applied sciences

MDPI

Article

Offloading of Atomic Tasks in Satellite Networks: A Fast Adaptive Resource Collaboration Method

Yanbing Li [1,†], Wei Zhao [2,†] and Huilong Fan [2,*]

[1] School of Cyber Science and Engineering, College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China; liyb@xju.edu.cn
[2] School of Computer Science and Engineering, Central South University, Changsha 410075, China; zwzhaowei@csu.edu.cn
* Correspondence: hlfanpro@csu.edu.cn
† These authors contributed equally to this work.

Abstract: With the explosive growth of multimedia services and the continuous emergence of new space tasks, the spatial task scheduling timeliness problem is of great concern. The high computational cost of existing task scheduling methods is not suitable for the time-varying scenarios of space-based networks. This paper proposes a scheduling optimization method containing an atomic task offloading model based on maximum flow theory and a dynamic caching model. Firstly, the model calculates the task offloading upper limit in the satellite network based on the maximum flow theory to achieve the maximum volume of offloaded tasks to improve the resource utilization of idle satellites. Then, we design onboard task offloading and buffer optimization algorithms to reduce the request load of single-satellite atomic tasks. The method improves the overall computational performance and timeliness of the satellite network and reduces the waiting time of atomic tasks competing for resources. Finally, we analyze the time complexity of the proposed method and construct a simulation experiment scenario. The performance comparison results with various baseline models show that the proposed method has certain time complexity and task execution timeliness advantages.

Keywords: task offload; task cache; maximum flow; satellite network; atomic task

Citation: Li, Y.; Zhao, W.; Fan, H. Offloading of Atomic Tasks in Satellite Networks: A Fast Adaptive Resource Collaboration Method. *Appl. Sci.* **2022**, *12*, 3319. https://doi.org/10.3390/app12073319

Academic Editor: Arcangelo Castiglione

Received: 8 February 2022
Accepted: 22 March 2022
Published: 24 March 2022

1. Introduction

Space-Terrestrial integration [1] is one of the basic national defense strategies for all countries in the future. As an important national information infrastructure, the integration of heaven and earth can realize the real-time interconnection of multifunctional satellites or satellite groups and closely integrate the resources of the sky, air, earth, ships, and islands. The future development of the Space-Terrestrial information network will need to cross the sky and earth platforms to realize the integrated management of space and earth resources. The existing Space-Terrestrial network management system is mainly completed on ground control center [2]. The ground control center mainly focuses on situational awareness, navigation, positioning, and satellite communication. The spacecraft are networked through various satellite networks with different functions to achieve global coverage of information interaction and mission processing capabilities [3]. As shown in Figure 1, the network control center is the pivot for directing the work of the satellites. The control center uses multiple computers to command and monitor the operation of the satellite, which responds to various commands from the satellite, arranging satellite work procedures, controlling satellite operating attitude, directing sensor work and information transmission, controlling onboard instruments and ground receiving stations to work cooperatively, etc. The various tasks on the space-based network are first broken down and calculated at the control center [4,5]. The instructions are sent to the satellite through the ground base station when there exists a visible time window between the satellite and

the ground base station, then the satellite performs the relevant operations after receiving the instructions to complete the tasks. As shown in Figure 1, blocks with different colors represent the different tasks in the request queue received by satellite. With the explosive growth of multimedia services and the emergence of new space missions, many requests can in turn be sent to the satellite when there is a visible time window between the satellite and the ground base station [5]. Therefore, it is difficult to guarantee the efficiency and real-time performance of existing space-based management methods.

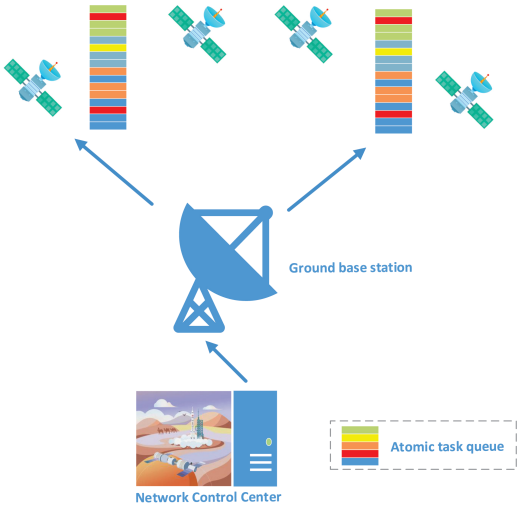


Figure 1. Overview of the task offloading method.

With the gradual increase in reliance on space-based information network services across industries and escalating and increasing commercial and military demands, "rapid response" and "efficient service" are fundamental aspects that require attention [5]. In spatial information networks, it is challenging to achieve fast resource allocation and high resource utilization with scarce dynamic satellite resources without reliable technical support for resource management [6]. With the explosive growth of multimedia services and the emergence of new space missions, the existing spatial information system management methods can no longer satisfy the high efficiency and real-time multitasking requirements for improving the response speed of satellite networks and increasing inter-satellite resource utilization [7]. With the rapid development of edge computing and the gradual maturity of related methods, most researchers are increasingly applying the ideas of edge computing to satellite networks.

Task offloading is a crucial technology for edge computing. Task offloading transfers terminal tasks to the edge cloud environment and provides resources for resource-constrained devices. Task offloading can be summarized in two phases. First, tasks with high resource requirements are reasonably assigned to proxy satellites with sufficient resources for processing [8]. Then the computation results are retrieved from the proxy satellites. Different factors can affect the whole process, such as radio channel communication, backhaul connection quality, device performance, cloud server availability, etc. [9]. Therefore, the key to resolving the task offloading problem is to specify appropriate offloading decisions that measure the performance of the approach through time delay and energy consumption. The merit metrics of the offloading strategy can be determined based on the time consumption metrics of the scheme, which we call the minimizing cost offloading decision.

There are still three challenges in solving the optimization problem of atomic task offloading in spatial information networks:

1. Although the time cost consumption of the traditional method is low, its performance cannot satisfy the high time-effective requirements for offloading atomic tasks in Spatio-temporal dynamic characteristics network environments.
2. The traditional satellite task offloading method does not consider the concept of inter-satellite collaborative computing, the inter-satellite resource coordination capability is low.
3. There is no optimization method for on-board task cache.

This paper proposes an optimization model to address the Offloading and Caching of satellite atomic tasks based on the Maximum Flow method (OCMF) to address the above problems. The method mainly includes three sub-models: the maximum task flow transmission model, the task offload optimization model, and the task cache optimization model.

To solve the problem of calculating the maximum number of unloaded atomic tasks for the joint transmission of multiple low-medium orbit satellites, the maximum task flow buffer model limits the atomic tasks scale upper limit that can be unloaded per satellite, ensuring the maximum number of unloaded tasks in the local satellite network. Task offloading optimization mainly includes two aspects: the optimization of task offloading from low-medium orbit satellites to high-orbit satellites, and the other is the optimization of task offloading from high-orbit satellites to low-orbit and medium-orbit satellites with idle resources. The model reduces the long-term waiting of atomic tasks so that when the satellite network benefits the most, the waiting time is consumed in the least. The task cache optimization model avoids the repeated calculation of many repeated atomic tasks and reduces the atomic task calculation time. Therefore, this paper mainly considers the maximum unloading of satellite atomic tasks and the full utilization of satellite resources, the reduction of the atomic tasks waiting time, and the reduction of atomic task calculation time, which provide new ideas for improving the efficiency and real-time performance of atomic tasks in satellite networks.

Our contributions can be briefly summarized as follows:

1. We propose the OCMF model, which divides the original problem into three parts and designs the maximum flow transmission, task offload, and task cache optimization models.
2. We introduce the maximum flow calculation method to increase the number of unloaded atomic calculation tasks in the satellite network.
3. We propose a method for high-orbit satellites to offload tasks to idle medium-earth satellites, which reduces the computational pressure of high-orbit satellites and makes full use of the remaining idle satellite resources.
4. We have analyzed and compared the performance of various algorithms. The OCMF method has certain advantages in terms of time complexity and task execution timeliness.

