Green Task Offloading with Integration of Communication and Computation for LEO Satellite Computing Networks

Jinlu Gu[1,2,3], Danpu Liu[1,2,3(✉)], and Zhilong Zhang[1,2,3]

[1] Beijing Laboratory of Advanced Information Network, Beijing, China
{gjl856,dpliu}@bupt.edu.cn
[2] Beijing Key Laboratory of Network System Architecture and Convergence, Beijing, China
[3] Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract. Low Earth Orbit (LEO) satellite computing networks have opened up new possibilities for handling onboard tasks in space, while facing the challenges such as latency, energy consumption, and satellite battery lifespan degradation. In the existing studies, onboard computations are often fully offloaded to ground stations, resulting in large delays and energy inefficiencies. Moreover, the impact of satellite battery discharge depth on network lifespan has been neglected. To address these issues, this paper proposes a novel approach to offload tasks among satellites, which enables in-space processing and collaborative computation. To effectively extend the overall lifespan of the satellite network, we formulate an optimization problem which aims to maximize the weighted sum utility of energy efficiency and satellite battery lifespan degradation. Furthermore, a green task offloading strategy based on genetic algorithm is proposed to solve the problem. Simulation results demonstrate its benefits in reducing both satellites'energy consumption and discharge depth. Compared to the baseline, the proposed strategy prolongs the lifespan of the satellite network by 42.2%.

Keywords: LEO satellites · task offloading · energy efficiency · cycle life · genetic algorithm

1 Introduction

The integration of satellite internet and edge servers has given rise to satellite computing networks, where each satellite node possesses computing capabilities, enabling them to handle certain onboard tasks. Satellite computing networks rely on satellites as computing nodes. There are some characteristics such as fast motion, wide coverage, and highly dynamic network topology. However, this integration also brings many challenges. For example, satellites rely on batteries as their power source during periods of no sunlight. Once the battery life is depleted, the satellite must be re-launched, resulting in high costs and expenses.

F. Gao et al. (Eds.): Chinacom 2023, LNICST 590, pp. 547–561, 2024.
https://doi.org/10.1007/978-3-031-67162-3_34

In existing studies, the research scenarios can be broadly categorized into two situations: offloading ground-generated tasks for computation on satellites, and offloading satellite-generated tasks for computation at ground stations. Regarding the ground-generated tasks, data is generated on the ground, and the satellite network accepts computing tasks from ground devices. For example, an offloading strategy was developed for tasks generated in IoT ground devices in [1], where the allocation of communication and computing resources were optimized to reduce latency and satellite's power consumption.

Ground tasks can also be offloaded to cloud servers, as discussed in [2], aiming to decrease energy consumption for ground users. Furthermore, in [3], the authors extended the previous architecture to enable individual satellites to offload tasks to up to four other satellites. Additionally, [4] introduced a joint optimization algorithm that addresses computing resource allocation, radio resource allocation, and offloading decisions in a dual-layer satellite scenario. Existing studies on tasks generated in satellites often involves offloading computations to ground stations, which often leads to significant delays and energy consumption. When optimizing these processes, factors such as energy consumption and latency are typically considered. For example, in [5], authors proposed a novel approach, Task scheduling and Resource allocation scheme with Data-driven Bandit Learning (TRDBL), to address the joint task scheduling and resource allocation problem. The problem of minimizing total system energy consumption in [6] was solved using the K-shortest path algorithm and knapsack algorithm. In [7], author achieved energy savings of up to 18% by intelligently selecting between edge and cloud computing for tasks. Lastly, a distributed virtual network function (VNF) placement (D-VNFP) algorithm was proposed to address the problem of minimizing network bandwidth cost and service end-to-end delay in [8].

