Delay- and Cost-Aware Dynamic Service Migration in Collaborative Satellite Computing

Weiwei Gao[1(✉)], Ao Zhou[1], Jianing Tang[1], Yuanzhe Li[2], and Shangguang Wang[1]

[1] State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
{weiweigao, aozhou, tangjianing, sgwang}@bupt.edu.cn
[2] Institute for AI Industry Research (AIR), Tsinghua University
liyuanzhe@air.tsinghua.edu.cn

Abstract. The renaissance in aerospace technology positions satellite computing as a promising frontier. Due to the wide coverage of satellites, users in remote areas can request computing services even if there is no terrestrial infrastructure. Most existing studies focus on computation offloading, neglecting the service migration issues that state dependency induces. This paper focuses on user-dependent state information services. Due to the periodic motion of satellites, they will move away from the users. Users may frequently change satellites to achieve a lower delay, but this behavior can cause tremendous network pressure. Addressing this problem poses challenges due to the limited onboard resources and the high computational complexity of optimal node decision-making. Our approach first aims to minimize delay under the constraint of long-term costs. Second, we employ Lyapunov optimization to stabilize long-term costs and decouple the problem into a more manageable single-slot problem. Since the decomposition problem is NP-hard, we use a distributed approximation-based best response update to reduce computational complexity. Additionally, we propose a dynamic programming-based offline service migration algorithm, assuming complete information availability. The simulation results demonstrate the effectiveness of the proposed offline and online algorithms.

Keywords: Satellite computing · Delay sensitive · Service migration · Cost optimization

1 Introduction

With the rapid development of aerospace technology, emerging low-Earth orbit satellite constellations perfectly complement terrestrial networks by providing services to areas beyond terrestrial coverage. Furthermore, advances in Commercial Off-The-Shelf (COTS) hardware have enhanced satellite computing capabilities [2], promoting the development of space intelligence applications such as processing space-native data [10,31], supporting multi-user interactions [28],

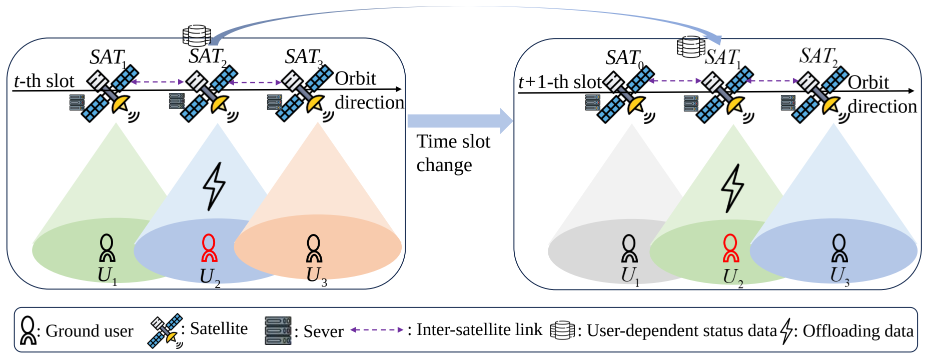


Fig. 1. A scenario of dynamic service migration.

managing content distribution networks [18], and analyzing offloading data for Internet of Things users in remote areas [14, 29].

Figure 1 shows the user-dependent state information task offloading scenario. Taking the ground user U_2 as an example, the satellite SAT_2 provides a service for the user U_2 in the time slot t. Since the motion of the on-orbit satellite is periodic, at time slot $t+1$, satellite SAT_2, which might be located far from user U_2. In this case, user U_2 has three options: (1) continue to obtain services from satellite SAT_2, which may require multi-hop inter-satellite links (ISLs) to access. Option (1) does not require data migration, thus avoiding delays and bandwidth costs brought by migration. However, this option may result in relatively long communication delays. Option (2) transfers the unfinished computation data of user U_2 and migrates the user-dependent state information to another satellite node such as satellite SAT_1 where the service required by U_2 is already deployed. Although option (2) does not require service deployment costs, it incurs migration delays and network bandwidth costs. Option (3) selects a satellite such as SAT_0 that does not have the service deployed but has sufficient computing resources and is in proximity to the user. Option (3) incurs migration delays, network bandwidth costs, and service deployment costs; however, it may reduce computing and transmission delays. Exploring how to provide low-delay services to users through collaboration on satellite edge nodes, while adhering to cost constraints, presents worthwhile research.

This paper investigates the delay- and cost-aware dynamic service migration to ensure service continuity and low delay for ground users in collaborative satellite computing. However, addressing this problem entails the following challenges: First, onboard resources such as bandwidth and computing resources are limited. Users with low delay requirements may access services from nearby computing nodes [15]. For user-dependent state information, migrating this state from one satellite node to another consumes significant bandwidth on the ISLs and depletes the resources of both the source and target satellite computing nodes. Active connections between satellite nodes must be maintained during data migration. Although recent studies have explored delay-aware dynamic service migration in satellite computing [11, 19, 35], they either did not consider inter-satellite collaboration or failed to constrain frequent service migration. Unnecessary or erroneous service migration exerts additional pressure on the system network [32]. This paper addresses both the costs faced by satellite network providers and the low latency requirements of users. Second, high algorithmic complexity. Given the vast scale of satellite constellations, sequentially searching for the optimal computing nodes significantly increases the algorithmic complexity, and improper service migration strategies further burden the network. We formulate this problem as an integer programming problem, known to be NP-hard [27]. Recent methods, including Gibbs sampling [9] and branch and bound techniques [8], address these integer programming challenges. However, these approaches require substantial computing resources and entail lengthy solution times.

To address the first challenge, we apply the Lyapunov optimization theory to ensure that the network costs for the operator remain stable under long-term constraints and decouple the long-term optimization problems into a more manageable single-slot problem. To address the second challenge, we employ a distributed approximation-based best response update technology to reduce computational complexity.

The contributions of this paper are summarized as follows:

- To avoid frequent migrations in pursuit of low delay within collaborative satellite computing, we consider the constraints of long-term service migration costs. Assuming all relevant information is known, we transform this problem into the shortest path problem and propose an offline service migration algorithm based on dynamic programming.
- In the absence of prior information, we initially apply Lyapunov optimization to ensure the stability of long-term costs and decompose the problem into a series of single-slot problems. We adopt a distribution approximation-based best response update to reduce computational complexity.
- The simulation results demonstrate that the proposed offline and online algorithms significantly outperform the benchmarks.

The remainder of this paper is organized as follows: Section 2 summarizes related work. Section 3 describes the models and formulates the problem formulation. Section 4 discusses the algorithm design for offline service migration based on dynamic programming and online service migration based on Lyapunov. Section 5 presents the simulation results. Section 6 is the conclusion and future work.