2. Related Work

With the rapid development of edge computing technology, task offloading has become a widely studied problem. Research on task offloading decisions can be divided into three categories. Task prediction decision methods based on deep learning, task decision methods based on reinforcement learning and task decision methods based on deep reinforcement learning. However, the task decision based on the deep learning method can predict the task state change in period and update the decision of task unloading plan in time. However, these method requires a large amount of computing power and more training data and the computation is time-consuming. The method obviously cannot satisfy the requirements for some scenarios with real-time requirements. Researchers have proposed a reinforcement learning method for scenarios with high requirements on timeliness. The reinforcement learning method can solve the problem of long computation time, but it requires a long training time and a relatively long excitation function. It is challenging to initiate the values of the parameters and local optima can quickly appear. In addition, reinforcement learning methods do not learn new data well as the scene changes rapidly. To address these problems, researchers have proposed a deep reinforcement learning model by combining the advantages of the above methods. The deep reinforcement learning model solves the

problems of excessive time consumption and low accuracy to a certain extent, but the algorithm still has some drawbacks in time consumption for those sensitive to timeliness and real-time scenes.

To explore the effective offloading decision and resource allocation methods of mobile edge computing, Wang et al. [10] developed a collaborative computing system and designed a new computing offloading strategy based on Q-learning to achieve the optimal resource allocation and offloading scheme. Gao et al. [11] Use Lyapunov optimization technology to transform time slots and propose an online algorithm in each time slot to determine the task offloading strategy. Liu et al. [12] proposed an online energy-saving task allocation and computing offloading strategy, taking into account service delay constraints, etc., adaptively determining task allocation and computing resource allocation. Sun et al. [13] Proposed an efficient Lyapunov online algorithm to perform joint task offloading and dynamic data caching strategies for computing tasks, reducing-edge computing latency, and energy consumption. Zhang et al. [14] proposed deep reinforcement learning to solve the offloading problem of cluster multi-service nodes and multi-dependency of mobile tasks in a large-scale heterogeneous mobile edge computing environment. Lu et al. [15] proposed a device-level and edge-level task offloading joint optimization scheme based on deep reinforcement learning, which achieved a good balance between task delay and task energy consumption. Yan et al. [16] taking the choice of the target server and the amount of data unloading as learning goals, a deep reinforcement learning strategy based on multi-agents to solve the problem of multi-agent environment instability.

Due to the strong coupling between combined offloading decision and task execution, the efficiency is very low when the numerical scale is large. Lu et al. [17] proposed a deep reinforcement learning framework based on the deep reinforcement learning structure. Yan et al. [18] proposed a random mixed integer nonlinear programming problem to optimize task offloading decisions, flexible computing resource scheduling, and wireless resource allocation. From the perspective of offloading decisions to maximize revenue. Zhang et al. [19] proposed a mobile carrier-based IoT edge cloud computing offloading scheme. Use the sensing device to generate tasks and transfer the tasks to the device, and then the device can decide to calculate the task locally on the MEC server or in the cloud center. In order to solve the problem of the rapid increase of vehicles and the in-vehicle terminal can not achieve efficient calculation, Gavras et al. [20] proposes a task offloading strategy for an edge-computing architecture based on reinforcement learning to compute for vehicular networking. Wang et al. [21] proposed task offloading algorithm effectively optimizes task latency and computational resource consumption in a multi-user and multi-server airborne edge computing scenario. Sun et al. [22] proposes a mobile computing edge-based offload solution for Telematics tasks from a global perspective to minimize the average time to complete a task. Yang et al. [23] achieves efficient unloading of vehicle tasks from a game-theoretic perspective and considering the time delay constraint of task unloading. Xiao et al. [24] proposed a mobile-aware partial task offloading algorithm to improve the utilization of vehicle computing resources. Raza et al. [25] A introduces an emerging IoT architecture in the blockchain and implements algorithms for task offloading and resource allocation in the form of smart contracts in the blockchain, ultimately optimizing resource allocation in the IoT. Xiaoa et al. [26] verifies that task offloading optimization is a potential game problem and solves it with a distributed algorithm of Lagrange multipliers. Lan et al. [27] considers the limited computational resources and battery capacity of existing mobile devices resulting in the inability to meet the demand of low computational power and latency, the authors use a potential game model to solve the distributed task offloading problem. Li et al. [28] proposes to describe task offloading as an integer for network load and transmission interference problems. The nonlinear programming problem is a task offloading algorithm based on a differential evolutionary algorithm. Chen et al. [29] proposes an algorithm that considers completion time constraints and task dependency requirements and schedules all tasks from different applications in a priority queue. The aim is to protect low-energy mobile devices and keep them

alive for as long as possible during the allocation process to establish an offloading policy. Fan et al. [30] considers quantitative user experience as an optimization goal for offloading decisions and develops an efficiency-based offloading decision algorithm. Feng et al. [31] proposes two computational offloading algorithms, namely binary offloading and partial offloading. Binary offloading offloads the task as a whole to the mobile edge computing server, and the server selects only the best offload location, thus reducing the total latency cost and energy consumption. Liu et al. [32] proposes an efficient and low-complexity offloading scheme. A series of reconfigurations based on reconfiguration linearization techniques are performed and then programmed by further utilizing the product alternating direction method (ADMM) and the convex function difference method (DC). The authors propose a parallel optimization framework and obtain a more favorable TN with two orders of magnitude time complexity lower than the centralized optimization algorithm. Wang et al. [33] investigates a potential game called multitasking pair offloading (POMT). Liu et al. [34] proposes a two-step method that aims to satisfy the delay constraint while minimizing the energy consumption of IoT devices, ultimately reducing the offloading energy for a given set of delay-constrained offloading tasks. Shan et al. [35] focuses on offloading tasks on a two-tier mobile edge computing environment to minimize the total energy consumed by users; Mazouzi et al. Pham et al. [36] proposes an energy-efficient and deadline-aware task offloading strategy based on channel constraints. It is to minimize the energy consumption of mobile devices while satisfying the constraints of mobile cloud workflow; Liu et al. [37] proposes an algorithm called JOBCA to obtain a feasible solution to the original problem by iteratively solving the urban road and winding road problems. Li et al. [38] investigates task offloading from a matching perspective, aiming to optimize the total network latency. Tucker et al. [39] proposes a load-aware MEC offloading method.

To improve task offloading timeliness and maximize revenue in the spatial information networks with time-varying characteristics, we propose a secondary task offloading and caching model based on maximum flow allocation. Firstly, the model calculates the maximum flow that can circulate on the time-varying network. Moreover, the satellite nodes in the network calculate the task cache based on the high-orbit satellite atoms and transfer the tasks that need to be uploaded to the high-orbit satellite based on the maximum flow limit. When the satellite task load is high, the task will be offloaded and the execution results are cached. When a high-orbiting satellite exceeds the computational tasks that can be loaded, the tasks are offloaded from the high-orbiting satellite to a low-orbiting or medium-orbiting satellite. Meanwhile, the high-orbiting satellite can quickly complete the task computation and response, improving the overall resource utilization.

3. Task Offloading Model

Future space-based network systems will face many resource request tasks. During the mission planning process, all missions are decomposed into indivisible atomic missions, which means that satellites will receive many atomic mission requests. As a result, a single satellite receives more resource requests for atomic tasks than it can provide. Most atomic missions would need to wait for other atomic missions to be executed, which causes high latency and low timeliness. High orbiting satellites have a wide coverage area and the use of inter-layer link technology allows high orbiting satellites to exchange data with targets within their coverage area. Therefore, this paper proposes to transfer individual satellite missions to high-orbiting satellites and use the powerful computing and storage capabilities of high-orbiting satellites to help individual satellites accomplish coordinated resource allocation to solve the problem of inefficient traditional mission execution.

Therefore, the critical problem to be solved in this paper is how to quickly offload the tasks to high-orbit satellites when the satellite receives multiple tasks. Although the resources and capabilities of high-orbit satellites are higher than those of medium-orbit and low-orbit satellites, the number of high-orbit satellites is relatively rare. If the number of missions exceeds the upper limit that the high-orbit satellites can accept, the high-

orbit satellites need to offload the mission to the medium or low-orbit satellites with free resources.

Satellite tasks can be decomposed into atomic tasks. The so-called atomic tasks are the smallest executable tasks that cannot be decomposed any further. Atomic tasks include resource requirements and task execution time. If a high-orbiting satellite has already offloaded the corresponding atomic task, then there is no need for a low-orbiting or medium-orbiting satellite to repeatedly offload the corresponding atomic task.