Typically, the power supply of a satellite relies on two components: solar panels and batteries [9]. The satellite can be powered by the solar panels when exposed to sunlight, and relies on the batteries when there is no sunlight available. As the charge and discharge cycles of the battery cells are limited, unrestrained energy consumption on the satellite can accelerate battery aging and increase the probability of malfunctions. The lifespan of a satellite battery is affected by its depth of discharge (DoD) and discharge rate. The higher the DoD value per cycle, the shorter the lifespan of the satellite battery under the same energy consumption. This means that in order to avoid the prematrue depletion of some satellite's battery and extend the overall lifespan of the satellite network, more data packets should be forwarded to the satellites with higher battery capacities. However, the impact of satellite battery discharge depth was neglected in the literature. Solely focusing on minimizing energy consumption in most studies may result in excessive discharge depth for individual or multiple satellites, leading to a decrease in the overall lifespan of the satellite network, which is not desirable.

To address the aforementioned issues, this paper takes a combined approach by considering both the energy efficiency of the satellite network and the degradation of satellite battery lifespan when formulating the problem. Moreover, this

paper presents a green task offloading strategy based on genetic algorithm (GA), which effectively extends the lifespan of the satellite network.

2 System Model

We assume a LEO network consisting of N satellites, where randomly selected satellites generate computing tasks with variable size. Partial offloading is considered, i.e. each satellite is equipped with a Mobile Edge Computing (MEC) server, and can both generate and receive tasks. As shown in Fig. 1, the orange and gray icons stand for the satellite generating and receiving task, respectively, while the satellite with no task is marked in white. The arrow represents the direction of task offloading. For satellite n, let ω_n^n represent the proportion of tasks left on this satellite, and ω_n^i represent the proportion of tasks offloaded to satellite i. The task allocation matrix is represented as

$$\mathbf{W(t)} = \begin{bmatrix} \omega_1^1 & \omega_2^1 & \cdots & \omega_N^1 \\ \omega_1^2 & \omega_2^2 & \cdots & \omega_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \omega_1^N & \omega_2^N & \cdots & \omega_N^N \end{bmatrix} \tag{1}$$

where $\omega_n^i \in [0, 1]$. We define a task indicator variable b_n and $b_n \in \{0, 1\}$. When there is a task generated on satellite n, $b_n = 1$, and $\omega_n^1 + \omega_n^2 + \cdots + \omega_n^N = 1$. When there are no tasks generated on satellite n, $b_n = 0$, and $\omega_n^1 + \omega_n^2 + \cdots + \omega_n^N = 0$, $n \in \{1, 2, \cdots, N\}$.

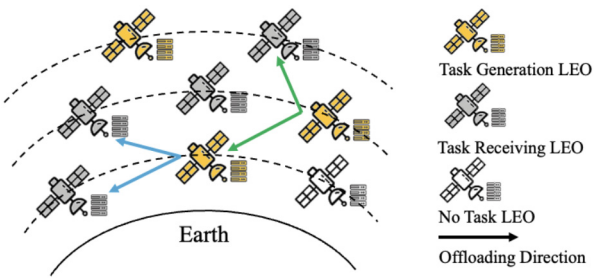

Fig. 1. System model. (Color figure online)

2.1 Delay Model

The tasks generated on satellite n may include both local computing part and the part that is offloaded to other satellites for computing. The latency of these two parts, which have different compositions, will be modeled separately.

A. Local Computing. When performing computations on local satellites, the task incurs only computing delay.

Computing delay: Let $T_{n,n}^{comp}$ denote the time required for local computing of tasks which can be expressed as

$$T_{n,n}^{comp} = \frac{F_n \cdot \omega_n^n \cdot X}{f_n} \tag{2}$$

where F_n represents the amount of tasks generated by satellite n, X is the number of required CPU cycles when computing one bit, and f_n denotes the computation capability (CPU cycles/s) of satellite n. f_n can be expressed as

$$f_n = h_n \cdot f_{hp} + (1 - h_n) \cdot f_{es} \tag{3}$$

where $h_n \in \{0, 1\}$. For satellite n, when using the high-performance mode ($h_n = 1$), it exhibits higher computational efficiency but consumes more energy accordingly. In this mode, the computation capability of satellite n (f_n) is determined by f_{hp}. When satellite n operates in the energy-saving mode ($h_n = 0$), it has lower computational efficiency but is more energy-efficient. In this mode, f_n is equal to f_{es}.