2 Related Work

In this section, we review related work and categorize it into two primary areas: dynamic service migration in mobile edge computing and dynamic service migration in satellite computing. Detailed analysis follows.

2.1 Dynamic Service Migration in Mobile Edge Computing

Service migration in mobile edge computing is a highly relevant topic, motivated primarily by the limited coverage of the ground network infrastructure and user mobility. Given the unpredictability of ground user mobility patterns, Ksentini et al. [16] analyzed user-specific one-dimensional mobility patterns. For a more realistic approach, Wang et al. [33] formulated the service migration problem as a Markov decision process based on two-dimensional user mobility patterns and devised an optimal service migration strategy. Wen et al. [34] used federated learning to protect user data privacy. In addition, unnecessary or erroneous service migrations place significant pressure on the network. To mitigate this, Liu et al. [22] focused on service quality in a distributed multi-user scenario, employing a counterfactual multi-agent reinforcement learning approach to minimize task completion times within energy constraints. Tuli et al. [32] introduced a preemptive migration technique using generative adversarial networks to predict when to migrate, although it does not determine where to migrate. Conversely, Kim et al. [15] addressed the problem of when and where services would migrate. Under resource limitations, they modeled the problem as an integer linear programming problem and implemented a heuristic solution. Josh Mwasinga et al. [25] utilized a deep Q-network (DQN) to balance quality of experience (QoE) against migration costs. Despite these valuable insights, these approaches are not directly applicable to satellite scenarios. Because the main consideration on the ground is predicting the user's movement trajectory, in the satellite scenario, since the satellite moves much faster than the user, we can assume that the user's geographical location is stationary.

2.2 Dynamic Service Migration in Satellite Computing

With commercial COTS hardware's advancement, satellite computing capabilities will increase. Users in remote areas without ground infrastructure offload data to satellites within sight for processing [4, 29, 38]. Furthermore, to achieve inter-satellite load balance, some works proposed the use of inter-satellite assisted computing to complete the offloading task [6, 36, 37]. These studies assume that the service is already deployed on the satellite. Therefore, when a user switches to the satellite computing node, there is no need to migrate the user-dependent state data. Recently, some researchers have begun to focus on the service migration problem in satellite computing. Deng et al. [11] proposed a bandwidth-aware service migration problem aimed at minimizing the long-term total delay of the system, under the assumption that all tasks meet their deadlines. Han et al. [12] utilized real-time traffic data from the satellite network and performed service migration in response to changes in uplink and downlink traffic. However, the above studies did not consider potential collaboration between satellites. We must fully leverage computing resources across satellites to increase the efficiency of on-orbit satellite resources.

Unlike the work mentioned above, our work concentrates on inter-satellite cooperative service migration. To prevent frequent migrations by users seeking low latency, this paper considers long-term migration costs as a constraint to minimize the delay. Depending on the availability of the required information, we propose two approaches: an offline service migration method based on dynamic programming and an online service migration method using Lyapunov optimization.

3 Models and Problem Formulation

3.1 Scenario Model

We consider a scenario in an integrated satellite-terrestrial network, as shown in Fig. 1. In this scenario, ground users in remote areas lack access to ground networks due to the absence of ground infrastructure. Furthermore, resource limitations prevent these users from processing computing tasks locally, necessitating reliance on satellites equipped with servers to provide network and computing services. For example, in the power grid system, to respond to fluctuations in energy demand, grid data must be regularly offloaded to satellites for analysis. This process enables automatic adjustments to the power supply during peak hours or the initiation of demand response strategies. Each ground user offloads tasks to a satellite within its coverage range through a wireless link in the Ka-band [30]. Without loss of generality, we assume that each user initially chooses the satellite closest to the distance. Additionally, considering the satellite's orbital speed of 7.3 km/s [3], the satellite may move out of the user's coverage range over time. To ensure service continuity and meet the low delay requirements of these ground users, it may be necessary to migrate the computing service to synchronize user-dependent status information.

In our scenario, we assume a constellation of $\mathcal{S} = \{1, 2, \ldots, s, \ldots, S\}$ satellites, where S consists of P orbital planes, each containing N satellites. Define $\mathcal{U} = \{1, 2, \ldots, u, \ldots, U\}$ as the set of all remote ground users. Let $D = \{d_s^t\}$ $(s \in \mathcal{S}, t \in \mathcal{T})$ represent the deployment of the service in the satellite network, with d_s^t indicating whether the required service is deployed on the satellite node s during the time slot t. If the required service is deployed on the satellite node s at time t, then $d_s^t = 1$; if it is not deployed, then $d_s^t = 0$. Let $x_{u,s}^t$ represent the binary decision variable for user u selecting the satellite node s for computation at time slot t. Users' computing satellite node selection decision can be denoted by $X^t = (X_1^t, \ldots, X_u^t, \ldots, X_U^t), X_u^t = (x_{u,1}^t, \ldots, x_{u,s}^t, \ldots, x_{u,S}^t)$. We assume that in each time slot t, each user is served by exactly one satellite node. Therefore, for user u, the decision to select a computing satellite node has the following constraints:

$$\sum_{s=1}^{S} x_{u,s}^t = 1, \forall u \in \mathcal{U}, t \in \mathcal{T}, \tag{1}$$

$$x_{u,s}^t \in \{0, 1\}, \forall u \in \mathcal{U}, s \in \mathcal{S}, t \in \mathcal{T}. \tag{2}$$

Additionally, to prevent satellite overloading, the number of ground users served by satellite s during time slot t must not exceed M_s. Consequently, the constraint can be established:

$$\sum_{u=1}^{U} x_{u,s}^{t} \leq M_s, \forall s \in \mathcal{S}, t \in \mathcal{T}. \tag{3}$$

3.2 Visibility Model

A ground user can only communicate with an LEO satellite when there is line-of-sight visibility between them [23]. We use binary variables $V_{u,s}^{t}$ to indicate whether ground user u and satellite s are visible to each other in time slot t; it is 1 if both are visible and 0 otherwise. Denote $\mathcal{U}_s \subset \mathcal{U}$ as the set of ground users within the coverage area of the satellite s. Similarly, $\mathcal{S}_u \subset \mathcal{S}$ represents the set of satellites to which the ground user u can connect. Given the predictable orbits of the satellites, we can utilize the *StarPerf* software to simulate the coverage of specific users by the satellites in each time slot [17].

