applied sciences

MDPI

Article

A Lyapunov-Optimized Dynamic Task Offloading Strategy for Satellite Edge Computing

Yifei Hu [1,2], Wenbin Gong [1,2,*] and Fangming Zhou [1,2]

[1] Innovation Academy for Microsatellites, Chinese Academy of Sciences, Shanghai 201210, China; huyf2021sh@163.com (Y.H.); zfmzjr@mail.ustc.edu.cn (F.Z.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: gongwb@microsate.com

Abstract: Satellite edge computing (SEC) has garnered significant attention for its potential to deliver services directly to users. However, the uneven distribution of receiving tasks among satellites in the constellation can lead to uneven utilization of computing resources. This paper proposes a task offloading strategy for SEC that aims to minimize the average delay and energy consumption of tasks by assigning them to appropriate satellite nodes. The approach uses Lyapunov optimization to convert the long-term optimization problem with task queue length constraints into an assignment problem within a single time slot and solve it based on the Hungarian algorithm. Experimental simulations have shown that the proposed algorithm performs better than other baseline algorithms.

Keywords: satellite edge computing; task offloading; Lyapunov optimization; Hungarian algorithm

Citation: Hu, Y.; Gong, W.; Zhou, F. A Lyapunov-Optimized Dynamic Task Offloading Strategy for Satellite Edge Computing. *Appl. Sci.* **2023**, *13*, 4281. https://doi.org/10.3390/app13074281

Academic Editors: Shu Fu and Yueyue Dai

Received: 8 March 2023
Revised: 27 March 2023
Accepted: 27 March 2023
Published: 28 March 2023

1. Introduction

Satellite networks are widely utilized for emergency communications, navigation and positioning, earthquake relief, and other applications due to their wide coverage, high data transmission rates, large capacity, and ability to operate independently of geographical limitations. Low Earth orbit (LEO) satellites have attracted the attention of scholars owing to their numerous advantages, including minimal transmission delays, support for global on-demand access, and terminal miniaturization, among others [1]. Currently, several countries have launched their own LEO satellite communication projects to supplement terrestrial services, such as Starlink, OneWeb, and O3b. Nevertheless, the majority of current LEO satellite networks employ the bent-pipe mode of data transmission, which involves forwarding large amounts of user data to the ground center for processing through the LEO network. This approach often leads to substantial network loads and long transmission delays due to complex inter-satellite links (ISLs) [2]. Additionally, the underutilization of LEO satellite communication and computing resources often leads to significant resource waste [2].

Multi-access edge computing (MEC) is an emerging computing paradigm that proposes the use of computing and communication resources near the user edge for task processing, instead of relying on cloud-centric resources [3]. This approach allows for data processing to be executed on edge servers, whereas the ground command center is responsible for storing processing results and conducting further data analysis. Drawing inspiration from MEC technology, edge servers are now being deployed on LEO satellite networks, creating what is known as an edge computing satellite (ECS) [4]. In this architecture, in-orbit resources of LEO satellites can be used directly by ground users for task processing, effectively mitigating time delays and network load issues. However, owing to the varying service regions of satellite networks, there are differences in task loads between satellites. The design of a rational computing offloading strategy for a satellite constellation can ensure a more balanced task load among satellites, and expedite task completion.

Therefore, a collaborative task processing approach among LEO satellites (MLSCTP) is worthy of investigation as it can effectively mitigate issues related to high transmission delay and network congestion caused by long-distance satellite-ground link transmissions. Moreover, it addresses the problem of high computational latency resulting from the limited computing power of a single satellite. However, the research on MLSCTP still faces several significant challenges [1,5]:

- The satellite topology is highly dynamic, and the ISLs are variable at different moments.
- Each satellite node is hard to manage during its operational phase as the task load changes, making its computing and communication resources change rapidly.
- No defined collaborative strategy between multiple LEO satellites to ensure efficient task scheduling while fully utilizing computing resources.

To address the issues mentioned above, this paper proposes a dynamic task offloading method based on the Lyapunov method and the Hungarian algorithm. In particular, for the computing offloading problem among satellites, the Lyapunov method is utilized to convert the long-horizon queue stability-constrained offloading problem into a single time-slot offloading optimization problem. Additionally, the problem is transformed into an assignment problem and solved using the Hungarian algorithm, which has low computational complexity and does not require any prior knowledge or multiple iterations to obtain optimal solutions. This approach provides a valuable reference for dynamic computing offloading among satellite constellations.

The contributions of this paper are summarized below:

- Consider a satellite edge computing scenario where the satellites in a constellation receive varying task loads. Each mission can be offloaded to any other satellite in the constellation for collaborative computing via an ISL. This problem has not been extensively studied by scholars.
- A strategy for offloading using the Lyapunov and Hungarian algorithms is proposed. The method is straightforward in principle, easy to implement, and has demonstrated superior performance compared with other benchmark algorithms.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. In Section 3, we describe the system model and problem formulation. Additionally, Section 4 presents the proposed algorithm. Simulation results and analysis are presented in Section 5. Section 6 summarizes the paper.

2. Related Works

Dynamic task scheduling strategies for collaborative computing among multiple satellites play an essential role in edge computing-enhanced LEO satellite networks. At present, some scholars have carried out the research. Liu et al. [6] proposed a novel system of task-oriented intelligent network architecture to deal with the problems arising in edge computing-enhanced sky-ground-water integrated networks. Xie et al. [7] analyzed the feasibility of MEC-enabled satellites-ground converged networks for processing tasks on satellites. An architecture called STECN is proposed, which discusses the solutions and technical challenges of using the resources of its heterogeneous edge computing clusters for task processing. In the above literature, although the authors propose and emphasize the importance and feasibility of task scheduling and resource allocation for MEC-enabled satellite clusters, none of them elaborated on specific task scheduling strategy schemes.