B. Task Offloading. When tasks are offloaded, delay typically consists of three components: transmission delay, propagation delay, and computation delay.

1) Transmission delay: The time required for data transmission from satellite n to m is denoted as $T_{n,m}^{trans}$ and can be expressed as

$$T_{n,m}^{trans} = \frac{F_n \cdot \omega_n^m}{R_{n,m}}. \tag{4}$$

According to the Shannon's theorem, the transmission rate between satellite n and satellite m can be expressed as

$$R_{n,m} = B_{n,m} \log_2 \left(1 + \frac{P_s G_n^T G_m^R L_{n,m}}{B_{n,m} N_0}\right) \tag{5}$$

where P_s is the transmission power of the satellite, G_n^T and G_m^R are the gains of the satellite's transmitting and receiving antennas, respectively. $L_{n,m}$ is the free space path loss between satellite n and m, which can be expressed as

$$L_{n,m} = \left(\frac{c}{4\pi \cdot d_{n,m} \cdot f_c}\right)^2 \tag{6}$$

where c is the speed of light, $d_{n,m}$ is the distance between satellite n and m, f_c is the communications carrier frequency. In space, we can assume only the line-of-sight (LoS) links exist. Typically, only free space path loss needs to be considered, and other losses can be ignored. $B_{n,m}$ represents the available bandwidth for satellite n to offload to m, and N_0 is the power spectral density of noise.

2) Propagation delay: The time it takes for data to propagate in space after being transmitted from satellite n is represented by $T_{n,m}^{prop}$ and can be expressed as

$$T_{n,m}^{prop} = \frac{d_{n,m}}{c}. \tag{7}$$

3) Computing delay: The processing time of data by satellite m is represented by $T_{n,m}^{comp}$ and can be defined as

$$T_{n,m}^{comp} = \frac{F_n \cdot \omega_n^m \cdot X}{f_m}. \tag{8}$$

Let $T_{n,m}^{sum}$ denote the total time consumed during the entire task offloading process of satellite n, which can be expressed as

$$T_{n,m}^{sum} = T_{n,m}^{trans} + T_{n,m}^{prop} + T_{n,m}^{comp}. \tag{9}$$

After determining the delay of the above two parts, the total delay required for a task on satellite n from generation to final processing can be obtained as

$$T_{sum} = \max\left(T_{n,m}^{sum}, T_{n,n}^{comp}\right), m = 1, 2 \cdots N, m \neq n. \tag{10}$$

2.2 Energy Consumption Model

The energy consumption of a satellite n resulting from task offloading and execution can be divided into three parts: static energy consumption, communication energy consumption, and computation energy consumption. The tasks of satellite n may include both local computation and offloading to other satellites for computation, and the energy consumption of the two parts may be different. The local computation part does not involve energy consumption on communication. The energy consumption is modeled as follows.

A. Static Energy Consumption. The energy consumption of a satellite without any load is denoted as E_n^{norm} and can be described as

$$E_n^{norm} = P_{norm} \cdot T_{sum} \tag{11}$$

where P_{norm} represents satellite static power consumption.

B. Communication Energy Consumption The energy consumed by satellites during data transmission is denoted as E_n^{comm} and can be formulated as

$$E_n^{comm} = P_s \cdot \sum_{m=1, m \neq n}^{N} T_{n,m}^{trans}. \tag{12}$$

C. Computation Energy Consumption The energy consumed during satellite data processing is denoted as E_n^{comp} and can be described as

$$E_n^{comp} = k \cdot X \cdot \left(\sum_{i=1}^{N} \omega_i^n F_i \right) \cdot (f_n)^2 \tag{13}$$

where k is the effective switching capacitance coefficient for satellite processors.