3.3 Network Model

We use a time-varying graph to represent the dynamic nature of satellite-terrestrial integrated networks. The links between satellites and ground users can change in minutes, whereas the network topology of the entire satellite constellation remains relatively stable over short periods. Therefore, we discretize time, representing the network topology within a time slot $t \in \mathcal{T} = \{1, 2, \ldots, t, \ldots, T\}$ by a graph $G_t = (V, E_t)$, where T, the mission period, is sufficiently long to guarantee service periods for each ground user. The vertex set V includes all satellites and geographically distributed ground users requesting services from the satellites. The edge set E_t denotes the connectivity between vertices for time slot t. An edge $e_{i,j}^{t} \in E_t$, where $i, j \in V$, signifies an accessible link between node i and node j in time slot t, potentially representing either an ISL or a user-satellite link (USL). Moreover, the well-known cross-grid (+grid) connection method is used between satellites [13]. Each satellite is connected to its four neighboring satellites: two in the same orbit and two to its left and right.

3.4 Task Model

In each time slot interval $\triangle t$, ground user u offloads a task by choosing an access satellite in the shortest distance, and then the access satellite routes the task data to the selected satellite computing node through the ISLs. The task from ground user u is represented by a 3-tuple $(p_u^t, c_u^t, T_u^{t,max})$, where p_u^t denotes the task size in bits, c_u^t represents the CPU cycles required to process one bit, and $T_u^{t,max}$ indicates the maximum tolerable delay.

3.5 Delay Performance

Computation Delay We assume that each satellite is equipped with a server that can provide computing resources to tasks from different ground users simultaneously. The computation delay of user u is given by:

$$T_u^{t,comp} = \sum_{s=1}^{S} x_{u,s}^t \frac{\sum_{u=1}^{U} x_{u,s}^t p_u^t c_u^t}{f_s^t}, \forall u \in \mathcal{U}, t \in \mathcal{T}, \tag{4}$$

where f_s^t is the computing capacity (in CPU cycles per second) of satellite s.

Communication Delay The communication delay between the ground user u and the computing satellite node s consists of two parts: the transmission and propagation delays from the ground user u to the access satellite s_0, and from s_0 to the computing satellite node s. For the wireless transmission from ground user u to access satellite s_0, let h_{u,s_0} be the channel gain between user u and access satellite s_0, and let p_u be the transmission power of user u. Denote B_{Ka} is the total bandwidth of the Ka-band, and let N denote the additive white Gaussian noise (AWGN) at satellite s_0. The transmission rate from user u to access satellite s_0 can be expressed as [4]:

$$R_{u,s_0}^t = \frac{B_{Ka}}{\sum_{u=1}^{U} x_{u,s_0}^t} \log_2 (1 + \frac{p_u \left| h_{u,s_0} \right|^2}{N}), \forall u \in \mathcal{U}, s_0 \in \mathcal{S}. \tag{5}$$

If the access satellite s_0 is not a computing node, the task is offloaded to the computing satellite node s through the ISLs. Then, the transmission rate from the access satellite s_0 to the computing satellite s can be expressed by the following formula [37]:

$$R_{s_0,s}^t = \frac{p_{s_0} L_{s_0,s} G_{s_0}^{TR} G_s^{RE}}{kT \left[E/(N' + N_0) \right] \cdot M}, \forall s, s_0 \in \mathcal{S}, t \in \mathcal{T}, \tag{6}$$

where p_{s_0} represents the transmission power of access satellite s_0, $G_{s_0}^{TR}$ and G_s^{RE} represent the transmission antenna gain of access satellite s_0 and the receiving antenna gain of computing satellite s, respectively. The constant k denotes the Boltzmann constant, T is the noise temperature, and $\frac{E}{N'+N_0}$ is the required ratio of received energy per bit to a specific noise density. M represents the link margin and $L_{s_0,s} = \left(\frac{c}{4\pi d_{s_0,s} f} \right)^2$ represents the loss of the ISL path, where c is the speed of light, f is the carrier frequency, and $d_{s_0,s}$ is the distance from the access satellite s_0 to the computing satellite s. If $s_0 = s$, then $d_{s_0,s} = 0$.

Given the service request size, ground user, and satellite constellation location information in each time slot t, the transmission delays $T_{u,s_0}^{t,trans}$ from ground user u to access satellite s_0 and the transmission delays $T_{s_0,s}^{t,trans}$ from access satellite s_0 to computing satellite node s can be denoted as the following equations respectively:

$$T_{u,s_0}^{t,trans} = \frac{p_u^t}{R_{u,s_0}^t}, \forall u \in \mathcal{U}, s_0 \in \mathcal{S}, t \in \mathcal{T}, \tag{7}$$

$$T_{s_0,s}^{t,trans} = \frac{p_u^t}{R_{s_0,s}^t}, \forall s, s_0 \in \mathcal{S}, t \in \mathcal{T}, \tag{8}$$

if $s_0 = s$, then the transmission delay $T_{s_0,s}^{t,trans} = 0$.

The transmission delay from user u to the computing satellite node s can be calculated using the following formula:

$$T_{u,s}^{t,trans} = T_{u,s_0}^{t,trans} + T_{s_0,s}^{t,trans}. \tag{9}$$

The propagation delay for user u to offload the task to access satellite s_0 can be expressed as:

$$T_{u,s_0}^{t,prop} = d_{u,s_0}/c, \tag{10}$$

where d_{u,s_0} represents the distance between the ground user u and the access satellite s_0 and can be represented as:

$$d_{u,s_0} = \sqrt{R^2 + (R+H)^2 - 2R(R+H) \cdot cos\gamma}, \tag{11}$$

where R represents the radius of the earth, H represents the orbital altitude of the satellite, and γ represents the geocentric angle.

Assume that the path of the user u's computing task through the satellite node i is expressed as $N_u = \left\{q_u^i \mid i \in \{1, \ldots, i, \ldots, I\}\right\}$. Then, the propagation delay between the access satellite s_0 and the computing satellite s can be denoted as:

$$T_{s_0,s}^{t,prop} = \sum_{i=1}^{I-1} T_{q_u^i q_u^{i+1}}^{t,prop}, \tag{12}$$

where $T_{q_u^i q_u^{i+1}}^{t,prop}$ is the propagation delay of one hop along q_u^i.