Bradley et al. [8] proposed a model for orbital edge computing and described the power and software optimization of the orbital edge. Cui et al. [9] proposed a joint offloading and resource allocation scheme for satellite-assisted vehicle networking. Song et al. [10] proposed a satellite MEC framework for terrestrial IoT to minimize the energy consumption of IoT mobile devices. They also proposed a computational offloading and resource allocation scheme. In the above works, each of the authors proposed suitable task scheduling and resource allocation schemes; however, none of them included the strategy of multi-satellite cooperative computing in the research model.

Zhiyuan Ren et al. [11] proposed to form a satellite fog network using a small satellite formation and designing a task scheduling strategy using the improved particle swarm algorithm, which can reduce the task processing latency while meeting energy consumption constraints. However, this paper only considers formation flying satellites, whose topology will not change and has little reference value for task scheduling of highly dynamic LEO satellite constellations.

Xiaobo Guo et al. [1] proposed a satellites collaborative computing method using business graph drive in LEO satellite networks. In the paper, Guo combined the computing resources of multiple satellites and ISLs resources to schedule the task to the satellites on the transmission path for processing. However, the authors only considered the scheduling of a single task and did not consider the case of multiple tasks arriving in sequence. Wang Cheng et al. [12] proposed to use the steady-state matrix of time-expanded graphs to represent the steady-state topology of dynamic LEO satellite networks. Additionally, the author designed a diffusion algorithm based on the matrix to achieve optimal task assignment for LEO satellite networks, which could efficiently complete the computational tasks while meeting delay constraints. However, this paper only considered the scheduling algorithm for a single-task scenario.

Based on this problem, Min et al. [13] suggested using satellites to assist ground users in computational offloading, where multiple satellites can directly connect to ground users and provide offloading. However, due to orbital coverage issues, most satellites in the actual satellite constellation cannot connect directly to ground users and must be further offloaded through the inter-satellite link. Dong et al. [14] considered using the full satellite constellation for offloading, considering the effects of multi-hop delay and energy consumption. Li et al. [15] proposed offloading the computing tasks of ground users to the over-the-top satellite. To address the unevenness of satellite network nodes, they proposed transferring computational tasks on under-resourced satellite nodes to more resource-rich nodes for collaborative processing through inter-satellite links, using a linear programming approach to solve related problems. However, these authors only considered offloading problems within a single time slot, and dynamic offloading strategies for satellite-ground networks under multiple time slots have not been investigated.

Furthermore, Tao et al. [5] proposed to jointly study the collaborative computation and resource allocation among LEO satellite networks. In the paper, the user tasks are selected for Local computing or offloaded to the satellite for collaborative computing. Specifically, the arrival of tasks was modeled as a Poisson process, and the task completion rate was maximized under the delay constraint. However, the author only considered the metric of task completion rate and did not consider the impact on task processing delay and energy consumption. Besides that, the DQN algorithm uses an experience replay mechanism. The interaction of the algorithm with the environment requires a large amount of memory resources, which is not friendly to satellites with limited resources [16,17].

To address the shortcomings of the scenarios and algorithms in the aforementioned studies, this paper proposes a multi-task, multi-time slots, multi-satellites offloading scenario, and suggests a succinct and effective algorithm as the solution.

3. System Model and Problem Description

3.1. System Model

A typical architecture for satellite edge computing is first outlined, as shown in Figure 1. In this configuration, the LEO satellites are equipped with edge computing servers, to which computational tasks from ground users (e.g., ships, aircraft, monitoring devices, and mobile communication devices) are uploaded. It enables the direct processing of received tasks on the satellite nodes, without the need for traditional task transfer back to the ground cloud center for computing. The satellites can communicate with the surrounding satellites through ISLs. Meanwhile, in each time slot, information on the use of satellite resources and the received tasks information is transmitted to the ground station through the backhaul link. The ground station updates the offload decision in real time based on this information

and sends it back to the satellite, taking into consideration the computing capabilities and task load status of each satellite in the constellation.

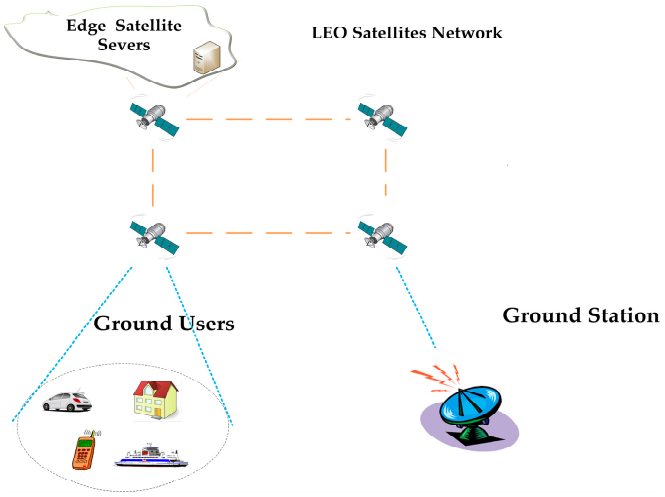

Figure 1. Satellites edge computing architecture.

The aforementioned scenario involves a constellation of N satellites orbiting the Earth in a periodic manner, with each period being divided into multiple time slots of equal duration τ. At every time slot, each satellite can receive new tasks, which are defined by $A_i(t) = \{D_i(t)\}$, where $D_i(t)(bit)$ represents the data amount of the task. Notably, the amount of user demand may vary depending on the geographic region covered by each satellite. For instance, in a given time slot, one satellite may be serving urban users, whereas another satellite remains idle over the ocean. In this scenario, the urban satellite could offload tasks to the idle ocean satellite for processing, effectively utilizing the constellation's computing resources to their full potential. To illustrate this, not all satellite $k(k \in N)$ could receive a task $A_k(t)$ at each time slot. The number of satellites that will receive a new task at each time slot, denoted by $M(M \leq N)$, and $D_i(t)$ follows a random distribution with a certain range $[\lambda_{min}, \lambda_{max}]$ and is independent and identically distributed. The tasks received by each satellite can be offloaded to other satellite nodes via ISLs for assistance in processing. A simple task offload schematic is employed to depict the scheduling process, as shown in Figure 2.