3.1. Traffic Model

Suppose a given satellite network $G = (V, E, T)$, V denotes all satellites, E denotes the visible time window that can be established between satellite nodes in the network G at time t. If satellite nodes u and v can establish a time window, then this implies that an edge (u, v) exists between two satellites. Otherwise, it is considered that no edge exists. Suppose the task $Q = \{1, 2, \ldots, q\}$. To describe the problem easily, we assume that the number of satellite antennas is 1. That is, only one communication link can be established between satellites at a certain time. After the inter-satellite visible time window is successfully established, an atomic task needs to continue to occupy the time window, and other tasks cannot interrupt the existing atomic task until the atomic task is completed. Let P_{s_i} represent the data transmission power of satellite s_i, and the data transmission rate $r_{u,v}$ between satellite u and satellite v in slot t can be defined as follows,

$$r_{u,v} = w_{u,v} \log_2 \left(1 + \frac{P_s h_{u,v}}{\sigma^2} \right) \tag{1}$$

Among them, σ^2 represents noise power, $w_{u,v}$ represents the channel bandwidth between satellite u and satellite v, $h_{u,v}$ is a constant which denotes the channel gain between satellites.

3.2. Local Calculation Model

Satellites generally have a certain degree of computing power. We define the time consumption of low-medium orbit satellites as follows,

$$T_{s,q}^{lm} = \frac{n_q}{a_s^{lm}}. \tag{2}$$

As shown in Equation (2), $T_{s,q}^{lm}$ denotes the time required for the atomic mission to calculate locally on the satellite s, n_q denotes the number of computing resources required for atomic tasks, a_s^{lm} denotes the computing power of the low-medium orbit satellite's CPU. Similarly, the time consumption of high-orbit satellites can be defined as the following formula,

$$T_{s,q}^{h} = \frac{s_q}{r_{u,v}} + \frac{n_k}{a_s^h} \tag{3}$$

where $T_{s,q}^{h}$ denotes the time required to calculate task q on the high-orbit satellite s, n_q denotes the number of computing resources required for task q, a_s^h denotes the computing power of high-orbit satellites, s_q denotes the data volume of task q.

3.3. Atomic Task Cache Model

We mainly cache atomic tasks received by satellites. The atomic task cache stores the calculation results of high-frequency atomic tasks on high-orbit satellites. The low-medium orbit satellite issues an atomic calculation task offload request. If the atomic calculation task on the low-medium orbit satellite has been cached to the high-orbit satellite, the low-medium orbit satellite does not need to offload the atomic calculation task to the high-orbit satellite repeatedly. We only need to transmit the cached result to the low-medium orbit satellite.

In Definition 1, We define computing task cache decision variables to determine whether to cache an atomic task to a high-orbit satellite.

Definition 1. *Given a satellite network $G = (V, E, T)$, the atomic task request queue Q received by the satellite $s, s \in V$, and q_i indicates a atomic task requests. Then at a certain moment t, $\rho^s_{q_i}(t)$ denotes that whether the calculation result of q_i has been cached, it has a value range in $\{0, 1\}$.*

The Definition 1 is about the cached decision variable range, where $\rho^s_{q_i}(t) = 1$ means that the calculation result of atomic task q_i has been cached, on the contrary, $\rho^s_{q_i}(t) = 0$ means that the atomic task q_i is not cached and needs to be calculated locally. Therefore, the atomic calculation task cache variable on the satellite s can be expressed as $\rho^s_Q(t) = \{\rho^s_{q_1}, \rho^s_{q_2}, \cdots, \rho^s_q\}$.

The use of inter-satellite links can achieve near-real-time data transmission between high-orbit satellites and low-medium orbit satellites. Therefore, we do not consider the time consumption of data transmission between high-orbit satellites and low-medium orbit satellites. In addition, we do not consider packet loss in the task queue, because our goal is to achieve minimal time cost consumption while completing all tasks.

3.4. Atomic Task Offloading Model

To avoid the repetition of the same tasks throughout the satellite network, we design an offloading model for the atomic task queues on the satellite.

As shown in Definition 2, We first define the offloading decision variable on the satellite s.

Definition 2. *Given a satellite network $G = (V, E, T)$, the atomic task request queue Q received by satellite $s, s \in V$, q_i represents one of the atomic task requests. At a slot t, decision variable $\varphi^s_{q_i}(t)$ will be offloading to the high-orbit satellite or not have a value range in $\{0, 1\}$.*

There are two main stages in the offloading decision of satellite tasks. $\varphi^s_{q_i}(t) = 0$ denotes that the current task is calculated locally. $\varphi^s_{q_i}(t) = 1$ means that the task will be offloaded to the high-orbit satellite for execution. Therefore,the decision variable on the satellite s can be expressed as $\varphi^s_Q(t) = \{\varphi^s_{q_1}, \varphi^s_{q_2}, \cdots, \varphi^s_q\}$.

Then we describe the calculation of the number of tasks remaining in the on-star task queue, taking into account the unloading of duplicate tasks in the task.Assuming that the atomic task request queue received by satellite s is $Q_s = \{q_1, q_2, \ldots, q_i\}$, some atomic task requests set ξ that processed by satellite s at time t is expressed as $Q^{\xi}_s(t)$, then the remaining atomic tasks waiting to be executed can be expressed as

$$Q_{rest}(t) = Q_s(t) - Q^{\xi}_s(t) \tag{4}$$

If an atomic mission already exists on a high-orbiting satellite, then the same atomic mission on a low- orbiting or medium-orbiting satellite does not need to be repeatedly offloaded, so the atomic computing mission to be offloaded is represented as follows,

$$Q_{offloading}(t) = Q_{rest}(t) - Q_{repeat}(t), \tag{5}$$

where $Q_{offloading}(t)$ denotes the atomic task queue that needs to be offloaded at time t, $Q_{rest}(t)$ indicates the atomic task queue waiting at time t, $Q_{repeat}(t)$ denotes the atomic task queue that has been buffered on the high-orbit satellite at time t.

Equations (4) and (5) give the process and calculation for updating the on-satellite task queue when a task is unloaded in the paper. The task queue remaining to be executed on the satellite is the original task queue minus the task queue that has already been executed on the satellite. Meanwhile, the tasks need to be calculated before they can be offloaded to ensure that all pending tasks have been executed and that the calculation results are kept. If there are tasks that have already been executed and the computation results have been retained, these tasks will not be offloaded.

3.5. Problem Formulation

To simplify the overly complex problem, we assume that the transfer rates are the same between all low-orbiting and medium-orbiting satellites and the transfer rates are the same between all high-orbiting and low-medium orbiting satellites. Furthermore, all atomic computing tasks on the low-orbiting and medium-orbiting satellites can be transferred to the high-orbiting satellites. Meanwhile, we assume that there are no atomic tasks that cannot be handled by high-orbiting satellites.

Meanwhile, we suppose that satellite networks can perceive each other and there is a visible time window between low-orbit and medium-orbit satellites, which links can be successfully established. Then, the time required to complete the task q execution on the satellite with the atomic task request in the initial state can be expressed as,

$$c_s(t) = \varphi_q^s(t) T_{s,q}^{lm} + \left(1 - \varphi_q^s(t)\right)\left(1 - \rho_q^s(t)\right) T_{s,q}^{h} \tag{6}$$

where $c_s(t)$ denotes the time required for the task execution on the satellite s at time t. Among them, $1 - \varphi_q^s(t)$ indicates that the task is offloaded and needs to be executed on a high-orbit satellite. Similarly, $1 - \rho_q^s(t)$ indicates that the task has been cached.

The paper aims to scheme the execution order of all tasks and the satellite resource allocation scheme so that the time consumed by the execution of all tasks is minimized. The problem can be formally formulated as

$$\begin{aligned} &\min_{s,q} \sum_{s \in S} \sum_{q \in Q_s} c_s(t) \\ &s.t. \\ &C1: T_{s,q}^{lm} \leq D_s, \ \forall s \in S, q \in Q_s, \\ &C2: \rho_{q_i}^s(t) \in \{0,1\}, \\ &C3: \varphi_{q_i}^s(t) \in \{0,1\}, \end{aligned} \tag{7}$$

As shown in Formula (7), $C1$ indicates that the atomic task needs to be computed by the deadline D_s, $C2$ indicates the range of values for the atomic task caching decision, $C3$ indicates the range of values for the atomic task unloading decision.

4. Method

To achieve the overall goal of time consumed by the execution of all tasks is minimized, we further consider the actual task planning process. When faced with multimedia and new space tasks, satellites will receive many computational tasks, which are broken down into more atomic tasks. The existing satellite computing power cannot cover all the atomic tasks. Therefore, it is necessary to offload some computational tasks to high-orbiting satellites with solid computational capabilities to reduce the single-satellite computational load, improve the efficiency of mission execution, and speed up the response time to computational tasks.