Let E_n^{sum} represent the total energy consumption which is the sum of the above three parts, which can be expressed as

$$E_n^{sum} = E_n^{norm} + E_n^{comm} + E_n^{comp}. \tag{14}$$

2.3 Discharge Depth Model

For the commonly used Li-ion batteries on satellites, the cycle life is influenced by multiple factors, including DOD, discharge rate, temperature, etc. The greater the DOD of a battery, the higher the cycle life degradation. For example, when comparing a battery discharged from 20% to 40% and another discharged from 60% to 80%, both provide the same energy. However, the latter causes greater degradation in cycle life. The following is a mathematical model of the relationship between cycle life and depth of discharge, and further a cycle life consumption model is provided.

A. The DOD of the n-th Satellite's Battery at a Certain Moment *t* is Defined as Follows

$$D_n(t) = \frac{C_n^{max} - C_n(t)}{C_n^{max}} \tag{15}$$

where C_n^{max} is the maximum power of the satellite battery, and $C_n(t)$ represents the battery capacity at time t. Moreover, $C_n(t_2) = C_n(t_1) - E_{n,t_1t_2}^{sum}$, where $E_{n,t_1t_2}^{sum}$ represents the total energy consumption of satellite n from t_1 to t_2.

B. Cycle Life Consumption Rate Function. We can get the cycle life consumption rate function from [10]:

$$f(D) = 10^{A(D-1)}(1 + A\ln 10 \cdot D) \tag{16}$$

where D is the current DOD of the battery. A is constant, depending on the type of the battery.

C. Cycle Life Consumption. Let $\frac{L_n^{t_1t_2}}{\hat{L}_n^{t_1}}$ denote the normalized cycle life consumption of the battery from t_1 to t_2, where $\hat{L}_n^{t_1} = \int_{D(t_1)}^{1} f(D)\mathrm{d}D$, and $L_n^{t_1t_2} = \int_{D(t_1)}^{D(t_2)} f(D)\mathrm{d}D$ is the cycle life consumption from t_1 to t_2.

3 Problem Formulation and Solution

3.1 Problem Formulation

Based on the energy consumption model and cycle life consumption model mentioned above, the utility function of satellite systems can be defined as

$$U_{total}^{t_1 t_2} = \left(\beta_1 \cdot \frac{\sum_{n \in \mathcal{V}} F_n}{\sum_{n \in \mathcal{V}} E_{n,t_1 t_2}^{sum}} - \beta_2 \cdot \sum_{n \in \mathcal{V}} \beta \cdot \frac{L_n^{t_1 t_2}}{\hat{L}_n^{t_1}} \right). \tag{17}$$

The utility function is defined in this way to take both the overall energy efficiency and the cycle life consumption of the battery from t_1 to t_2 into account. The purpose of this design is to achieve a balance during task processing. Therefore, we can ensure high energy efficiency of the system while minimize the consumption of the battery's cycle life.

The first part of the utility function represents the overall energy efficiency, which is the ratio of the total amount of tasks to the total energy consumed on processing all tasks. This ratio indicates how effectively the system utilizes energy during task completion. A higher overall energy efficiency means that the system can utilize energy more efficiently during task processing, allowing for the completion of more tasks given a certain energy resource.

The second part of the utility function represents the cycle life consumption of the battery. The cycle life consumption measures the impact of energy consumption during task completion on the battery's lifespan. By considering cycle life consumption, the utility function can assess the degree of battery lifespan degradation during task processing.

By combining overall energy efficiency and cycle life consumption, the utility function allows for the adjustment of the weighting coefficients β_1 and β_2 to balance the importance of both factors, and β is a scale factor. A higher value of β_1 indicates a greater emphasis on energy efficiency, while a smaller β_1 focuses more on battery lifespan consumption. This design enables the system's behavior to be flexibly adjusted according to specific requirements and priorities, aiming to achieve optimal performance and resource utilization efficiency.