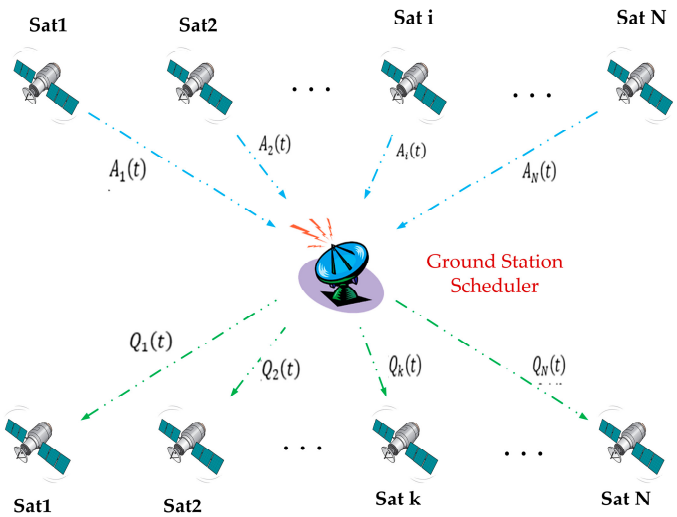

Figure 2. Task Offload Schematic.

3.2. Task Queue Model

The decision vector is defined as $o(t) = \{o_1(t), o_2(t), \cdots, o_M(t)\}$, in which $o_i(t) = \{o_i^1(t), o_i^2(t), \cdots, o_i^N(t)\}$. $o_i^k(t)$ denotes the offload selection for time slot t, which will be defined as

$$o_i^k(t) \in \{0,1\}, i \in \{1,2,\cdots,M\}, k \in \{1,2,\cdots,N\}. \tag{1}$$

The new task, $A_i(t)$, is assigned to the kth satellite when $o_i^k(t) = 1, k \in \{1,2,\cdots,N\}$. Furthermore, the task will remain in the ith satellite for processing without being offloaded when $o_i^k(t) = 1, i = k$. At time slot t, the quantity of data offloaded to the kth satellite is represented by the following equation

$$D_k(t) = \sum_{i=1}^{M} o_i^k(t) * D_i(t), i \in \{1,2,\cdots,M\}, k \in \{1,2,\cdots,N\}. \tag{2}$$

Specifically, to indicate that each task can only be offloaded to one satellite, we define

$$\sum_{k=1}^{N} o_i^k(t) = 1, i \in \{1,2,\cdots,M\}, k \in \{1,2,\cdots,N\}. \tag{3}$$

Furthermore, each satellite can only receive one task at a time slot. It is expressed as

$$\sum_{i=1}^{N} o_i^k(t) = 1, i \in \{1,2,\cdots,M\}, k \in \{1,2,\cdots,N\}. \tag{4}$$

The computational capacity of the satellite is denoted as $C_k(t)$ (bit/s). This denotes the processing capacity of the kth satellite, measured in terms of the amount of data it can handle per unit time.

Then, at time slot t, the amount of data in the kth satellite's task queue can be expressed as $Q_k(t+1)$ and is given by the following expression

$$Q_k(t+1) = \max[Q_k(t) - C_k(t) * \tau + D_k(t), 0], k \in \{1,2,\cdots,N\}. \tag{5}$$

Here, $Q_k(0)$ is initialized as zero for all $k \in \{1,2,\cdots,N\}$. Finally, we define the task queue stability requirement of the system [18] as follows:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[Q_k(t)] \leq 0, k \in \{1,2,\cdots,N\}. \tag{6}$$

3.3. Offloading Model

In this section, we will examine the impact of task offloading on both latency and energy consumption. Latency consists of four components: transmission latency, waiting latency, computation latency, and return latency. Typically, the return latency is negligible as the results are relatively small after computation. Energy consumption is comprised of two components: transmission energy and computational energy.

For the kth satellite, if there are tasks offloaded at time slot t, the transmission latency can be calculated as follows:

$$T_k^{trans}(t) = \sum_{i=1}^{M} o_i^k(t) * \frac{D_i(t)}{B} * Hop_i^k(t), \tag{7}$$

in which B represents the bandwidth of the ISL, and $Hop_i^k(t)$ denotes the shortest hop from the ith satellite to the kth satellite at time slot t. It should be noted that, unlike fixed terrestrial networks, satellites move at high speeds, causing their topology to vary at different time slots. Therefore, the shortest hop between two satellites may vary at different time slots. However, the changing topology can be represented by a connection matrix [1], which allows us to obtain the shortest hop between any two satellites at any given time slot.

To prevent tasks from being split due to changes in the topology of the satellite network, we assume that the transmission time for a single hop does not exceed the length of a time slot τ [18]. The propagation time for data is very short as the signal travels through space at the speed of light. Therefore, we ignore the propagation time to simplify the analysis.

The energy consumption associated with offloading to the kth satellite is defined as

$$E_k^{trans}(t) = T_k^{trans}(t) * P^{trans}, \tag{8}$$

in which, P^{trans} (J/s) is the transmission power of satellites.

$$T_k^{com}(t) = \frac{\sum_{i=1}^{M} o_i^k(t) * D_i(t)}{C_k(t)}, i \in \{1, 2, \cdots, M\}. \tag{9}$$

The computational latency is defined as

And the computational energy consumption can be expressed as

$$E_k^{com}(t) = T_k^{com}(t) * P_k^{com}, \tag{10}$$

in which, P_k^{com} (J/s) is the computing power of the kth satellite. Furthermore, the waiting time for a task in the queue is defined as

$$T_k^{wait}(t) = \frac{Q_k(t)}{C_k(t)}. \tag{11}$$

Therefore, for the kth satellite, the latency at time slot t can be expressed as

$$T_k(t) = T_k^{trans}(t) + T_k^{wait}(t) + T_k^{com}(t). \tag{12}$$

And the energy consumption is presented as

$$E_k(t) = E_k^{trans}(t) + E_k^{com}(t). \tag{13}$$

3.4. Probelm Formulation

The optimization objective of this paper is to reduce the average weighted sum of latency and energy consumption over a long period of task offloading, while ensuring that the system queue is stable.