4.1. Problem Conversion

High orbit satellites have comprehensive coverage and better performance than that of low-medium orbit satellites. However, the number of high-orbiting satellites is sparse in practice. Therefore, if many atomic calculations are offloaded to high-orbiting satellites for execution, the processing capacity of high-orbiting satellites may not receive all the offloaded tasks. Therefore, we propose a re-offloading method for high-orbiting satellites considering the practical situation.

As shown in the Figure 2, the satellites s_1, s_2, s_5 and s_6 are all medium and low orbit satellites, where s_1 and s_2 are satellites with atomic missions in the initial state. s_1 and s_2 first consider whether the computational tasks need to be offloaded in the received atomic missions. If the atomic mission needs to be offloaded, a data transfer link is established with the high-orbiting satellites to offload the mission to the high-orbiting satellites s_3 and

s_4. If the number of computational tasks on the high-orbiting satellites s_3 and s_4 exceeds the limit and cannot continue to receive task offload requests from low- and medium-orbiting satellites, then the advantage of the task offload function will no longer exist. The computational tasks requiring task offloading need to wait for local computational resources to become available before they can be executed, as in the traditional case. In order to keep the advantages of mission offloading always available and to efficiently use idle satellite resources, we offload atomic tasks that can no longer be processed by high-orbiting satellites to other idle satellites for execution, as shown in Figure 2. The in-orbit satellites s_3 and s_4 offload computational tasks that they cannot process to other idle satellites s_5 and s_6.

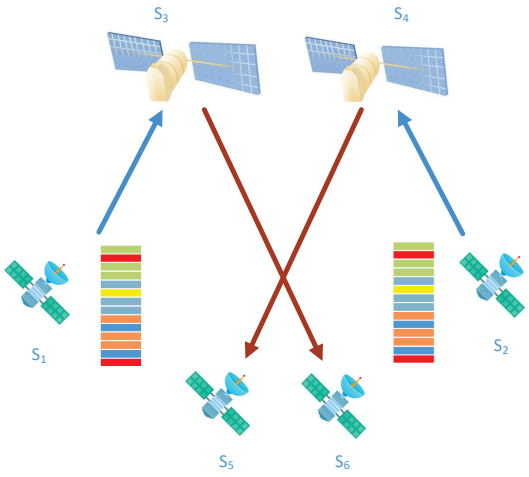


Figure 2. Mission offloading and re-offloading process of all the satellites.

Due to the frequent dynamic changes in the space-based network, the location of satellites also changes dynamically, while the number of mission requests from satellites also changes over time. Therefore, we have to solve the problem of offloading atomic missions from satellites in dynamic network environments. According to the analysis in the above chapters, we can divide the minimum time optimization problem of atomic missions into two parts. The first part is the offloading of atomic missions. Offload some of the atomic missions from the medium and low orbit satellites to the high orbit satellites. This reduces the computational pressure on the low-medium orbiting satellites and makes full use of the high-orbiting satellites' large amount of computational power. The second part is the caching of atomic tasks, which means that there will be more duplication of atomic computing tasks on the medium and low orbiting satellites. Suppose that an atomic computing task on a low-medium orbiting satellite is offloaded to a high-orbiting satellite. In this case, we only need to wait for the high orbiting satellite to return the computation results to the medium and low orbiting satellites, avoiding duplicate buffering and reducing duplicate offloads to save data transfer time.

At the same time, we observe that multiple Low Earth Orbit (LEO) satellites need to offload some of their atomic tasks to the High Elliptical Orbit (HEO) satellites, which is caused by the number of antennas in the HEO satellites. The antennas can only match the communication links established by one or a few satellites, which causes other satellites to wait until the other satellites end their occupation of the high-orbiting satellite links. Waiting for link resources in this way is unwise and a waste of resources in a spatial information network that changes dynamically in space and time. Therefore, we re-decompose the problem into three parts.

In summary, this paper aims at the low utilization of spatial information network resources, poor timeliness, and long waiting time for atomic tasks problems. The paper proposes the idea of solving resource contention based on secondary task offloading. The problem is to improve the overall resource utilization rate and achieve the goal of quickly responding to many atomic tasks and assigning specified computing resources.

As shown in the Figure 3, the blue satellites in the middle and low orbit satellites on the left represent the initial state of the satellites at a specific time t. Meanwhile, these blue satellites have received many atomic computing task requests. Link resources between low-medium orbit satellites and high-orbit satellites are limited, and it is not advisable to establish links between many low-medium orbit satellites and a specific high-orbit satellite simultaneously. Therefore, allocating the offloading tasks is crucial if multiple satellites need to offload a certain number of atomic tasks to high-orbit satellites at time t. As shown in Figure 3, we consider the satellites as nodes and the links between them as edges to create a small directed network. By calculating the maximum traffic of the network, we can observe the satellite network in red in the virtual box on the left.

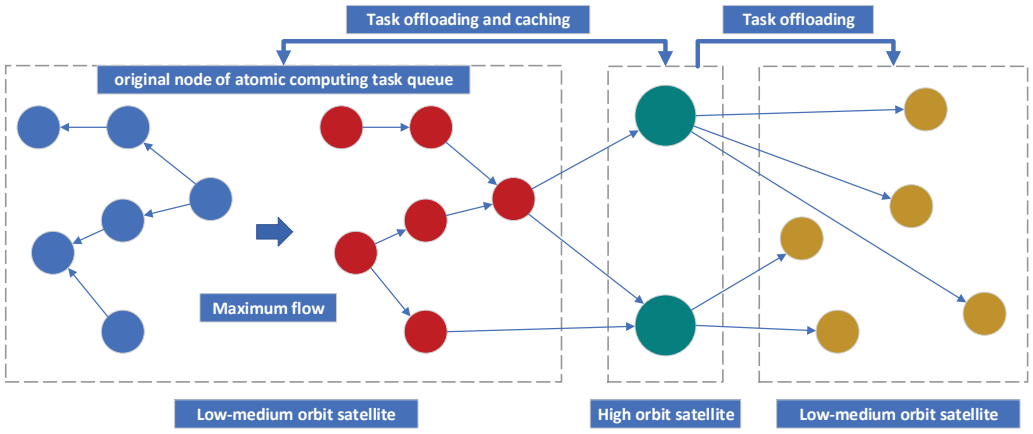


Figure 3. Mission offloading of low-medium orbit satellites and re-offloading process of high-orbit satellites.

Meanwhile, we can know the maximum amount of data transmitted on each edge according to the solution result of the maximum flow. According to the edge source node, we can obtain the maximum tasks that the node can unload at the current time t. The calculation tasks that need to be offloaded on other satellites are transmitted to a low-medium orbit satellite that establishes a communication link with the high-orbit satellite. Then the satellite transmits the atomic calculation task request to the high-orbit satellite. As shown in Figure 3, the green node in the middle is the high-orbit satellite. After receiving numerous computing tasks on the high-orbit satellite, if the number of tasks reaches the threshold of tasks acceptable to the high-orbit satellite, then unload the remaining computing tasks that cannot be received by high-orbit satellites to the yellow low-medium orbit satellites in the idle state. Therefore, the above process includes a one-time task cache and two-time task offloads.

4.2. Algorithm Design

4.2.1. Maximum Task Flow Transmission

Satellites change with time and space, so the visible time window between satellites is limited, which leads to precious link resources between satellites. If multiple satellites need to offload unquantized missions to high-orbiting satellites, the time window limitation can make it difficult. Therefore, we would need to calculate the maximum amount of data to be transferred on a network consisting of satellites that currently need to offload atomic

missions. We transform part of the problem into the problem of calculating the maximum flow in a dynamic network, which can be expressed as follows,

$$\begin{aligned} &\max \sum_{(u,v)\in E} f(u,v) \\ &s.t. \\ &C1: \sum_{(u,v)\in E} f(u,v) - f(v,u) = 0, \ (u \neq s,t), \\ &C2: \sum_{(u,v)\in E} f(i,v) - f(v,j) = v(f), \ (i = s), \\ &C3: \sum_{(u,v)\in E} f(u,v) - f(v,u) = -v(f), \ (u = t), \end{aligned} \tag{8}$$

As shown in the Formula (8) f denotes a feasible flow of a link in a satellite network, u and v denote the satellite nodes in the network which can establish a time window, s denotes the only source node with indegree zero, and t denotes the only sink node with outdegree zero. The Formula (8), which solves a feasible flow in the network to maximize the flow $v(f)$ on the network, where the constraint $C1$ represents the conservation of flow, and the so-called conservation of flow refers to the intermediate node no flow is stored. When initially dividing the number of satellite tasks to unload, we can assume that the node does not store the flow in solving the maximum flow, and the task is unloaded after the maximum flow is solved. $C2$ and $C3$ indicate the need to meet traffic demand. In addition, the bandwidth constraints of the link need to be satisfied.