A green task offloading problem to optimize the utility at a given snapshot is then formulated as follows

$$\max_{\mathbf{W(t)},\mathbf{H(t)}} \quad U_{total}^{t_1 t_2} \tag{18}$$

$$\text{s.t.} \quad \left(\sum_{i=1}^{N} \omega_i^n F_i \right) \cdot X \leq Z_n, \qquad \forall v_n \in V, \tag{19}$$

$$C_n(t_2) \geq 0, \qquad \forall v_n \in V, \tag{20}$$

$$T_{sum} \leq T_{th}, \qquad \forall v_n \in V, \tag{21}$$

where $V = \{v_1, v_2, \ldots, v_n\}$. $\mathbf{W(t)}$ represents the task allocation matrix defined in section II, and $\mathbf{H(t)} = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix}$ represents the matrix of computation modes for each satellite.

Constraint (19) ensures that the computational load assigned to each satellite does not exceed a specified value. Constraint (20) ensures that the remaining energy of each satellite cannot be negative. Constraint (21) represents the total delay of offloading tasks, which should not exceed a specified value. This optimization problem is solved once per snapshot.

3.2 Genetic Algorithm-Based Solution

The proposed satellite task offloading problem is a mixed-integer programming problem, with decision variables comprising of two parts. One part is the task assignment matrix $\mathbf{W(t)}$, where each variable is continuous. The other part is the computation mode selection matrix $\mathbf{H(t)}$ for the satellite, where each variable is discrete, including two cases of 0 and 1. As a result, we turn to a heuristic algorithm based on the genetic algorithm to address this optimization problem.

The GA-based green task offloading strategy solution is specified in Algorithm 1. The initial population is generated by randomly selecting N_P feasible solutions, each consisting of two parts: an offloading policy matrix and a satellite computing mode selection matrix. Some individuals with higher fitness are selected as elites from the current population, which are then passed down to the next generation. The elite children ensures the quality of the population in each iteration. For each generation, biologically-inspired heuristic operators, such as selection, crossover, and mutation, are used to search for high-quality solutions. Individuals with higher fitness values are selected to replace the current population and form the next generation. This algorithm process is iterated until the maximum number of generations is reached. Finally the best individual obtained in the end serves as the solution to the aforementioned problem, i.e., the offload strategy and satellite computing mode selection for this snapshot.

We then analyze the complexity of the proposed genetic algorithm for task offloading. We consider several factors, including the size of the population (N_P), the maximum number of generations (N_G), the number of elites (N_E), the crossover rate ($\alpha 1$), and the mutation rate ($\alpha 2$).

Time Complexity:

Generating random initial individuals: The time complexity of generating N_P feasible individuals randomly is $\mathcal{O}(N_P)$.

Computing and storing fitness values: Evaluating the fitness function for the N_P individuals requires $\mathcal{O}(N_P)$ time complexity.

Iterating over N_G generations: a. Selecting elite individuals: Selecting the top N_E individuals from the population has a time complexity of $\mathcal{O}(N_E)$. b. Applying crossover and mutation operators: The time complexity for applying crossover and mutation depends on the length of the encoding and can be represented as $\mathcal{O}(N_P{*}M)$, where M is the encoding length. c. Updating the population: The time complexity for updating the population is $\mathcal{O}(N_P)$. d. Computing and storing fitness values: Evaluating the fitness function for the updated population takes $\mathcal{O}(N_P)$ time complexity.

Therefore, the overall time complexity for N_G generations can be expressed as $\mathcal{O}(N_G{*}(N_P + N_P{*}M + N_P))$, which simplifies to $\mathcal{O}(N_G{*}N_P{*}M)$.

Space Complexity:

Population storage: Storing N_P individuals, each with an encoding length of M, requires $\mathcal{O}(N_P{*}M)$ space complexity.

Temporary storage: The crossover and mutation operations may require temporary storage, which also has a space complexity of $\mathcal{O}(N_P{*}M)$.