The propagation delays from user u to the computing satellite node s can be calculated using the following formula:

$$T_{u,s}^{t,prop} = T_{u,s_0}^{t,prop} + T_{s_0,s}^{t,prop}. \tag{13}$$

Since the size of the computation results is much smaller than the data size of the task input and the downlink rate is significantly higher than the uplink rate, the transmission delay from the computing satellite back to the user can be ignored [38]. However, due to the considerable distance between satellites and users, the propagation delay of the satellite-user link cannot be ignored. Assume that in time slot t, the routing path $E_{s,u}^t = \left\{e_{s,i_1}^t, e_{i_1,i_2}^t, \ldots, e_{i_{k-1},i_k}^t, e_{i_k,m}^t, e_{m,u}^t\right\}$ for transmitting the computing result from satellite s to user u passes through user access satellite m. Let $D_{E_t} = \left\{d_{i,j}^t \mid i,j \in V, i \neq j, e_{i,j}^t \in E_t\right\}$ represent the distance between node i and node j in the graph at time slot t. Therefore, the propagation delay of the result from the computing satellite s to the user u is given by:

$$T_{s,u}^{t,prop} = \sum_{e_{i,j}^t \in E_{s,u}^t} \frac{d_{i,j}^t}{c}. \tag{14}$$

The communication delay from ground user u to computing satellite node s can be expressed as follows:

$$T_{u,s}^{t,comm} = T_{u,s}^{t,trans} + T_{u,s}^{t,prop} + T_{s,u}^{t,prop}. \tag{15}$$

When considering the satellite node selection decision, the communication delay experienced by ground user u can be further described as follows:

$$T_u^{t,comm} = \sum_{s=1}^{S} x_{u,s}^t T_{u,s}^{t,comm}, \forall u \in \mathcal{U}, t \in \mathcal{T}. \tag{16}$$

3.6 Migration Delay and Cost Model

When the user changes the selected satellite computing node, additional delays and costs are incurred due to service migration and potential service deployment. We have completed the models for computation and communication. Next, we introduce the migration delay model and the migration cost model.

Migration Delay Model Assume that the task p_u^t is processed on satellite i in the time slot $t+1$. If the required service is not deployed on satellite i, that is, $d_i^t = 0$, the service must be quickly deployed. Subsequently, the unfinished computing tasks and user-dependent state data must be migrated from satellite s to satellite i to synchronize service status. In this paper, we adopt live container migration technology [1]. The container itself does not store the service configuration files for the applications within it. Checkpoint and Restore in Userspace (CRIU) technology can save the running status information of specified processes in the system to the service configuration files. Based on these files, the running status of the original process can be restored on other satellite nodes. During the computing migration process, only the period when the service is not running affects user-perceived latency. Therefore, we only consider periods when the service is not operational, hereafter referred to as downtime. After the migration is completed, unfinished computing tasks can continue to be processed on the satellite i. In our previous computation model, we took into account the computing delay of the entire task on the satellite node, including the delay of the unfinished task. Therefore, after the task is migrated to a new satellite node, the delay of the unfinished part must be subtracted from the previous total computing delay, and the delay of this unfinished part must be added to the new satellite node.

At time slot t, user u's unfinished tasks size can be expressed as follows:

$$p_u^{t,mig} = min\left\{p_u^t, max\left\{0, p_u^t(1 - \frac{\triangle t - T_u^{t,comm}}{T_u^{t,comp}})\right\}\right\}. \tag{17}$$

Therefore, the migration delay of user u can be expressed as follows:

$$\begin{aligned} T_u^{t,mig} = &\sum_{s=1}^{S} x_{u,s}^t \sum_{i=1}^{S} x_{u,i}^{t+1} \theta_t \delta_{s,i} + \sum_{i=1}^{S} x_{u,i}^{t+1} \frac{\sum_{u=1}^{U} x_{u,i}^{t+1} p_u^{t,mig} c_u^t}{f_i^{t+1}} \\ &- \sum_{s=1}^{S} x_{u,s}^t \frac{\sum_{u=1}^{U} x_{u,s}^t p_u^{t,mig} c_u^t}{f_s^t}, \forall u \in \mathcal{U}, \end{aligned} \tag{18}$$

where θ_t represents the downtime for computing migration, $\delta_{s,i}$ is the Kronecker delta function that describes whether satellite i and satellite s are the same satellite. The function is defined as $\delta_{s,i} = \begin{cases} 1, s \neq i, \\ 0, s = i, \end{cases} \forall s, i \in S$.

Cost Model Although satisfactory service quality is provided to users by migrating service profiles between satellite nodes to adapt to the periodic movement of satellites, service migration between satellite nodes incurs additional costs. For example, migrating across satellite nodes results in heavy network bandwidth usage due to the data of users' outstanding tasks at time slot t and each user's service profile, in addition to the deployment cost of new services.

Migration Cost Let $C_{u,s,i}^{t}$ represent the migrating cost of user u' data from the source satellite computing node s to the destination satellite computing node i in time slot t, and we use the product of the migration data size and the propagation delay from satellite s to satellite i. Then the migration cost at time t can be expressed as follows:

$$C_u^{t,mig} = \sum_{s=1}^{S} x_{u,s}^{t} \sum_{i=1}^{S} x_{u,i}^{t+1} C_{u,s,i}^{t,mig} \delta_{s,i}, \forall u, i \in U, \tag{19}$$

where $C_{u,s,i}^{t,mig} = p_u^{t,mig} \times T_{u,i}^{t,prop}$. Without losing generality, we assume that $C_{u,s,i}^{t} = 0, \forall s = i$.

Service Deployment Cost If the service is not deployed on satellite node i, that is $d_i^t = 0$, the service deployment cost is:

$$C_u^{t,deploy} = \chi_{\{d_i^t = 0\}} \theta, \tag{20}$$

where $\chi_{\{x\}}$ is an indicative function. If the event x is true, then $\chi_{\{x\}} = 1$; otherwise, it is equal to 0. θ is a positive number that represents the cost of service deployment.

The total cost for user u in time slot t is:

$$C_u^t = C_u^{t,mig} + C_u^{t,deploy}. \tag{21}$$

Due to the scarcity of on-orbit communication resources, we introduce C_u^{arg} to represent the long-term cost budget across T time slots, which prevents users from frequently migrating services in pursuit of low latency and is subject to the following constraints:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} C_u^t \leq C_u^{avg}, \forall u \in \mathcal{U}. \tag{22}$$

3.7 Problem Formulation

The total delay of user u in time slot t can be expressed as follows:

$$T_u^t = T_u^{t,comm} + T_u^{t,comp} + T_u^{t,mig}. \tag{23}$$

The total delay for all users is:

$$F(t) = \sum_{u=1}^{U} T_u^t. \tag{24}$$

To optimize the average user response time within the cost budget, the dynamic service migration problem can be formulated as:
$\mathcal{P}_1$:

$$\min_{X^1,\dots,X^\infty} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} F(t) \tag{25}$$

$$s.t. \quad T_u^{t,comp} + T_u^{t,comm} \le T_u^{t,max}, \forall t \in \mathcal{T}, \tag{25a}$$

$$(1) - (3), (22). \tag{25b}$$

However, solving problem $\mathcal{P}_1$ requires detailed information about satellite nodes and the specific service requirements of each user in each time slot. This includes the parameters of the user's mission, the computing capabilities of all satellite nodes, and the real-time status of the satellite network, making it extremely challenging to obtain the optimal solution directly. In response, we present two service migration strategies based on the completeness of the information. The first is an offline service migration strategy, which accesses complete future information in advance. The second is an online service migration strategy, which only knows the information of the current time slot. We provide a detailed introduction to two strategies in Section 4.