As shown in Figure 4, it describes the process of calculating the maximum flow on a given instantaneous satellite network. The result of the maximum flow calculation can be used to determine the maximum number of task offloads that can be transmitted on a satellite. As shown in Subfigure (a) represents a small satellite network consisting of low-medium orbiting satellites at t time. There is a visible time window between the satellites, and each satellite has an atom that needs to be offloaded. During the task offloading process, sources, and sinks are not unique in the network. Subfigure (b) represents the computational process of the maximum flow. First, we create two virtual nodes as unique sources and sinks in the satellite network. We can use the single source and single sink maximum flow calculation method to solve the satellite network. The maximum traffic that can be transmitted on the subgraph Subfigure (c) is the number of missions that can be offloaded from each satellite based on the result of the maximum traffic solution. The missions to be transmitted are then transferred to the specific satellite and the offloaded mission data is transmitted by the satellite to the high-orbiting satellite.

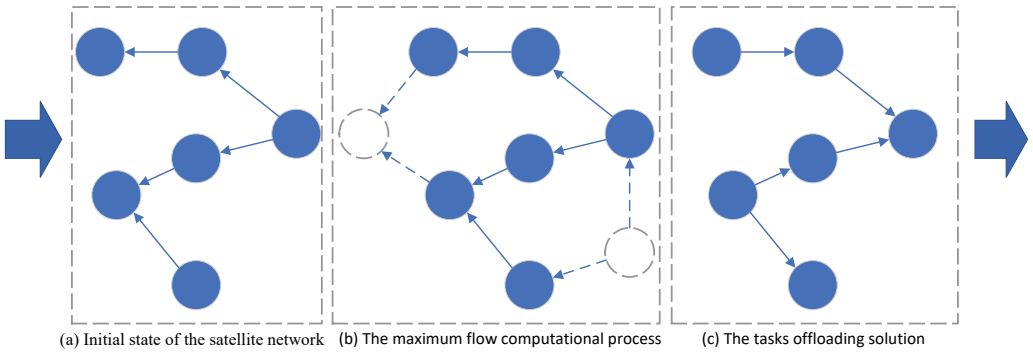


Figure 4. Maximum flow calculation method for task offloading quantity allocation of dynamic satellite network.

The solving process of the maximum tasks offloaded on each satellite in the satellite network composed of low-medium orbit satellites is shown in the Algorithm 1. It mainly calculates the maximum traffic path of the network composed of all low-medium orbit satellites that receive the atomic task request queue and the maximum allowable transmission traffic. Considering the satellite network has time-varying characteristics, we calculate the task offloading plan in satellite networks over time. First, given a fixed period and a graph composed of satellites and communication links $G^{'}$ (lines 1–2). We first obtain the number of tasks that need to be unloaded on each satellite (line 5) and then use the classic Ford-Fulkerson [40] algorithm to calculate the maximum traffic path and traffic distribution on the graph $G^{'}$ at the current moment (line 6). If the number of unloadable tasks on a satellite s is greater than the maximum number of tasks allowed in the maximum stream result, only the number of tasks allowed in the maximum stream result will be allowed to be transmitted. Conversely, the number of tasks to be offloaded on the transmission satellite is allowed (lines 7–11). Finally, the results of all task offloading assignments during the period are returned.

Algorithm 1 Optimal allocation of task offloading quantity

Input: Given a directed graph $G = (V, E, T)$, Given time range τ, Given the atomic task request queue v_q on each satellite.
Output: The number of atomic tasks that can be unloaded by each satellite in a certain period $O_{timeRange}(V)$.

1: set $timeRange = \tau$
2: set $G^{'} = G(V, E)$
3: **repeat**
4: **repeat**
5: Calculate the number of tasks that need to be unloaded by the satellites $O_t(V)$;
6: [$route$,f]=FordFulkerson($G^{'}$);
7: **if** $task(s) > f(s, v)$ **then**
8: $O_t(s) = f(s, v)$
9: **else**
10: $O_t(s) = task(s)$
11: **end if**
12: **until** V
13: **until** $timeRange$
14: **return** $O_{timeRange}(V)$

4.2.2. Optimal Task Offloading

Through the above calculation, we can get the maximum number of tasks that can be offloaded on the satellite, so the Formula (7) needs to meet the constraint of the maximum task offload. From the task offloading optimization problem proved in the literature [41], the task offloading problem can be transformed into a convex optimization problem.

Given $\rho = \rho^0$, the Formula (7) can be transformed into a convex optimization problem. It can be regarded as a function of ρ and redefined as a function of $f(\varphi_s)$, which denotes as follows,

$$\begin{aligned} & f(\varphi_s) = \varphi_s T_{s,q}^{lm} + (1 - \varphi_s)\left(1 - \rho^0\right) T_{s,q}^{h} \\ & s.t. \\ & C1: T_{s,q}^{lm} \leq D_s, \forall s \in V, q \in Q, \\ & C2: \varphi_s \in \{0, 1\}, \ \forall s \in V, \\ & C3: \sum_{q=1}^{Q} \varphi_s^q \leq O(s), \ \forall s \in V, q \in Q, \end{aligned} \tag{9}$$

As shown in Equation (9), $f(\varphi_s)$ denotes the task offloading decision objective function with φ as the variable, where $C1$ indicates that the task execution cannot exceed the time threshold. $C2$ indicates that the value range of the decision variable is 0 or 1, and $C3$ indicates the task of satellite offloading. The quantity cannot exceed the amount of data that can be accommodated by the transmission link's maximum bandwidth and cannot exceed the amount of task data transferred by the maximum stream scheme.

As shown in Algorithm 2, according to the calculation result of the maximum flow calculation, obtain the upper limit of the task that the satellite can unload at the current moment (line 5). The number of buffers for a mission is constant, so a constant value is assigned to the number of buffers available for each satellite (line 6). According to the Formula (9), use the Lagrangian expansion to solve and return the result of task offloading quantity (lines 7–11).

Algorithm 2 Satellite mission offloading optimization calculation

Input: Given a directed graph $G = (V, E, T)$, Given time range τ, Given the atomic task request queue v_q on each satellite. **Output:** An array of offloading tasks $arr(V)$

1: set $timeRange = \tau$
2: set $G^{'} = G(V, E)$
3: **repeat**
4: **repeat**
5: Calculate the maximum number of tasks that can be uninstalled $O(V)$
6: set $\rho_v = cache(v)$
7: set $\varphi(v) = Lagrange(v)$
8: set $arr(v).add(\varphi(v))$
9: **until** V
10: **until** $timeRange$
11: **return** $arr(V)$

4.2.3. Optimal Task Cache

In solving the task cache problem, we can treat the task offload variable as a constant. Meanwhile, the Formula (7) can be transformed into a function with ρ as the independent variable. The formula as shown in the follows,

$$\begin{aligned} & f(\rho_s) = \left(1 - \varphi^0\right)(1 - \rho_s)T_{s,q}^h \\ & s.t. \\ & C1 : T_{s,q}^{lm} \leq D_s, \forall s \in V, q \in Q, \\ & C2 : \rho_s \in \{0, 1\}, \ \forall s \in V. \end{aligned} \tag{10}$$

As shown in (10), $C1$ denotes that the task needs to be completed within the deadline, and $C2$ indicates the value range of the task cache state.

The Formula (10) can be transformed into a linear programming problem about φ. This problem can be solved using the branch and bound method to solve the optimal value [41–43]. The specific process of the algorithm is shown as follows.

As shown in Algorithm 3, the satellite's task set is converted into a tree, which represents all possible combinations of tasks that need to be cached locally (line 5). The abounding function prunes some non-optimal branches in the tree structure. Each branch uses the depth-first rule DFS to calculate its cost. When all branches are searched, the algorithm returns the optimal solution [41].

Algorithm 3 Satellite mission buffer calculation method

Input: Given a directed graph $G = (V, E, T)$, Given time range τ, Given the atomic task request queue v_q on each satellite. **Output:** An array of offloading tasks $C(V)$

```
set timeRange = τ
set G' = G(V, E)
repeat
  repeat
    repeat
      set tree(s) = tranToTree(O(V))
    until V
    repeat
      C(V) = search(tree)
    until tree
until timeRange
return C(V)
```

As for the influence of the cache size on the offloading decision, the cache of tasks is realized by using the storage and communication capabilities of high orbit satellites. High orbiting satellite resources are precious, so they can not occupy too much space for caching. The cache size depends on the frequency of similar tasks, and tasks with high frequency can be cached. We cache the task calculation results that occur many times to the satellite. The larger the cache space, the fewer tasks to offload, and the greater the total revenue of the satellite network.