We define the cost function of offloading tasks to the kth satellite in time slot t as $Cost = T_k(t) + \beta * E_k(t)$, where β represents the weight assigned to energy consumption, reflecting its relative importance in the optimization problem. The optimization objective function can be defined as follows

$$\begin{gathered} P1: \min \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=1}^{N} Cost. \\ s.t. (1)(3)(4)(6) \end{gathered} \tag{14}$$

4. Task Offload Optimization Algorithm

4.1. Probelm Conversion

Long-term constraints and optimization objectives pose significant challenges to solving the optimization problem $P1$, primarily because obtaining task information for future time slots is challenging. $P1$ can be characterized as a Markov decision process (MDP), which conventional MDP methods such as reinforcement learning can potentially address. However, the general MDP algorithm is highly complex, rendering it unsuitable for real-time communication requirements. Furthermore, the MDP algorithm requires storing the optimal action for each state, which can significantly occupy satellite storage space.

The Lyapunov optimization method is a promising online optimization approach that does not require a priori knowledge of statistical and future data [19]. One such approach

is the Lyapunov-optimized dynamic task offloading strategy, which has been proposed by Dai et al. in their paper [20]. This approach optimizes the task offloading decision based on the Lyapunov optimization technique, which can effectively balance the trade-off between system stability and task completion time. Other recent studies in the field of mobile edge computing include the joint task offloading and resource allocation approach proposed by Jiang et al. [21]. The Lyapunov method can convert the optimization of long-term objectives into solving the minimal value for each time slot, based on theoretical principles.

First, we define the virtual task queues vector in the satellite constellation as

$$\theta(t) \triangleq [Q_1(t), Q_2(t), \cdots, Q_N(t)]. \tag{15}$$

The Lyapunov function can be expressed as

$$L(\theta(t)) \triangleq \frac{1}{2}\sum_{k=1}^{N} Q_k(t)^2. \tag{16}$$

The Lyapunov drift is defined as

$$\Delta(\theta(t)) = L(\theta(t+1)) - L(\theta(t)), \tag{17}$$

which shows the variation in the system queue between two time slots. Analyze the $\Delta(\theta(t))$ as

$$\begin{aligned}
\Delta(\theta(t)) &= L(\theta(t+1)) - L(\theta(t)) \\
&= \tfrac{1}{2}\textstyle\sum_{k=1}^{N} Q_k(t+1)^2 - \tfrac{1}{2}\sum_{k=1}^{N} Q_k(t)^2 \\
&\le \tfrac{1}{2}\Big(\textstyle\sum_{k=1}^{N}(Q_k(t) - C_k(t)*\tau + D_k(t))^2 - \sum_{k=1}^{N} Q_k(t)^2\Big) \\
&= \tfrac{1}{2}\Big(\textstyle\sum_{k=1}^{N} Q_k(t)^2 + 2*\sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau) + \sum_{k=1}^{N}((D_k(t) - C_k(t)*\tau)^2 - \sum_{k=1}^{N} Q_k(t)^2\Big) \\
&= \textstyle\sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau) + \tfrac{1}{2}\sum_{k=1}^{N}((D_k(t) - C_k(t)*\tau)^2 \\
&\le \textstyle\sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau) + \tfrac{1}{2}\sum_{k=1}^{N} D_k(t)^2 + (C_k(t)*\tau)^2 \\
&\le \delta + \textstyle\sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau)
\end{aligned} \tag{18}$$

As $D_k(t)$ conforms to a random distribution with upper bound λ_{max} and $C_k(t)$ is the satellite CPU's computing capacity, $\frac{1}{2}\sum_{k=1}^{N} D_k(t)^2 + (C_k(t)*\tau)^2$ must have an exact max value δ at time slot t. Furthermore, we add up the $\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t))$ of each time slot to obtain

$$\Delta(\theta(t)) + \mathrm{V}\textstyle\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t)) \le \delta + \sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau) + V\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t)), \tag{19}$$

in which V represents the weight to adjust the importance of the two metrics. The right half of the equation is called the drift plus penalty function. Summarize (19), and we can obtain

$$\begin{aligned}
&\lim_{T\to\infty}\tfrac{1}{T}(\textstyle\sum_{t=0}^{T-1}\Delta(\theta(t)) + \mathrm{V}\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t))) \\
&= \lim_{T\to\infty}\Big(\tfrac{L(\theta(t))}{T} + \tfrac{\mathrm{V}}{T}\textstyle\sum_{t=0}^{T-1}\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t))\Big) \\
&= \lim_{T\to\infty}\tfrac{L(\theta(t))}{T} + \lim_{T\to\infty}\tfrac{\mathrm{V}}{T}\textstyle\sum_{t=0}^{T-1}\sum_{k=1}^{N}\mathrm{Cost}.
\end{aligned} \tag{20}$$

Hence, the long-term optimization problem $P1$ is transformed into finding the offloading strategy for each time slot that minimizes the drift plus penalty function within that time slot. The objective function $P1$ is converted to

$$\begin{aligned}
P2: \min_{O(t), t\epsilon T} &\mathbb{E}\Big[\delta + \textstyle\sum_{k=1}^{N} Q_k(t)*(D_k(t) - C_k(t)*\tau) + V\sum_{k=1}^{N}(T_k(t) + \beta * E_k(t))\Big]. \\
&s.t.(1)(3)(4)\#
\end{aligned} \tag{21}$$

Once $P2$ is determined for all time slots, the solution for $P1$ can be obtained. The optimal offloading strategy for each time slot can be computed based solely on the offloading decisions made during that time slot, without the need to consider offloading decisions made in other time slots. This greatly simplifies the optimization problem $P1$. The drift-plus-penalty algorithm is able to guarantee that the long-term constraint of a stable queue holds constantly, and similar proofs have been provided and will not be repeated in this paper [22,23]. Since the task queue of the LEO satellite satisfies boundedness assumptions, the proposed algorithm can achieve solutions that are arbitrarily close to optimal by setting V to a sufficiently large value while maintaining mean rate stable with $O(V)$ for the virtual task queue [19].