4 Algorithm Design

4.1 Dynamic Programming-based Offline Service Migration

In this section, we assume that complete data with all relevant information is available. This section focuses on the offline service migration problem based on this assumption. Consider a certain time period T, which includes consecutive moments $(t_0, \dots, t_0 + T - 1)$. Within this period, the offline service migration problem can be expressed as:
$\mathcal{P}_2$:

$$\min_{X^{t_0},\dots,X^{t_0+T-1}} \sum_{t=t_0}^{t_0+T-1} F(t) \tag{26}$$

$$s.t. \quad \sum_{s=1}^{S} x_{u,s}^{t} = 1, \forall u \in \mathcal{U}, t \in \{t_0, \ldots, t_0 + T - 1\}, \tag{26a}$$

$$x_{u,s}^{t} \in \{0, 1\}, \forall t \in \{t_0, \ldots, t_0 + T - 1\}, \tag{26b}$$

$$\sum_{u=1}^{U} x_{u,s}^{t} \le M_s, \forall s \in \mathcal{S}, t \in \{t_0, \ldots, t_0 + T - 1\}, \tag{26c}$$

$$\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} C_u^t \le C_u^{avg}, \forall t \in \{t_0, \ldots, t_0 + T - 1\}. \tag{26d}$$

As shown in Fig. 2, a virtual graph illustrates the transformation of the offline service migration problem into a shortest path problem. The problem $\mathcal{P}_2$ can be equivalently transformed into a problem where $F(t)$ constitutes the weighted shortest path problem. The figure depicts the starting node as the initial state, while each subsequent node represents potential states of satellite computing nodes that the user may select in different time slots. In the time slot t_0, when calculating the weight of X^{t_0}, the state X^{t_0-1} of the previous moment is known and fixed, so the weight depends solely on X^{t_0}. The node D is a virtual node created to ensure the identification of the shortest path, with the weight of the edge connected to the node D set to 0. The shortest path corresponds to the optimal service migration strategy, and the sum of the shortest path weights represents the minimum value of the objective function, that is, the optimal value. We use the Dijkstra algorithm to solve the shortest path problem described above.

4.2 Lyapunov Optimization-based Online Service Migration

First, we employ the Lyapunov optimization technique to transform the long-term optimization problem into a series of short-term or real-time optimization problems. By introducing drift-plus-penalty terms, decisions can be made even in the absence of complete future information. We then use the best response update method to solve the real-time optimization problem of the optimal selection of satellite computing nodes in a given time slot.

Decoupling Long-Term Optimization Problems Using the Lyapunov Optimization Algorithm The available information is incomplete in most scenarios. Therefore, optimal long-term strategies must be continuously adjusted based on system dynamics, including satellite mobility, satellite-ground network conditions, and unstable service request patterns. Lyapunov optimization can adapt to the dynamic changes of the system. First, we define a virtual queue for each user u:

$$\Theta_u(t+1) = \Theta_u(t) + max\left\{C_u^t - C_u^{avg}, 0\right\}, \tag{27}$$

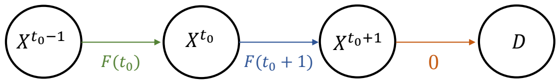


Fig. 2. Transformation of shortest path problem when T=2.

where $\Theta_u(t)$ represents the queue backlog length of user u at time slot t, and $\Theta_u(0) = 0$. The quadratic Lyapunov function is: $\mathcal{L}(\Theta(t)) = \frac{1}{2}\sum_{u=1}^{U}(\Theta_u(t))^2$. $\mathcal{L}(\Theta(t))$ is a scalar value representing the number of queues in all queues. Smaller values of $\mathcal{L}(\Theta(t))$ indicate a lower risk of overflow for queues and therefore higher stability of the system. To further enhance the system's stability, we define the one-slot conditional Lyapunov drift function, which is defined as follows:

$$\triangle(\Theta(t)) = \mathbb{E}\left[\mathcal{L}(\Theta(t+1)) - \mathcal{L}(\Theta(t))|\Theta(t)\right], \tag{28}$$

where $\Theta(t) = (\Theta_1(t), \Theta_2(t), \ldots, \Theta_u(t), \ldots, \Theta_U(t))$. If the drift function $\triangle(\Theta(t))$ converges to zero, it indicates that the long-term queue backlog constraint (21) is satisfied, thus ensuring the stability of the queue. At any time slot t, regardless of the satellite computing node selection strategy employed, the Lyapunov drift satisfies the following inequality:

$$\triangle(\Theta(t)) \leq \triangle_\Theta + \mathbb{E}\left[\sum_{u=1}^{U}\Theta_u(t)(C_u^t - C_u^{avg})|\Theta(t)\right], \tag{29}$$

where $\triangle_\Theta = \frac{1}{2}\sum_{u=1}^{U}(C_u^t - C_u^{avg})^2$ for each time slot t. To optimize system performance, we define the instantaneous objective function $F(t)$ as a single-slot Lyapunov penalty, comprising the sum of task delay. We now adopt the Lyapunov drift-plus-penalty framework to solve problem $\mathcal{P}_1$. First, we give the upper limit of the drift-plus-penalty:

$$\begin{aligned}\triangle(\Theta(t)) + V\cdot\mathbb{E}\left[F(t)|\Theta(t)\right] &\leq \triangle_\Theta + V\cdot\mathbb{E}\left[F(t)|\Theta(t)\right] + \\ &\mathbb{E}\left[\sum_{u=1}^{U}\Theta_u(t)(C_u^t - C_u^{avg})|\Theta(t)\right],\end{aligned} \tag{30}$$

where the non-negative parameter V serves as the trade-off factor between queue stability and system delay. The lower the value of V, the greater the emphasis on system stability.