5. Numerical Experiments

5.1. Time Complexity

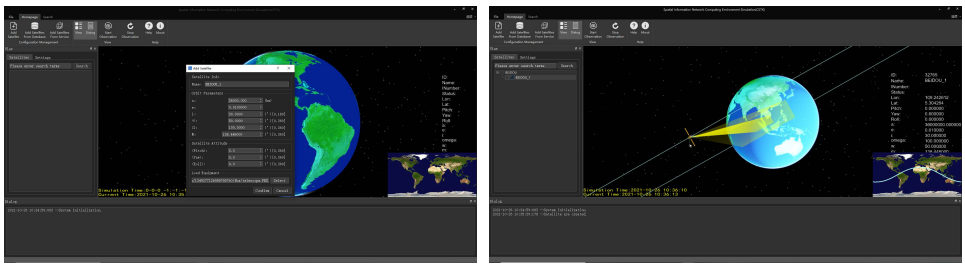
We divide the offloading of satellite atomic tasks into three parts: the maximum stream transmission calculation, the optimization of atomic task offloading, and the optimization of task cache. This paper designs corresponding optimization solutions for these three problems. As shown in the Algorithm 1, the algorithm mainly uses the Ford-Fulkerson algorithm to calculate the maximum flow value and path of the network. The algorithm's time complexity is $O(Ef)$, where E is the edges, and f is the maximum flow in the graph. Each expansion path can be found within the time of $O(E)$ and increase the traffic by at least the entire amount of 1, with an upper limit of f. According to the time range and the calculation of each satellite, the algorithm includes period traversal and satellite nested traversal. The total time complexity of the algorithm is $O(TEVf)$. The atomic task offloading optimization Algorithm 2 mainly uses the Lagrangian expansion and Karush-Kuhn-Tucker (KKT) conditions to solve the optimization problem. The algorithm also considers the traversal of satellites in discrete time and the traversal of tasks on each satellite, so the algorithm time complexity is $O(TV)$. The task cache optimization algorithm mainly involves constructing and searching the node tree. The algorithm's time complexity in discrete time is $O(T(V + VE))$. In summary, the algorithm time complexity of the model *OCMF* proposed in this paper is $O(TV(Ef + 1 + E + 1))$, which can be simplified to $O(TEVf)$. In the whole process of computing task offloading algorithm, f is constant, so the time complexity of *OMD* algorithm can be expressed as $O(TEV)$.

In space information networks, the location of satellites changes continuously with time, and the visible time window between satellites also changes dynamically. The communication cost between low-orbiting satellites is different from that between high-orbiting satellites, which can be found in the Formulas (2) and (3). The link resources between satellites are valuable, and most satellites have weak storage and computational capabilities. Therefore, if an algorithm has excellent time complexity, it can save a lot of waiting time, optimize the mission execution efficiency in general, reduce the mission waiting time, and improve resource utilization. Research on task offloading is increasingly inclined to use deep reinforcement learning methods. However, the learning time of deep reinforcement

learning methods is long and cannot meet the low latency requirements for task offloading in time-varying satellite networks.

5.2. Experimental Fundamentals

To study the key problems of mission scheduling, link optimization, collaborative computing, and network topology discovery in spatial information networks. We independently developed a Chinese Satellite Tool Kit (CSTK) based on actual satellites and common mission data in spatial information networks, which is based on the Java Agent DEvelopment Framework (JADE) (accessed on 20 October 2021 for link https://jade-project.gitlab.io/) framework, using the two-line element (TLE) set for the satellite and calculates satellite positions based on the Sgp4 and sdp4 packages [44]. The system simulates satellite data, including satellite payload data, satellite orbit data, inter-satellite visible time window and satellite ground visible time window data, satellite resource capacity, and satellite resource quantity. In addition, we simulated data for common Earth observation application scenarios, including Earth observation area, mission execution time demand, mission resource type, and mission resource demand. In addition, we developed various computational libraries required for the CSTK system. The experiments in this paper are done based on the CSTK experimental simulation platform, coded in Python. CSTK calculates the orbits of subsatellite points in the two-dimensional plane using the satellite orbit parameter data provided by celestrak (https://celestrak.com/, accessed on 20 October 2021). The main software interface of CSTK is shown in the Figure 5.



(**a**) Satellite management (**b**) Observation scene

Figure 5. Spatial information network computing environment simulation platform.

We complete the performance validation of the numerical experiment based on the simulation capabilities of satellite and mission data provided by the CSTK platform. In this experiment, we use multiple datasets to validate the performance of different algorithms. Each dataset contains satellite and task data of varying sizes. We observe the effectiveness of our algorithm and baseline model approach by the running results on each dataset. The experimental data details are shown in Table 1.

Table 1. Multi-scale experimental datasets.

ID	Scale	
	Tasks	**Satellites**
D1	50	10
D2	100	20
D3	500	50
D4	1000	100
D5	1500	200
D6	5000	500

As shown in Table 1, where 'ID' denotes the dataset number, 'Scale' denotes the satellite and task scales. From the above table, it can be observed that our experiment uses six datasets with increasing size of the dataset. The minimum number of missions is 50,

the maximum is 5000, the minimum number of satellites is 10, the maximum number is 500. With the increase of data set size, the satellite network needs to deal with more tasks, and the interaction between satellites is more complex, which requires higher timeliness of task processing algorithm. And this is also closer to the explosive growth trend of multimedia services and the continuous emergence of new space tasks.

5.3. Parameter Setting

As shown in Table 2, where the parameter '*gamma*' denotes the kernel coefficient and its values are selected in the range of $0.01, 0.1, 1, 10$, the parameter '*n_clusters*' denotes the dimension of the projection subspace and its values are selected in the range of $2, 3, 4, 5, 6$, the parameter '*n_neighbours*' denotes the number of neighbors to use when constructing the affinity matrix using the nearest neighbors method and its value is 10, the parameter '*degree*' denotes the degree of the polynomial kernel and its value is 3. We select the optimal aggregation result from the above set by training, and the set of three categories after the clustering is completed selected by default. The default minimum start time for all tasks should be greater than 0.

Table 2. Spectral clustering parameters setting in OCMF process.

gamma	*n_clusters*	*n_neighbors*	*Degree*
0.01, 0.1, 1, 10	2, 3, 4, 5, 6	10	3

As shown in Table 3, where the parameter lb denotes the minimum value of each independent variable, ub denotes the maximum value of each independent variable, and the maximum value of each variable comes online as the number of resources currently owned by the satellite. $prob\_mut$ denotes the variance probability, w denotes the inertia weight of algorithm PSO, $c1$ denotes the individual memory of algorithm PSO, $c2$ denotes the collective memory of algorithm PSO, F denotes the variance coefficient of algorithm DE, $size\_pop$ denotes the population size.

Table 3. Experimental parameter setting of baseline model.

Algorithms	*lb*	*ub*	*prob_mut*	*w*	*c*1	*c*2	*F*	*size_pop*
PSO	0	-	-	0.8	0.5	0.5	-	200
GA	0	-	0.001	-	-	-	-	200
DE	0	-	0.001	-	-	-	0.5	200

5.4. Results Analysis

The time cost consumption of task $c(q_i)$ includes the task waiting time $w(q_i)$ and the task execution time $e(q_i)$. Ideally, if the resources are sufficient, then the task does not need to wait and the time cost of the task only includes the task execution time. However, the Ideal Scenario of Sufficient Resources (ISSR) is extremely difficult to occur on spatial information networks. In this experiment, the approach that approximates the time cost consumption of ISSR is more effective. In addition, the First Come, First Service (FCFS) scheduling method executes the task queues on the satellite according to the task order. Therefore, the waiting time of the current task includes the sum of the execution time of the previous tasks and the waiting time, i.e., $w(q_{i+1}) = c(q_i)$, then $c(q_{i+1}) = c(q_i) + e(q_{i+1})$. We analyze the task time cost incurred by the method OCMF with the cost consumption of ISSR, FCFS, PSO, GA, and DE methods. The numerical results of the simulation experiments are shown in Table 4.

Table 4. Time cost calculation results for multiple algorithms on different datasets.