4.2. Offloading Algorithm Based on The Hungarian Method

It is evident that for the transformed optimization objective $P2$, the offloading decisions for each time slot are sufficient to determine $P2$ for that slot. Assuming that in cases where only a subset of the satellites in the constellation receives a new task at this time slot $(M < N)$, a virtual task $A_l(t) = \{0\}$ is assigned to the remaining satellites. Specifically, in a specific time slot, the offloading decision $o(t)$ is defined as an $N \times N$ matrix, which is shown as

$$o = \begin{bmatrix} o_1^1 & \cdots & o_1^N \\ \vdots & \ddots & \vdots \\ o_N^1 & \cdots & o_N^N \end{bmatrix}. \tag{22}$$

It is supposed to find a specific offloading strategy $\pi \in S$ to minimize $P2$, in which S is the set of all feasible solutions. Combined with constraints (1) (3) (4), the optimization problem $P2$ can be defined as a classic assignment problem [24]. The Hungarian method is a widely adopted approach for solving this type of problem [24]. Referring to the offloading matrix o, we define a cost matrix of the same size as

$$L = \begin{bmatrix} L_1^1 & \cdots & L_1^N \\ \vdots & \ddots & \vdots \\ L_N^1 & \cdots & L_N^N \end{bmatrix}. \tag{23}$$

$L_i^j \in L, i, j \in \{1, 2, \cdots, N\}$ represents the value of objective function if task A_i is offloaded to the jth satellite, which is calculated by $L_i^j = Q_j(t) * \left(D_j(t) - C_j(t) * \tau\right) + V * (T_j(t) + \beta * E_j(t))$. The Hungarian method can solve the best matching that minimizes $P2$, i.e., the optimal solution of the problem by certain transformation. The detailed steps are shown below

1. For each row in the matrix L, identify its minimum entry and subtract it from all entries in that row;
2. For all entries L_i^j equal to 0, mark them as starred zeros, provided that there are no existing starred zeros in the same row or column;
3. Cover each column that contains a starred zero. If all columns are covered, proceed to Step 7;
4. Repeat the following procedure until all zeros in the matrix are covered: Find an uncovered zero and prime it. If there are no starred zeros in the same row as the primed zero, proceed to Step 5. Otherwise, cover this row and uncover the column containing the starred zero;
5. Continue to construct a series of alternating primed and starred zeros until no additional uncovered zeros can be found. Then, proceed to Step 6;
6. Locate the smallest entry among all uncovered entries in the matrix. Subtract this entry from all uncovered entries, and add it to all entries that are covered twice. Return to Step 4;

7. The algorithm terminates when all entries in the matrix are covered. The set of assignments corresponding to the starred zeros constitutes the optimal solution to the assignment problem.

According to the obtained optimal solution, then the optimization objective $P2$ can be solved. The optimal solution for $P2$ across all time slots is computed, followed by obtaining the optimal solution for $P1$. The procedure of the dynamic offloading algorithm that employs Lyapunov optimization and the Hungarian algorithm (DOALH) is shown in Algorithm 1.

Algorithm 1 The DOALH algorithm

Input: System parameter $Hop, A(t), C(t), B, P^{trans}, P^{com}, T$
Algorithm

1. Obtain the optimization objective function based on *P1*;
2. Convert the objective function to *P2* utilizing Lyapunov optimization;
3. For $t < T$:

   Using the Hungarian algorithm to determine the optimal offloading strategy in the specific time slot;
4. Counting the optimal solutions for all T time slots;

Output: The minimum mean value of *Cost* in T time slots

The complexity of the DOALH algorithm is $TO(N^2)$.

5. Simulation and Analysis

5.1. Parameters Setting

The system has been constructed using the Iridium Next constellation [25]. To establish the specific simulation parameters of the DOALH algorithm, we referred to existing studies [1,11] and outlined them in Table 1. To validate the efficacy of the proposed algorithm, we present the following alternative approaches for comparison:

1. Local Computing (Local): All tasks are processed solely within the receiving satellite, without any computational offloading.
2. Greedy Offloading (Greedy): In each time slot, the task with the highest number of pending tasks is offloaded to the satellite with the lowest task queue for processing.
3. DQN Offloading (DQN): In a similar study, the authors modeled the computational offloading process as an MDP and used Deep Q-Networks (DQN) to determine the optimal offloading strategy. The reward function is set as $P2$ [5].

Table 1. System parameters.

Satellites computational capacity C_k (Mb/s)	[10, 12]
Number of satellites in the constellation N	6×11
The bound of random distribution $[\lambda_{min}, \lambda_{max}]$(Mb)	[10, 15]
Satellites transmission power P^{trans} (J/s)	400
Satellites computational power P^{com} (J/s)	[100, 200]
Length of one time slot τ (s)	10
Simulation time slots T	360

5.2. Simulation Analysis

As we have previously analyzed, the value of V has a crucial impact on both the task queue length and the performance of the algorithm. As shown in Figure 3, $M = 50, \beta = 0.1, B = 500(\mathrm{Mb/s})$ (a) Average task queue length per satellite vs. V of DOALH algorithm; (b) Average task queue length per satellite vs. different algorithms. Figure 3a, when the value of V is small, the queue can be kept stable directly. However, when the value of V is large, the queue grows gradually and eventually converges to an approximately stable state. The stable length of the queue convergence is positively related to the value of V. As Figure 3b reveals, among the different algorithms, the DOALH algorithm guarantees the optimal solution for $P2$ at each time slot, resulting in the shortest average queue length. The DQN algorithm adopts $P2$ as the reward function, which also guarantees the convergence of the queue to a lower value. The Greedy algorithm also results in a shorter queue length compared with the Local algorithm due to its greedy strategy. The Local algorithm generates a large queue buildup because it only considers Local processing and does not make rational use of other available satellite resources.