Problem $\mathcal{P}_1$ can be further transformed into minimizing the upper bound of drift-plus-penalty:

$\mathcal{P}_3$:

$$\min_{X^t}\sum_{u=1}^{U}\Theta_u(t)C_u^t + VF(t) \tag{31}$$

$$s.t. \quad (1)-(3),(24a). \tag{31a}$$

Algorithm 1 Online Satellite Computing Node Selection

```
1: Initialize: $\Theta_u(0) = 0, \forall u \in U$.
2: for each time slot $t = 1, 2, \ldots, \infty$ do
3:     Find $\left(\mathbf{X}^t\right)^*$ for problem $\mathcal{P}3$ by Algorithm 2.
4:     Update the length of virtual queues $\Theta_u(t+1)$ based on $\left(\mathbf{X}^t\right)^*$ according to (27).
5: end for
```

So far, we have transformed the long-term optimization problem $\mathcal{P}_1$ into a real-time optimization problem $\mathcal{P}_3$. Algorithm 1 outlines the implementation of the online satellite computing node selection algorithm. In each time slot t, by solving $\mathcal{P}_3$, we achieve a near-optimal satellite computing node selection and subsequently update the virtual queue of users served by the satellite to prepare for computing in the next time slot. Given that the objective function of $\mathcal{P}_3$ is non-linear and the variables are binary, $\mathcal{P}_3$ is classified as a non-linear integer programming problem, which is typically NP-hard [20]. We employ the best response update method to overcome this challenge and secure a solution within each time slot.

Using Best Response Update Method to Solve a Single-Slot Optimization Problem First, we denote the objective function of problem $\mathcal{P}_3$ as $O\left(\mathbf{X}^t\right)$, where $\mathbf{X}^t \in \mathcal{X}^t$ represents a feasible service migration strategy. Due to the large scale of the in-orbit satellite constellation, we employ distributed best response update technology to reduce the running time of service migration decisions and achieve faster search for service migration nodes. In the distributed service policy update, each user $u \in U$ typically adopts a greedy approach, using the best response update to optimize its migration decision deterministically. This method significantly reduces time complexity, primarily because it leverages users' individual effective decisions instead of conducting random decision exploration. According to the computing delay Eq. (4), it is evident that resource competition among multiple users will affect service performance. Inspired by the application of non-cooperative games in computation offloading in mobile edge computing environments [5], we transform problem $\mathcal{P}_3$ into a congestion game [7] with cost functions among users. Let $\mathbf{X}_{-u}^t = \left\{X_1^t, \ldots, X_{u-1}^t, X_{u+1}^t, \ldots, X_U^t\right\}$ denote the service migration decisions of all users except user u in time slot t. The service migration problem faced by user u involves selecting a suitable satellite computing node with the goal of minimizing the objective function, which consists of delay and migration cost, i.e.,

$$X_u^t = \arg\min_{X_u^t} O_u\left(X_u^t, \mathbf{X}_{-u}^t\right), \forall u \in \mathcal{U}. \tag{32}$$

When each migration decision is executed in a manner acceptable to users, the service migration decision reaches a Nash equilibrium. To simplify the decision process, we propose a method based on the best response update [24].

Definition 1 Given the service migration decisions of all other users $\mathbf{X}_{-u}^t$, if $O_u((X_u^t)^*, \mathbf{X}_{-u}^t) \le O_u(X_u^t, \mathbf{X}_{-u}^t)$, the service migration decision for user u is the best response update [26].

Specifically, assume that when a user sends a service request to a connected satellite node, the system assigns a unique ID to the service. Subsequently, we update all service migration decisions in the random order of the IDs assigned to the services. For user u, who has the smallest current index in the list of users to be updated, we assign it to a preferred satellite node to minimize its current cost as defined by Definition 1. Therefore, we can formulate the following service migration decision update process:

$$\begin{aligned} X_u^t(\lambda+1) = \arg\min O_u \big(X_u^t, \big\{X_1^t(\lambda+1), \ldots, \\ X_{u-1}^t(\lambda+1), X_{u+1}^t(\lambda), \ldots, X_U^t(\lambda)\big\}\big), \end{aligned} \tag{33}$$

where λ is response update round. The pseudocode is shown in Algorithm 2.

Algorithm 2 Best Response Update Based Service Migration Policy Search

```
1: Input: $\mathcal{P}_3$
2: Output: Service migration decision $(\mathbf{X}^t)^*$.
3: Initialization: Initialize the service migration decisions $\mathbf{X}^t(0) =$
   $(X_1^t(0), X_2^t(0), \ldots, X_U^t(0))$ as randomly assigning satellite node for each user
   and the response update iteration round as $\lambda = 0$.
4: while $\mathbf{X}^t(\lambda)$ does not reach a Nash equilibrium do
5:    for user $u = 1$ to $U$ do
6:       Select the proper satellite computing node where user $u$ can minimize its own
         cost according to (33) and gain the corresponding migration decision $X_u^t$.
7:    end for
8:    Update $\lambda = \lambda + 1$
9:    Update $X^t(\lambda+1) = \big(X_1^t(\lambda+1), \ldots, X_U^t(\lambda+1)\big)$
10: end while
11: Return the service migration decision $(\mathbf{X}^t)^*$.
```

Next, we analyze the computational complexity of Algorithm 2. From line 5 to line 7, the system updates the service migration decisions of all U users for each update iteration. Since each update involves a sorting operation, and considering that there are S satellites, the complexity of each sorting operation is $O(\log S)$. Consequently, the complexity of each iteration is $O(US \log S)$. Assuming the algorithm requires I iterations to converge to the Nash equilibrium, the total time complexity of Algorithm 2 is $O(IUS \log S)$.

5 Simulation

5.1 Simulation Settings

We randomly select 100 ground stations from the open-source SatNOGS project as our ground users [21]. In the simulation, we configure a constellation of 288

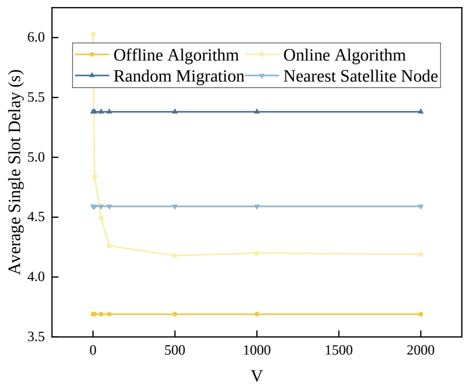


Fig. 3. Average single slot delay under values of V.

satellites distributed across 12 orbits, with each orbit containing 24 satellites. The orbital altitude of these satellites is 550 km, and their inclination is 53°. The task size for each time interval follows a uniform distribution of [$5e6$, $1e7$] bits. The number of CPU cycles required for each bit task is 1000. The transmission rate of an ISL follows a uniform distribution of [100M, 300M] bits / s, while the computational intensity of the satellite follows a uniform distribution of [$1e9$, $2.4e9$] megacycles / s. The inter-satellite transmission power is set to 0.1W, The noise power is $2e^{-26}$, and the energy consumption constant k is 10^{-26}.