Thus, the overall space complexity of the genetic algorithm is $\mathcal{O}(N_P{*}M)$.

Algorithm 1: GA-based offloading algorithm

Input: size of population N_P, maximum number of generations N_G, number of elites N_E, rate of crossover $\alpha 1$, rate of mutation $\alpha 2$

Output: task allocation matrix **W(t)**, computation modes matrix **H(t)**

generate N_P feasible individuals randomly as parents *pop*;
evaluate fitness function and store fitness values;
for $i \leftarrow$ *1 to* N_G **do**
    select the best N_E individuals in *pop* as elites *pop*1;
    create new individuals by applying crossover and mutation operators;
    save the two individuals in children as *pop*2;
    update $pop \leftarrow pop1 + pop2$;
    evaluate fitness function and store fitness values;
end
return *the best individual*

4 Simulation Results

Simulation results are presented to assess the performance of the proposed heuristic strategy, named "GA Offloading," which combines task offloading scheduling and satellite computation mode selection. This method was compared with three baseline approaches: 1. No Offloading: The satellite does not offload tasks and performs computations on the local satellite. 2. Random Offloading: The satellite randomly offloads tasks generated by satellites. 3. Greedy Offloading [11]: The satellite first offloads tasks to satellite 1 until satellite 1 reaches its maximum computing capacity. Afterward, tasks are sequentially offloaded to the remaining satellites.

We consider a small part of a satellite network consisting of six LEO satellites. The workload generated by the satellites is primarily concentrated on two of them. The task generation of the satellites follows a Poisson distribution. During the task processing, the satellite remains in a condition without sunlight exposure, which means it is powered solely by batteries. The initial battery charge of the satellites is assumed to be at full capacity unless otherwise specified. The simulation parameter setting is shown in Table 1 [12]. Here, we consider energy efficiency and battery lifespan degradation to be equally important, thus setting $\beta_1 = \beta_2 = 1$.

Table 1. Simulation Parameters

Parameter	Value
X	100
f_{hp}	10 Gcycles/s
f_{es}	5 Gcycles/s
$B_{n,m}$	100 MHz
P_{norm}	30 W
P_s	50 W
G_n^T	27 dBi
G_m^R	24 dBi
N_0	-174 dBm/Hz
c	3×10^8 m/s
f_c	28 GHz
k	10^{-25}
A	0.8
C_n^{max}	180000 J
β	1000
β_1, β_2	1, 1

Figure 2 presents the energy consumption of each satellite after processing 100 consecutive tasks. The simulation results reveal that energy consumption increases with the number of task executions. The GA Offloading method consistently exhibits lower energy consumption compared to the other three methods, with a performance improvement of approximately 4.9%.

As shown in Fig. 3, the cycle life consumption of each satellite is depicted after undergoing continuous processing of 100 tasks. Here, we set $\beta = 1$, and the task intensities for satellites 1 and 4 are in the magnitude of 10^6 bits, while the remaining satellites have task intensities in the magnitude of 10^4 bits. In the case of No Offloading, since no offloading is performed, the cycle life consumption is mainly concentrated on satellites 1 and 4. For Greedy Offloading, the cycle life consumption is primarily focused on satellites 1 and 2, as tasks are preferentially offloaded to satellite 1 and then follow a sequential order. Random Offloading, being completely random, results in relatively unpredictable cycle life consumption. The proposed method, GA Offloading, ensures that the cycle life consumption of each satellite remains at a lower level, effectively extending the lifespan of the satellite network.