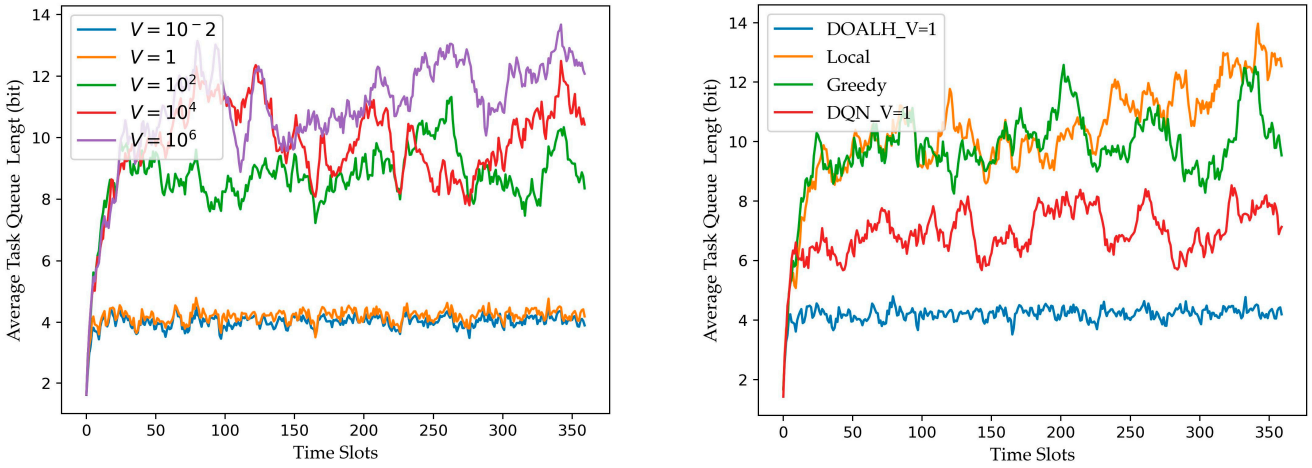

Figure 3. $M = 50, \beta = 0.1, B = 500$ $(\mathrm{Mb/s})$ (**a**) Average task queue length per satellite vs. V of DOALH algorithm; (**b**) Average task queue length per satellite vs. different algorithms.

According to Figure 4, the Greedy algorithm attains the highest average cost, primarily because it solely focuses on achieving an improved balance in the queue without considering the costs associated with delay and energy consumption resulting from ISL transmission. The Local and Greedy algorithms are simpler and are not impacted by the value of V, enabling them to maintain a consistent trend. However, both the DQN algorithm and the DOALH algorithm exhibit a substantial decreasing trend as the value of V increases. Eventually, they converge to a stable value. This phenomenon can be attributed to the fact that as the value of V increases, the algorithm places more emphasis on minimizing the *Cost* incurred by the task, which allows it to achieve a more optimal solution. The simulation results align with the preceding analysis of the algorithm. Thus, V should be chosen for a good trade-off between the overall *Cost* and the required task queue length.

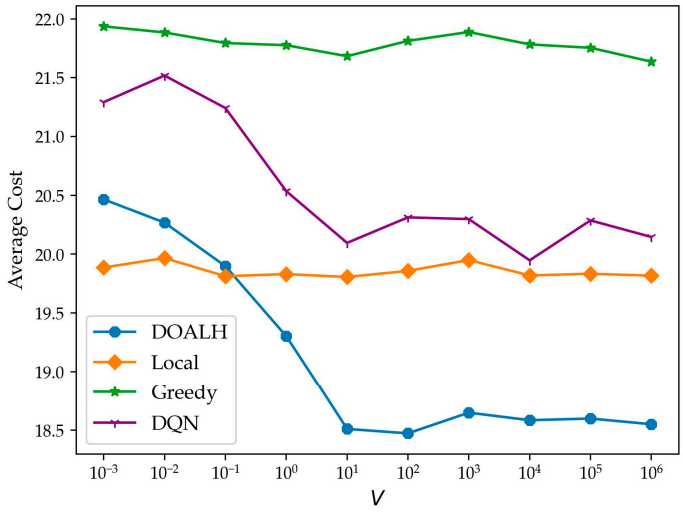

Figure 4. $M = 50, \beta = 0.1, B = 500$ (Mb/s) Average *Cost* vs. V.

Furthermore, we need to investigate the impact of the energy consumption weight β on performance. A larger β results in a higher weight of energy consumption. As demonstrated in Figure 5, the *Cost* clearly increases as β increases. The algorithms exhibit linear growth, albeit with varying slopes, which correspond to the energy consumption per time slot of offloading strategies. The Greedy algorithm aims to balance the queue by offloading more tasks to other satellites, resulting in the highest energy consumption for offloading strategies. On the other hand, the DOLAH algorithm is capable of maintaining the lowest *Cost*.

