 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 Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5260–5279, May 2024.
[24] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "UAV-enabled age-optimal data collection in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2019, pp. 1–6.
[25] J. Liu, P. Tong, X. Wang, B. Bai, and H. Dai, "UAV-aided data collection for information freshness in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2368–2382, Apr. 2021.
[26] Q. Wu, M. Cui, G. Zhang, F. Wang, Q. Wu, and X. Chu, "Latency minimization for UAV-enabled URLLC-based mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 23, no. 4, pp. 3298–3311, Apr. 2024.
[27] Z. Han, T. Zhou, T. Xu, and H. Hu, "Joint user association and deployment optimization for delay-minimized UAV-aided MEC networks," *IEEE Wireless Commun. Lett.*, vol. 12, no. 10, pp. 1791–1795, Oct. 2023.
[28] M. Hua, Y. Wang, C. Li, Y. Huang, and L. Yang, "UAV-aided mobile edge computing systems with one by one access scheme," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 664–678, Sep. 2019.
[29] 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 Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6240–6254, Oct. 2020.
[30] Y. Hao, Z. Song, Z. Zheng, Q. Zhang, and Z. Miao, "Joint communication, computing, and caching resource allocation in LEO satellite MEC networks," *IEEE Access*, vol. 11, pp. 6708–6716, 2023.
[31] Y. Jing, J. Wang, C. Jiang, and Y. Zhan, "Satellite MEC with federated learning: Architectures, technologies and challenges," *IEEE Netw.*, vol. 36, no. 5, pp. 106–112, Sep./Oct. 2022.
[32] Y. Wang, J. Zhang, X. Zhang, P. Wang, and L. Liu, "A computation offloading strategy in satellite terrestrial networks with double edge computing," in *Proc. IEEE Int. Conf. Commun. Syst.*, 2018, pp. 450–455.
[33] Y. Wang, J. Yang, X. Guo, and Z. Qu, "A game-theoretic approach to computation offloading in satellite edge computing," *IEEE Access*, vol. 8, pp. 12510–12520, 2020.
[34] S. Zhang, A. Liu, C. Han, X. Liang, X. Xu, and G. Wang, "Multi-agent reinforcement learning-based orbital edge offloading in sagin supporting internet of remote things," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20472–20483, Dec. 2023.
[35] W. Lin, T. Huang, X. Li, F. Shi, X. Wang, and C.-H. Hsu, "Energy-efficient computation offloading for UAV-assisted MEC: A two-stage optimization scheme," *ACM Trans. Internet Technol.*, vol. 22, no. 1, pp. 1–23, 2021.
[36] 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.
[37] M. Wang, L. Zhang, P. Gao, X. Yang, K. Wang, and K. Yang, "Stackelberg-game-based intelligent offloading incentive mechanism for a multi-UAV-assisted mobile-edge computing system," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15679–15689, Sep. 2023.
[38] D. Wang, W. Wang, Y. Kang, and Z. Han, "Distributed data offloading in ultra-dense LEO satellite networks: A stackelberg mean-field game approach," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 112–127, Jan. 2023.
[39] Q. Tang, Z. Fei, B. Li, and Z. Han, "Computation offloading in LEO satellite networks with hybrid cloud and edge computing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9164–9176, Jun. 2021.
[40] H. Liao, Z. Zhou, X. Zhao, and Y. Wang, "Learning-based queue-aware task offloading and resource allocation for space–air–ground-integrated power IoT," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5250–5263, Apr. 2021.
[41] Z. Yang, S. Bi, and Y.-J. A. Zhang, "Online trajectory and resource optimization for stochastic UAV-enabled MEC systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5629–5643, Jul. 2022.
[42] S. Wang et al., "Federated learning for task and resource allocation in wireless high-altitude balloon networks," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17460–17475, Dec. 2021.
[43] Z. Ning et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, May 2023.
[44] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.
[45] P. Lai et al., "Edge user allocation with dynamic qualityof service," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2019, pp. 86–101.
[46] H. Zeng, X. Zhu, Y. Jiang, Z. Wei, and T. Wang, "A green coordinated multi-cell NOMA system with fuzzy logic based multi-criterion user mode selection and resource allocation," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 480–495, Jun. 2019.

Ying Chen (Senior Member, IEEE) received the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2017. She was a joint PhD student with the University of Waterloo, Waterloo, ON, Canada from 2016 to 2017. She is a professor with the Computer School, Beijing Information Science and Technology University, Beijing. Her current research interests include Internet of Things, mobile edge computing, wireless networks and communications, machine learning, etc. She is the recipient of the Best Paper Award with IEEE SmartIoT 2019, the 2016 Google PhD Fellowship Award, and the 2014 Google Anita Borg Award, 2022 Outstanding Contribution Award in 18th EAI CollaborateCom, respectively. She is/was the leading guest editor of *Journal of Cloud Computing*, TPC member of IEEE HPCC, and PC member of IEEE Cloud, CollaborateCom, IEEE CPSCom, CSS, etc. She is also the reviewer of several journals such as the *IEEE Wireless Communications Magazine*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Internet of Things Journal*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*.

Jie Zhao received the MEng degree in computer science and technology from the Beijing Information Science and Technology University, China, in 2024. His current research interests include edge computing, Internet of Things, and Game theory.

Yuan Wu (Senior Member, IEEE) received the PhD degree in electronic and computer engineering from the Hong Kong University of Science and Technology, in 2010. He is currently an associate professor with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China, and also with the Department of Computer and Information Science, University of Macau. His research interests include resource management for wireless networks, mobile edge computing and edge intelligence, and integrated sensing and communications. He received the Best Paper Award from the IEEE ICC'2016, WCSP'2016, IEEE TCGCC'2017, IWCMC'2021, and WCNC'2023. He is currently on the editorial board of *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network Science and Engineering*, and *IEEE Internet of Things Journal*.

Jiwei Huang (Senior Member, IEEE) received the BEng and PhD degrees in computer science and technology from Tsinghua University, in 2009 and 2014, respectively. He was a visiting scholar with the Georgia Institute of Technology. He is currently a professor and the vice-dean with the College of Artificial Intelligence, China University of Petroleum, Beijing, China, and the Director of the Beijing Key Laboratory of Petroleum Data Mining. His research interests include services computing, Internet of Things, and edge computing. He has published one book and more than 70 articles in international journals and conference proceedings, including the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Vehicular Technology*, ACM SIGMETRICS, IEEE ICWS, and IEEE SCC. He is currently on the editorial board of *Chinese Journal of Electronics and Scientific Programming*.

Xuemin Sherman Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a university professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering Foreign member, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT), in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario), in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the technical program committee chair/co-chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the chair for the IEEE ComSoc Technical Committee on Wireless Communications. Dr. Shen is the president of the IEEE ComSoc. He was the vice president for Technical & Educational Activities, vice president for Publications, member-at-large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and member of IEEE fellow Selection Committee of the ComSoc. He served as the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network,* and *IET Communications*.