The performance of our proposed offline and online algorithms is compared with two benchmarks:

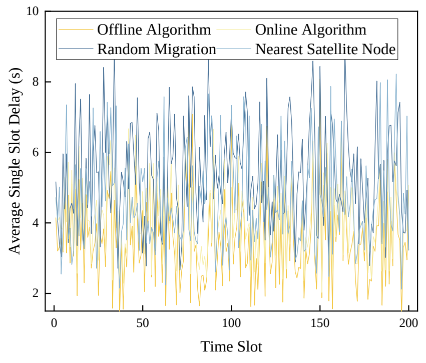
Random Migration This method randomly selects a new satellite node to perform the task when the user's nearest satellite changes.

Nearest Satellite Node This method consistently selects the nearest satellite node to the user to perform the task. It deploys services on demand based on the satellite node's service deployment status.

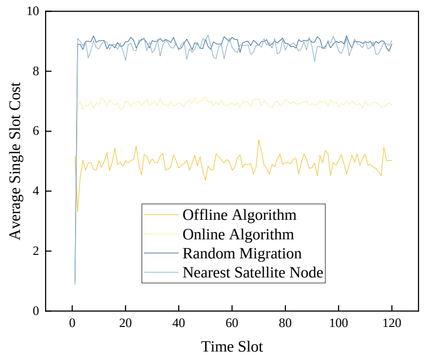
5.2 Simulation Results

Figure 3 shows the average delay variation of four algorithms under different Lyapunov optimization control parameter V values. We can observe that as the V value increases from 1 to 2000, the average delay gradually decreases in the online algorithm. This phenomenon indicates that a larger V value increases the emphasis on delay, leading to a gradual reduction in delay.

Figs. 4(a) and 4(b) show the average delay and the average cost performance of four algorithms in different time slots, respectively. The offline method achieves the best results in terms of both average delay and average cost due to its full use of future system information. The performance of the online method

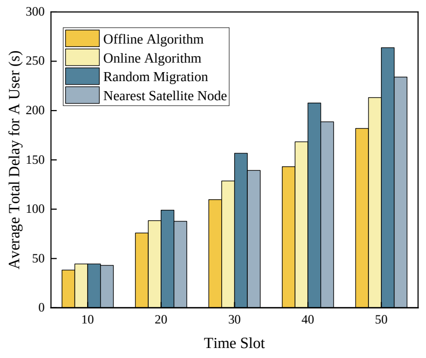


(a) Average delay v.s. Time slot

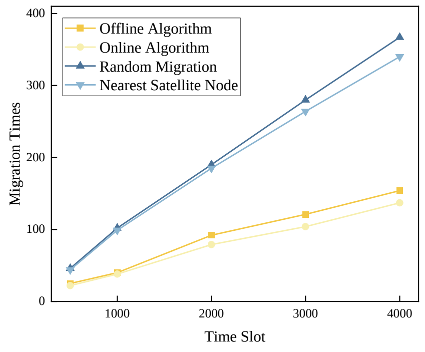


(b) Average cost v.s. Time slot

Fig. 4. Average delay and average cost of each algorithm in different time slots.



(a) Total delay v.s. Time slot



(b) Migration times v.s. Time slot

Fig. 5. Relationship between delay and migration times in different time slots.

is closest to that of the offline method because the online method considers the long-term payoff of current decisions. What's more, the average delay varies with each user, primarily due to fluctuations in the overall computing resource usage of the satellite at different times. The Random Migration and the Nearest Satellite Node perform similarly in terms of cost, the latter outperforms the former in terms of delay because the latter usually has lower transmission latency.

Figure 5 illustrates the relationship between delay and migration times. Fig. 5(a) illustrates the total delay experienced by each user over a period of time, while Fig. 5(b) displays the total migration times within 4000-time slots. Frequent migrations (such as Random Migration) and infrequent migrations can increase the total delay in completing tasks. In contrast, the proposed online and offline algorithms achieve a more reasonable number of migrations through optimization, effectively reducing the total delay in completing user tasks.

6 Conclusion and Future Work

Given the dynamic nature of the satellite network and user-dependent state information services, this paper proposes a service migration strategy adapted to the movement of satellites to ensure the continuity of service for the ground user. Firstly, to balance low delay and low cost, we consider minimizing service response time under long-term cost constraints. Secondly, based on the availability of information, we differentiate between two service migration scenarios: one is the offline algorithm with complete future details, and the other is the online algorithm with only current information available. For the offline service migration algorithm, we transform it into the shortest path problem and use dynamic programming to solve it. For online service migration, we initially employ the Lyapunov optimization algorithm to decompose the long-term optimization problem into a real-time, single-time slot optimization problem that is NP-hard. We use the best response update method to determine the optimal satellite computing node selection for each time slot and then apply a distributed approximation-based best response update to reduce computational complexity. The simulation results indicate that, compared to other baseline methods, the Lyapunov-based online service migration strategy significantly reduces the latency of the service response, closely approaching the performance of the offline algorithm. In future work, it will be interesting to investigate the energy consumption of satellites when making migration decisions because on-orbit power resources are scarce.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (U21B2016, 62032003, 62302262) and in part by the Fundamental Research Funds for the Central Universities of China under Grant 2024ZCJH11.

References

1. Benjaponpitak, T., Karakate, M., Sripanidkulchai, K.: Enabling live migration of containerized applications across clouds. In: 2020 IEEE Conference on Computer Communications (INFOCOM). pp. 2529–2538 (2020)
2. Bhattacherjee, D., Kassing, S., Licciardello, M., Singla, A.: In-orbit computing: An outlandish thought experiment? In: Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets). pp. 197–204 (2020)
3. Bhattacherjee, D., Singla, A.: Network topology design at 27,000 km/hour. In: Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies (CoNEXT). pp. 341–354 (2019)
4. Cao, X., Yang, B., Shen, Y., Yuen, C., Zhang, Y., Han, Z., Poor, H.V., Hanzo, L.: Edge-assisted multi-layer offloading optimization of LEO satellite-terrestrial integrated networks. IEEE J. Sel. Areas Commun. **41**(2), 381–398 (2022)
5. Chen, J., Deng, Q., Yang, X.: Non-cooperative game algorithms for computation offloading in mobile edge computing environments. J. Parallel Distrib. Comput. **172**, 18–31 (2023)