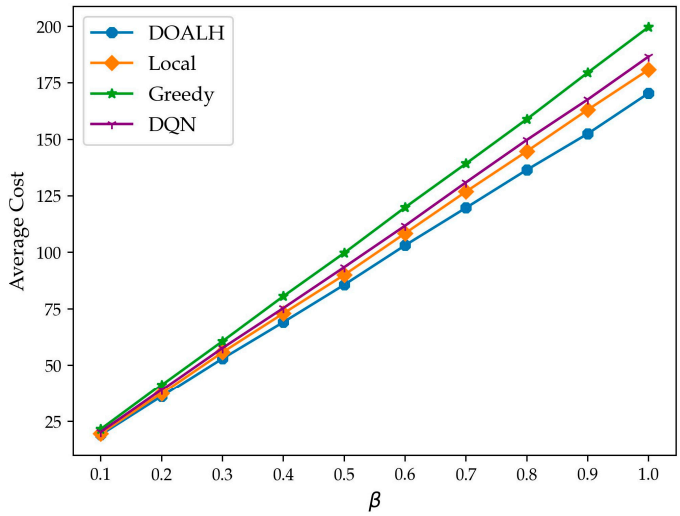

Figure 5. $V = 10, M = 50, B = 500$ (Mb/s) Average *Cost* vs. β.

Next, we investigate the correlation between *Cost* and the number of satellites that receive tasks during each time slot M. The value of M reflects the operational workload of the satellite constellation, where a larger value indicates that more tasks are assigned to satellites during each time slot, resulting in fewer idle satellites and computing resources in orbit. As depicted in Figure 6, the *Cost* increases as M increases. This is due to the longer queue length of tasks with more tasks received in the constellation. With limited computational resources available to assist in processing, tasks require more waiting time. The DOALH algorithm always achieves optimal cost by ensuring that all satellite resources are fully utilized, regardless of the value of M.

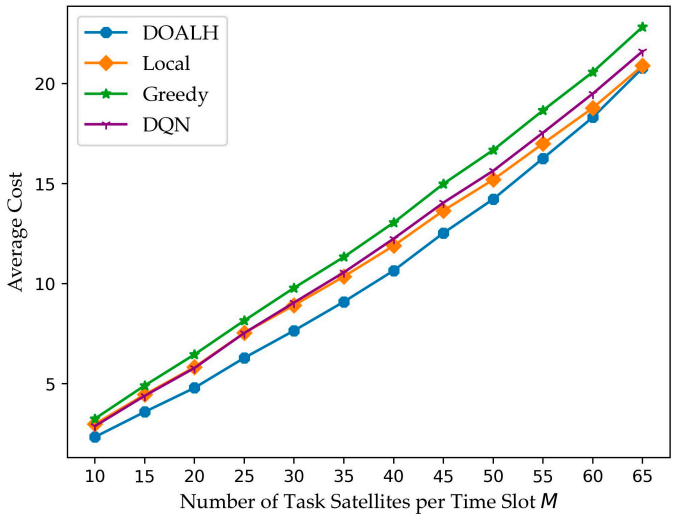

Figure 6. $V = 10, \beta = 0.1, B = 500$ (Mb/s) Average *Cost* vs. M.

The bandwidth of the ISL has a significant impact on the computational offload. In order to simulate both conventional band communication rates and possible future laser communication rates, the bandwidth was set between 100 Mb/s and 1000 Mb/s. This allows for an accurate assessment of the transmission capabilities of the link in relation to the offloading *Cost*. The Local algorithm has remained largely unchanged, as it excludes the possibility of offloading tasks to other satellites for computational assistance. Changes in the bandwidth of ISLs do not have an impact on this algorithm. However, the other three algorithms have undergone various modifications. Specifically, the Greedy algorithm changes most significantly because it prioritizes offloading tasks to the satellite with the smallest queue for processing. Additionally, the DOALH algorithm still performs best, as depicted in Figure 7.

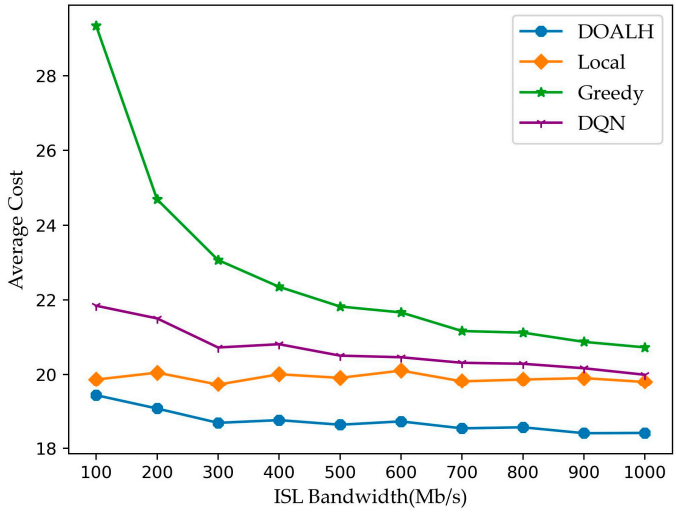

Figure 7. $M = 50, \beta = 0.1, V = 10$ Average *Cost* vs. ISL bandwidth.

It is important to note that while the DQN algorithm showed some level of performance, it was not outstanding in comparison with the DOALH algorithm. Throughout the simulation, we made several parameter modifications to observe its performance, but it still fell short of the DOALH algorithm. Reinforcement learning algorithms typically require a large amount of historical data for training and a long training time to converge to a better solution. Additionally, their performance heavily depends on the algorithm parameter settings, making it challenging to implement in practical engineering applications.

The DOALH algorithm, on the other hand, can quickly obtain optimal solutions without requiring any of the aforementioned information. As a result, it holds significant reference value for practical engineering applications.

6. Conclusions

We propose an offloading strategy for the dynamic task allocation problem in satellite constellations, utilizing a combination of Lyapunov and Hungarian algorithms. Firstly, we construct a dynamic model for task offloading among satellites and formulate an optimization problem that minimizes the average delay and energy consumption with a long-term constraint on the task queue length. To solve this problem, we employ Lyapunov optimization to convert the long-term optimization objective into an assignment problem within a single time slot. The Hungarian algorithm is then utilized to solve the resulting assignment problem. We validate the effectiveness of our algorithm through experimental simulations, investigating the impact of various parameters. Our results demonstrate the efficacy of our proposed approach.