Algorithm	Time Consumption (s)					
	D1	D2	D3	D4	D5	D6
ISSR	18.94	31.82	157.46	294.19	471.91	1530.55
OCMF	14.23	30.55	158.28	292.76	452.11	1523.86
FCFS	48.53	92.22	543.37	976.78	1422.89	5019.90
PSO	65.07	116.07	3635.15	17,667.49	52,237.70	113,016.89
GA	63.16	107.67	920.51	17,702.55	50,301.48	44,665.74
DE	66.04	110.88	7904.71	34,649.01	80,317.24	62,888.36

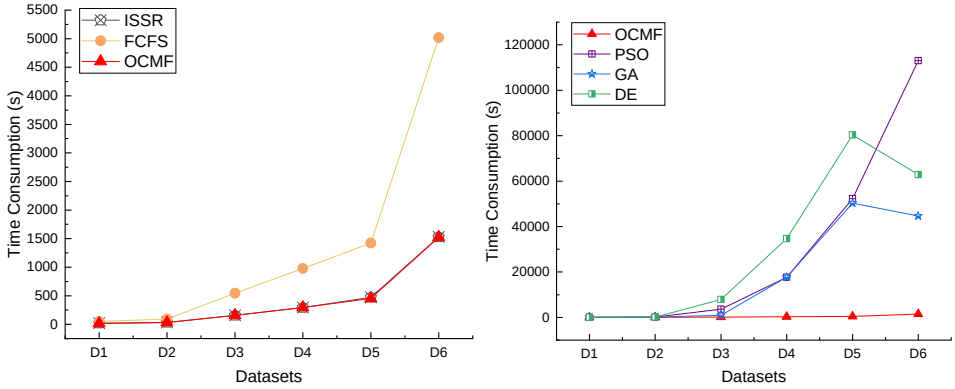
As shown in Table 4, we can observe the task time calculation results of OCMF and multiple baseline algorithms on six datasets. The experimental results show that the task time consumed by the OCMF approach increases with the growing datasets. The incremental time cost of OCMF on datasets D1 and D2 are relatively small. However, the increment of time cost increases by the increasing amount of the datasets. Moreover, OCMF has lower time consumption than ISSR, FCFS, PSO, GA, and DE algorithms on multiple datasets.

The ISSR time cost includes executing all tasks themselves without the waiting time consumption. We can notice that the OCMF time consumption is slightly lower than ISSR, which is caused by the caching mechanism in the proposed method OCMF. If a task has been executed in the previous time and the task execution result is cached, then the same task can be accessed without repeated execution and the result data can be obtained from the cache of the high-orbiting satellite.

FCFS is a traditional task scheduling method that considers the task wait time and the task execution time. The task waiting time includes the execution time of the previous tasks and the waiting execution time. Although FCFS does not include the inter-satellite link transmission time consumption, the algorithm has the disadvantage of large waiting time consumption. Therefore, the FCFS scheduling method is inferior to OCMF.

Although the swarm intelligence algorithm has a good performance in various application scenarios, the results on space-based information networks are not promising due to the tall task offloading operations times of the group intelligence algorithm, which leading a large time consumption for inter-satellite link transmission. As the Table 4 shows, we can observe that the PSO, GA, and DE algorithms have more significant time cost consumption, whereas the GA algorithm has lower time cost consumption. PSO time cost on dataset D4 is much lower than the DE time cost on dataset D4. However, PSO the time consumption on dataset D6 is much larger than that of DE on dataset D6. As the dataset increases, the growth rate of PSO cost consumption is greater than that of DE.

As shown in Figure 6, the horizontal coordinates indicate the six scale-up datasets and the vertical coordinates indicate the time consumption calculated by different methods. Subplot (a) depicts the time comparison between OCMF, ISSR, and FCFS, where the time consumption trace of OCMF and ISSR are similar. However, ISSR is a desirable hypothetical approach. The time cost tends of FCFS increase with the data volume increases and the growth rate reaches the highest at dataset D6. Therefore, it is not suitable for scheduling scenarios with high volume tasks which causes the time consumption growth rate of FCFS to be faster with the growing datasets. Subfigure (b) depicts the OCMF, PSO, GA, and DE methods performance comparison result, where the OCMF method has the lowest time consumption and the growth rate does not change significantly with the dataset scale increases. During the dataset D1 to D5, the time cost growth rate of the group wisdom algorithm shows an increasing trend. The growth rates of GA and DE show a decreasing trend on dataset D6. Therefore, GA and DE have advantages in dealing with large-scale task scheduling problems. According to subfigures (a) and (b), it can be seen that the proposed algorithm OCMF can better adapt to the growth of the task scale and has less cost consumption.



(**a**) (**b**)

Figure 6. Performance comparison between OCMF and various classic methods. (**a**) Performance comparison between OCMF, ISSR and FCFS approach. (**b**) Performance comparison between OCMF, PSO, GA and DE approach.

6. Conclusions

This paper studies the offloading of atomic tasks in space-based networks. Aiming at the pain points of high response latency of multi-task requests, poor task calculation timeliness, and poor resource utilization in the spatial information network, we propose a low time complexity algorithm based on graph-related theories and edge computing-related ideas to achieve optimal offloading of satellite atomic tasks and cache and make full use of satellite resources. We compared the time complexity of several baseline algorithms for task offloading and analyzed the performance between the algorithms and found that our method has certain advantages in terms of time complexity and task execution timeliness. The time complexity of the algorithm designed in this paper depends mainly on solving the maximum flow. We optimize the maximum flow algorithm for dynamic networks to effectively improve the overall algorithm calculation efficiency so that we improve satellites' overall computing power and timeliness and improve the high-latency status of task calculations. In future research, there are still some problems to be solved. The model constraints constructed in this article are not perfect, such as power and radiation constraints. In general, the calculation methods for the maximum number of tasks, task offloading and caching strategies, and the second offloading of atomic tasks proposed in this paper can provide some new ideas for improving the timeliness and resource utilization of space-based networks.