6. Chen, Q., Meng, W., Quek, T.Q., Chen, S.: Multi-tier hybrid offloading for computation-aware iot applications in civil aircraft-augmented sagin. IEEE J. Sel. Areas Commun. **41**(2), 399–417 (2022)
7. Chen, X., Huang, J.: Database-assisted distributed spectrum sharing. IEEE J. Sel. Areas Commun. **31**(11), 2349–2361 (2013)
8. Chen, Y., Zhang, Q., Zhang, Y., Ma, X., Zhou, A.: Energy and time-aware inference offloading for dnn-based applications in leo satellites. In: 2023 IEEE 31st International Conference on Network Protocols (ICNP). pp. 1–6 (2023)
9. Chen, Y., Ma, X., Zhou, A., Wang, S.: Cooperative content caching and distribution for satellite cdns. In: 2023 IEEE 31st International Conference on Network Protocols (ICNP). pp. 1–6 (2023)
10. Denby, B., Lucia, B.: Orbital edge computing: Nanosatellite constellations as a new class of computer system. In: Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). pp. 939–954 (2020)
11. Deng, P., Gong, X., Que, X.: A bandwidth-aware service migration method in leo satellite edge computing network. Comput. Commun. **200**, 104–112 (2023)
12. Han, H., Wang, H., Cao, S.: Space edge cloud enabling service migration for on-orbit service. In: 2020 12th International Conference on Communication Software and Networks (ICCSN). pp. 233–239 (2020)
13. Handley, M.: Delay is not an option: Low latency routing in space. In: Proceedings of the 17th ACM Workshop on Hot Topics in Networks (HotNets). pp. 85–91 (2018)
14. Ji, Z., Wu, S., Jiang, C.: Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks. IEEE J. Sel. Areas Commun. **41**(11), 3414–3429 (2023)
15. Kim, T., Sathyanarayana, S.D., Chen, S., Im, Y., Zhang, X., Ha, S., Joe-Wong, C.: Modems: Optimizing edge computing migrations for user mobility. IEEE J. Sel. Areas Commun. **41**(3), 675–689 (2022)
16. Ksentini, A., Taleb, T., Chen, M.: A markov decision process-based service migration procedure for follow me cloud. In: 2014 IEEE International Conference on Communications (ICC). pp. 1350–1354 (2014)
17. Lai, Z., Li, H., Li, J.: Starperf: Characterizing network performance for emerging mega-constellations. In: 2020 IEEE 28th International Conference on Network Protocols (ICNP). pp. 1–11 (2020)
18. Lai, Z., Li, H., Zhang, Q., Wu, Q., Wu, J.: Cooperatively constructing cost-effective content distribution networks upon emerging low earth orbit satellites and clouds. In: 2021 IEEE 29th International Conference on Network Protocols (ICNP). pp. 1–12 (2021)
19. Li, Z., Jiang, C., Lu, J.: Distributed service migration in satellite mobile edge computing. In: 2021 IEEE Global Communications Conference (GLOBECOM). pp. 1–6 (2021)
20. Liberti, L.: Undecidability and hardness in mixed-integer nonlinear programming. Rairo-Oper. Res. **53**(1), 81–109 (2019)
21. Libre Space Foundation: Satnogs: An open source ground station and network (2014), https://satnogs.org/, Last accessed 2014
22. Liu, C., Tang, F., Hu, Y., Li, K., Tang, Z., Li, K.: Distributed task migration optimization in mec by extending multi-agent deep reinforcement learning approach. IEEE Trans. Parallel Distrib. Syst. **32**(7), 1603–1614 (2020)
23. Lv, M., Peng, X., Xie, W., Guan, N.: Task allocation for real-time earth observation service with LEO satellites. In: 2022 IEEE Real-Time Systems Symposium (RTSS). pp. 14–26 (2022)

24. Milchtaich, I.: Congestion games with player-specific payoff functions. Games Econ. Behav. **13**(1), 111–124 (1996)
25. Mwasinga, L.J., Le, D.T., Raza, S.M., Challa, R., Kim, M., Choo, H.: Rasm: Resource-aware service migration in edge computing based on deep reinforcement learning. J. Parallel Distrib. Comput. **182**, 104745 (2023)
26. Ouyang, T., Zhou, Z., Chen, X.: Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. IEEE J. Sel. Areas Commun. **36**(10), 2333–2345 (2018)
27. Papadimitriou, C.H.: On the complexity of integer programming. J. ACM **28**(4), 765–768 (1981)
28. Pfandzelter, T., Bermbach, D.: Celestial: Virtual software system testbeds for the leo edge. In: Proceedings of the 23rd ACM/IFIP International Middleware Conference (Middleware). pp. 69–81 (2022)
29. Qin, Z., Yao, H., Mai, T., Wu, D., Zhang, N., Guo, S.: Multi-agent reinforcement learning aided computation offloading in aerial computing for the internet-of-things. IEEE Trans. Serv. Comput **16**(3), 1976–1986 (2022)
30. Song, Z., Hao, Y., Liu, Y., Sun, X.: Energy-efficient multiaccess edge computing for terrestrial-satellite internet of things. IEEE Internet Things J. **8**(18), 14202–14218 (2021)
31. Tao, B., Chabra, O., Janveja, I., Gupta, I., Vasisht, D.: Known knowns and unknowns: Near-realtime earth observation via query bifurcation in serval. In: 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 809–824 (2024)
32. Tuli, S., Casale, G., Jennings, N.R.: Pregan: Preemptive migration prediction network for proactive fault-tolerant edge computing. In: 2022 IEEE Conference on Computer Communications (INFOCOM). pp. 670–679 (2022)
33. Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., Leung, K.K.: Dynamic service migration in mobile edge computing based on markov decision process. IEEE/ACM Trans. Netw. **27**(3), 1272–1288 (2019)
34. Wen, Y., Geiping, J., Fowl, L., Goldblum, M., Goldstein, T.: Fishing for user data in large-batch federated learning via gradient magnification. In: 2022 ACM International Conference on Machine Learning (ICML). pp. 1–17 (2022)
35. Wu, H., Yang, X., Bu, Z.: Task offloading with service migration for satellite edge computing: A deep reinforcement learning approach. IEEE Access **12**, 25844–25856 (2024)
36. Zhang, H., Liu, R., Kaushik, A., Gao, X.: Satellite edge computing with collaborative computation offloading: An intelligent deep deterministic policy gradient approach. IEEE Internet Things J. **10**(10), 9092–9107 (2023)
37. Zhang, X., Liu, J., Zhang, R., Huang, Y., Tong, J., Xin, N., Liu, L., Xiong, Z.: Energy-efficient computation peer offloading in satellite edge computing networks. IEEE Trans. Mob. Comput. **223**(4), 3077–3091 (2023)
38. Zhu, X., Jiang, C.: Delay optimization for cooperative multi-tier computing in integrated satellite-terrestrial networks. IEEE J. Sel. Areas Commun. **41**(2), 366–380 (2022)