7. Future Work

This paper provides some insights into the future deployment of satellite-based Internet services for terrestrial users. However, the offloading model proposed in this study remains relatively simplistic. Future research should aim to address the complexities introduced by the heterogeneity of satellite computing resources and communication capabilities, as well as the more uneven frequency of task generation. At the same time, this paper focuses solely on simulating the architecture of the Iridium NEXT constellation, and thus the impact of traditional satellite constellations on the proposed offloading strategy is explored. However, the potential impact of emerging giant constellations, such as Starlink, has yet to be studied. These factors are likely to further complicate the task offloading problem, and warrant further study.

Author Contributions: Conceptualization, Y.H.; Formal analysis, Y.H.; Investigation, Y.H.; Methodology, Y.H.; Project administration, W.G.; Software, Y.H.; Supervision, W.G.; Validation, Y.H.; Visualization, F.Z.; Writing—original draft, Y.H.; Writing—review and editing, F.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guo, X.; Ren, Z.; Cheng, W.; Ji, Z. Inter-satellite cooperative computing scheme driven by business graph in leo satellite network. *Space-Integr.-Ground Inf. Netw.* **2021**, *2*, 35–44.
2. Han, J.; Wang, H.; Wu, S.; Wei, J.; Yan, L. Task scheduling of high dynamic edge cluster in satellite edge computing. In Proceedings of the 2020 IEEE World Congress on Services (SERVICES), Online, 18–24 October 2020; IEEE: New York, NY, USA, 2020; pp. 287–293.
3. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1657–1681. [CrossRef]
4. Tang, Q.; Fei, Z.; Li, B.; Han, Z. Computation offloading in leo satellite networks with hybrid cloud and edge computing. *IEEE Internet Things J.* **2021**, *8*, 9164–9176. [CrossRef]
5. Leng, T.; Li, X.; Hu, D.; Cui, G.; Wang, W. Collaborative computing and resource allocation for leo satellite-assisted internet of things. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 4212548.
6. Liu, J.; Du, X.; Cui, J.; Pan, M.; Wei, D. Task-oriented intelligent networking architecture for the space–air–ground–aqua integrated network. *IEEE Internet Things J.* **2020**, *7*, 5345–5358. [CrossRef]

7. Xie, R.; Tang, Q.; Wang, Q.; Liu, X.; Yu, F.R.; Huang, T. Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues. *IEEE Netw.* **2020**, *34*, 224–231. [CrossRef]
8. Denby, B.; Lucia, B. Orbital edge computing: Machine inference in space. *IEEE Comput. Archit. Lett.* **2019**, *18*, 59–62. [CrossRef]
9. Cui, G.; Long, Y.; Xu, L.; Wang, W. Joint offloading and resource allocation for satellite assisted vehicle-to-vehicle communication. *IEEE Syst. J.* **2020**, *15*, 3958–3969. [CrossRef]
10. Song, Z.; Hao, Y.; Liu, Y.; Sun, X. Energy-efficient multiaccess edge computing for terrestrial-satellite internet of things. *IEEE Internet Things J.* **2021**, *8*, 14202–14218. [CrossRef]
11. Ren, Z.-Y.; Hou, X.-W.; Guo, K.; Zhang, H.-L.; Chen, C. Distributed satellite cloud-fog network and strategy of latency and power consumption. *J. ZheJiang Univ. (Eng. Sci.)* **2018**, *52*, 1474–1481.
12. Wang, C.; Ren, Z.; Cheng, W.; Zheng, S.; Zhang, H. Time-expanded graph-based dispersed computing policy for leo space satellite computing. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; IEEE: New York, NY, USA, 2021; pp. 1–6.
13. Jia, M.; Zhang, L.; Wu, J.; Guo, Q.; Gu, X. Joint computing and communication resource allocation for edge computing towards Huge LEO networks. *China Commun.* **2022**, *19*, 73–84. [CrossRef]
14. Dong, F.; Huang, T.; Zhang, Y.; Sun, C.; Li, C. A Computation Offloading Strategy in LEO Constellation Edge Cloud Network. *Electronics* **2022**, *11*, 2024. [CrossRef]
15. Chengcheng, L.; Yasheng, Z.; Chenhua, S. Computation Offloading for Satellite-based Edge Computing in LEO Satellite Network. *Radio Commun. Technol.* **2022**, *48*, 401–407.
16. Zou, J.; Hao, T.; Yu, C.; Jin, H. A3c-do: A regional resource scheduling framework based on deep reinforcement learning in edge scenario. *IEEE Trans. Comput.* **2020**, *70*, 228–239. [CrossRef]
17. Qi, F.; Zhuo, L.; Xin, C. Deep reinforcement learning based task scheduling in edge computing networks. In Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 9–11 August 2020; IEEE: New York, NY, USA, 2020; pp. 835–840.
18. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
19. Neely, M.J. Stochastic network optimization with application to communication and queueing systems. *Synth. Lect. Commun. Netw.* **2010**, *3*, 1–211.
20. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.; Min, G.; Dustdar, S.; Liu, J. Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems. *IEEE Trans. Ind. Inform.* **2022**, *19*, 662–672. [CrossRef]
21. Jiang, H.; Dai, X.; Xiao, Z.; Iyengar, A.K. Joint task offloading and resource allocation for energy-constrained mobile edge computing. *IEEE Trans. Mob. Comput.* **2022**. [CrossRef]
22. Li, N.; Hu, Y.; Chen, Y.; Zeng, B. Lyapunov optimized resource management for multiuser mobile video streaming. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 1795–1805. [CrossRef]
23. Huang, C.; Wang, H.; Zeng, L.; Li, T. Resource scheduling and energy consumption optimization based on Lyapunov optimization in fog computing. *Sensors* **2022**, *22*, 3527. [CrossRef]
24. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
25. Iridium. Available online: https://www.iridium.com/blog/iridium-next-review/ (accessed on 4 March 2023).