Author Contributions: Conceptualization, Y.L.; methodology, H.F. and Y.L.; software, H.F.; validation, Y.L. and H.F.; formal analysis, H.F. and Y.L.; investigation, Y.L.; resources, H.F. and Y.L.; data curation, H.F. and W.Z.; writing—original draft preparation, H.F.; writing—review and editing, H.F. and Y.L.; visualization, H.F. and W.Z.; supervision, Y.L.; project administration, Y.L. and W.Z.; funding acquisition, Y.L. and W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant U1911401 and the National Natural Science Foundation of China under Grant U1435215.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Yao, H.; Wang, L.; Wang, X.; Lu, Z.; Liu, Y. The space-terrestrial integrated network: An overview. *IEEE Commun. Mag.* **2018**, *56*, 178–185. [CrossRef]
2. Gür, G. Spectrum sharing and content-centric operation for 5G hybrid satellite networks: Prospects and challenges for space-terrestrial system integration. *IEEE Veh. Technol. Mag.* **2019**, *14*, 38–48. [CrossRef]
3. Bi, Y.; Han, G.; Xu, S.; Wang, X.; Lin, C.; Yu, Z.; Sun, P. Software defined space-terrestrial integrated networks: Architecture, challenges, and solutions. *IEEE Netw.* **2019**, *33*, 22–28. [CrossRef]
4. Chan, V.W. Optical satellite networks. *J. Light. Technol.* **2003**, *21*, 2811. [CrossRef]
5. Xu, S.; Wang, X.W.; Huang, M. Software-defined next-generation satellite networks: Architecture, challenges, and solutions. *IEEE Access* **2018**, *6*, 4027–4041. [CrossRef]
6. Kota, S.L. Multimedia satellite networks: Issues and challenges. In *Multimedia Systems and Applications*; International Society for Optics and Photonics: Bellingham, WA, USA, 1999; Volume 3528, pp. 600–618.
7. Tang, Z.; Zhao, B.; Yu, W.; Feng, Z.; Wu, C. Software defined satellite networks: Benefits and challenges. In Proceedings of the IEEE Computers, Communications and IT Applications Conference, Beijing, China, 20–22 October 2014; pp. 127–132.
8. Zhou, S.; Wang, G.; Zhang, S.; Niu, Z.; Shen, X.S. Bidirectional mission offloading for agile space-air-ground integrated networks. *IEEE Wirel. Commun.* **2019**, *26*, 38–45. [CrossRef]
9. Shang, B.; Yi, Y.; Liu, L. Computing over space-air-ground integrated networks: Challenges and opportunities. *IEEE Netw.* **2021**, *35*, 302–309. [CrossRef]
10. Wang, J.; Hu, J.; Min, G.; Zomaya, A.Y.; Georgalas, N. Fast Adaptive Task Offloading in Edge Computing based on Meta Reinforcement Learning. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 242–253. [CrossRef]
11. Gao, Z.; Hao, W.; Han, Z.; Yang, S. Q-Learning-Based Task Offloading and Resources Optimization for a Collaborative Computing System. *IEEE Access* **2020**, *8*, 149011–149024. [CrossRef]
12. Liu, L.; Qin, X.; Zhang, Z.; Zhang, P. Joint Task Offloading and Resource Allocation for Obtaining Fresh Status Updates in Multi-Device MEC Systems. *IEEE Access* **2020**, *8*, 38248–38261. [CrossRef]
13. Sun, Y.; Wei, T.; Li, H.; Zhang, Y.; Wu, W. Energy-Efficient Multimedia Task Assignment and Computing Offloading for Mobile Edge Computing Networks. *IEEE Access* **2020**, *8*, 36702–36713. [CrossRef]
14. Zhang, N.; Guo, S.; Dong, Y.; Liu, D. Joint task offloading and data caching in mobile edge computing networks. *Comput. Netw.* **2020**, *182*, 107446. [CrossRef]
15. Lu, H.; Gu, C.; Luo, F.; Ding, W.; Liu, X. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Gener. Comput. Syst.* **2020**, *102*, 847–861. [CrossRef]
16. Yan, P.; Choudhury, S. Optimizing Mobile Edge Computing Multi-Level Task Offloading via Deep Reinforcement Learning. In Proceedings of the International Conference on Communications (ICC)—ICC 2020, Dublin, Ireland, 7–11 June 2020; pp. 1–7.
17. Lu, H.; Gu, C.; Luo, F.; Ding, W.; Zheng, S.; Shen, Y. Optimization of Task Offloading Strategy for Mobile Edge Computing Based on Multi-Agent Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 202573–202584. [CrossRef]
18. Yan, J.; Bi, S.; Zhang, Y.J.A. Offloading and Resource Allocation With General Task Graph in Mobile Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 5404–5419. [CrossRef]
19. Zhang, Q.; Gui, L.; Hou, F.; Chen, J.; Zhu, S.; Tian, F. Dynamic Task Offloading and Resource Allocation for Mobile-Edge Computing in Dense Cloud RAN. *IEEE Internet Things J.* **2020**, *7*, 3282–3299. [CrossRef]
20. Gavras, A.; Karila, A.; Fdida, S.; May, M.; Potts, M. Future Internet Research and Experimentation: The FIRE Initiative. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 89–92. [CrossRef]
21. Wang, K.; Wang, X.; Liu, X.; Jolfaei, A. Task Offloading Strategy Based on Reinforcement Learning Computing in Edge Computing Architecture of Internet of Vehicles. *IEEE Access* **2020**, *8*, 173779–173789. [CrossRef]
22. Sun, J.; Gu, Q.; Zheng, T.; Dong, P.; Valera, A.; Qin, Y. Joint Optimization of Computation Offloading and Task Scheduling in Vehicular Edge Computing Networks. *IEEE Access* **2020**, *8*, 10466–10477. [CrossRef]
23. Yang, S. A Task Offloading Solution for Internet of Vehicles Using Combination Auction Matching Model Based on Mobile Edge Computing. *IEEE Access* **2020**, *8*, 53261–53273. [CrossRef]
24. Xiao, Z.; Dai, X.; Jiang, H.; Wang, D.; Chen, H.; Yang, L.; Zeng, F. Vehicular Task Offloading via Heat-Aware MEC Cooperation Using Game-Theoretic Method. *IEEE Internet Things J.* **2020**, *7*, 2038–2052. [CrossRef]
25. Raza, S.; Liu, W.; Ahmed, M.; Anwar, M.R.; Wang, S. An efficient task offloading scheme in vehicular edge computing. *J. Cloud Comput. Adv. Syst. Appl.* **2020**, *9*, 28. [CrossRef]
26. Xiao, K.; Gao, Z.; Shi, W.; Qiu, X.; Yang, Y.; Rui, L. EdgeABC: An architecture for task offloading and resource allocation in the Internet of Things. *Future Gener. Comput. Syst.* **2020**, *107*, 498–508. [CrossRef]
27. Lan, Y.; Wang, X.; Wang, D.; Liu, Z. Task Caching, Offloading and Resource Allocation in D2D-Aided Fog Computing Networks. *IEEE Access* **2019**, *7*, 104876–104891. [CrossRef]
28. Li, Y.; Jiang, C. Distributed task offloading strategy to low load base stations in mobile edge computing environment. *Comput. Commun.* **2020**, *164*, 240–248. [CrossRef]
29. Chen, X.; Liu, Z.; Chen, Y.; Li, Z. Mobile Edge Computing Based Task Offloading and Resource Allocation in 5G Ultra-Dense Networks. *IEEE Access* **2019**, *7*, 184172–184182. [CrossRef]

30. Fan, Y.; Zhai, L.; Wang, H. Cost-Efficient Dependent Task Offloading for Multiusers. *IEEE Access* **2019**, *7*, 115843–115856. [CrossRef]
31. Feng, W.; Yang, C.; Zhou, X. Multi-user and multi-task offloading decision algorithms based on imbalanced edge cloud. *IEEE Access* **2019**, *7*, 95970–95977. [CrossRef]
32. Liu, J.; Wang, S.; Wang, J.; Liu, C.; Yan, Y. A Task Oriented Computation Offloading Algorithm for Intelligent Vehicle Network With Mobile Edge Computing. *IEEE Access* **2019**, *7*, 180491–180502. [CrossRef]
33. Wang, Y.; Tao, X.; Zhang, X.; Zhang, P.; Hou, Y.T. Cooperative Task Offloading in Three-Tier Mobile Computing Networks: An ADMM Framework. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2763–2776. [CrossRef]
34. Liu, Z.; Wang, K.; Li, K.; Zhou, M.T.; Yang, Y. Parallel Scheduling of Multiple Tasks in Heterogeneous Fog Networks. In Proceedings of the 25th Asia-Pacific Conference on Communications (APCC), Ho Chi Minh City, Vietnam, 6–8 November 2019.
35. Shan, F.; Luo, J.; Jin, J.; Wu, W. Offloading Delay Constrained Transparent Computing Tasks with Energy-Efficient Transmission Power Scheduling in Wireless IoT Environment. *IEEE Internet Things J.* **2018**, *6*, 4411–4422. [CrossRef]
36. Mazouzi, H.; Boussetta, K.; Achir, N. Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: A theoretical and an experimental study. *Comput. Commun.* **2019**, *144*, 132–148. [CrossRef]
37. Pham, Q.V.; Le, L.B.; Chung, S.H.; Hwang, W.J. Mobile Edge Computing with Wireless Backhaul: Joint Task Offloading and Resource Allocation. *IEEE Access* **2019**, *7*, 16444–16459. [CrossRef]
38. Liu, P.; Li, J.; Sun, Z. Matching-Based Task Offloading for Vehicular Edge Computing. *IEEE Access* **2019**, *7*, 27628–27640. [CrossRef]
39. Li, L.; Zhou, H.; Xiong, S; Yang, J; Mao, Y. Compound Model of Task Arrivals and Load-Aware Offloading for Vehicular Mobile Edge Computing Networks. *IEEE Access* **2019**, *7*, 26631-26640. [CrossRef]
40. Tucker, A. A note on convergence of the Ford-Fulkerson flow algorithm. *Math. Oper. Res.* **1977**, *2*, 143–144. [CrossRef]
41. Yu, G. Tasks Offloading Strategy with Caching Mechanism in Mobile Margin Computing. *Comput. Appl. Softw.* **2019**, *36*, 114–119.
42. Wang, Y.; Sheng, M.; Ye, Q.; Zhang, S.; Zhuang, W.; Li, J. Optimal Dynamic Multi-Resource Management in Earth Observation Oriented Space Information Networks. *arXiv* **2019**, arXiv:1907.12717.
43. Guo, K.; Sheng, M.; Quek, T.Q.; Qiu, Z. Task offloading and scheduling in fog RAN: A parallel communication and computation perspective. *IEEE Wirel. Commun. Lett.* **2019**, *9*, 215–218. [CrossRef]
44. Vallado, D.A.; Crawford, P.; Hujsak, R.; Kelso, T. Revisiting spacetrack report# 3. In Proceedings of the AIAA Astrodynamics Specialist Conference, Keystone, CO, USA, 21–24 August 2006; Volume 6753, p. 446.