In Fig. 4, we can observe the connection between cycle life consumption and the count of task executions for various offloading methods. As the task execution count increases, the curves of each method show an upward trend. This

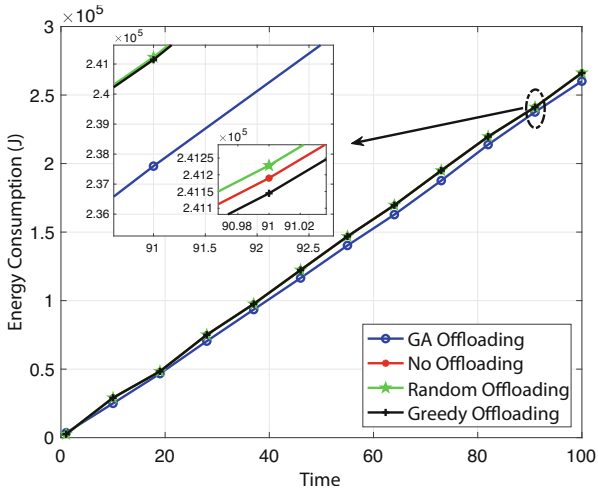

Fig. 2. Energy consumption for different offloading methods across various satellites.

is due to the fact that the discharge depth of the battery gradually increases, resulting in greater cycle life consumption. From the simulation results, it can be observed that GA Offloading has the slowest rate of increase, outperforming Random Offloading, No Offloading, and Greedy Offloading. In a word, our proposed algorithm provides an offloading method that exhibits slower battery aging. After 100 executions, GA Offloading demonstrates a 42.2% reduction in cycle life consumption compared to Random Offloading.

After the continuous processing of 100 tasks, Fig. 5 showcases how total cycle life consumption varies with task intensity for different offloading methods. We set the task intensity for satellites 1 and 4 to range from the magnitude of 10^5 bits to 10^6 bits, while the task intensity for the remaining four satellites is set to the magnitude of 10^4 bits. From the simulation results, it can be observed that the cycle life consumption of the satellites increases with higher task intensities. This is because higher task intensities result in greater energy consumption and, consequently, higher power consumption. GA Offloading exhibits the slowest rate of increase, outperforming the other three methods. When the input task size is on the order of 10^6 bits, the curve of the GA Offloading method exhibits a 25.1% reduction in total cycle life consumption compared to Random Offloading.

Figure 6 illustrates the relationship between the utility function values and the number of task executions for different offloading methods. As the number of task executions increases, each method exhibits a declining trend. This is due to the marginal changes in energy efficiency for each method, while cycle life consumption continues to increase, resulting in an overall decrease in utility. From the simulation results, it can be observed that GA Offloading shows the slowest decline rate, outperforming Random Offloading, No Offloading, and Greedy Offloading. Overall, our proposed algorithm achieves an offloading method with high utility value. After 100 executions, GA Offloading method shows a utility improvement of 7.8% compared to Random Offloading.

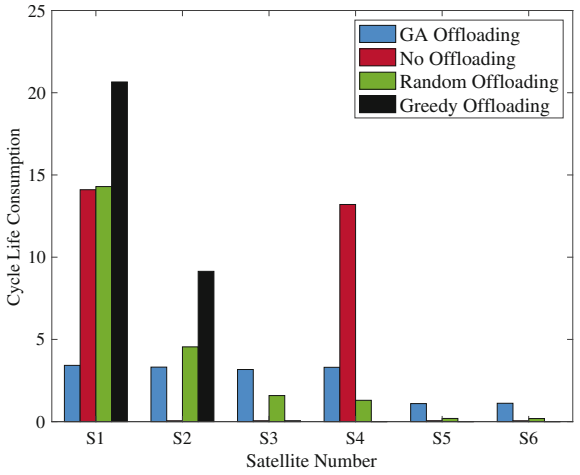

Fig. 3. Cycle life consumption for different offloading methods across various satellites.

The relationship between the utility function values of various offloading methods and the initial battery discharge depth is depicted in Fig. 7. The figure illustrates the relationship between the utility values of continuous processing of 100 tasks and initial discharge depths of 0, 0.1, 0.2, 0.3, 0.4, and 0.5. From the simulation results, it can be observed that as the initial discharge depth increases, the utility values gradually decrease. This is because when the computational workload remains the same, the same amount of energy is consumed. However, if the initial discharge depth is larger, the battery's life degradation will

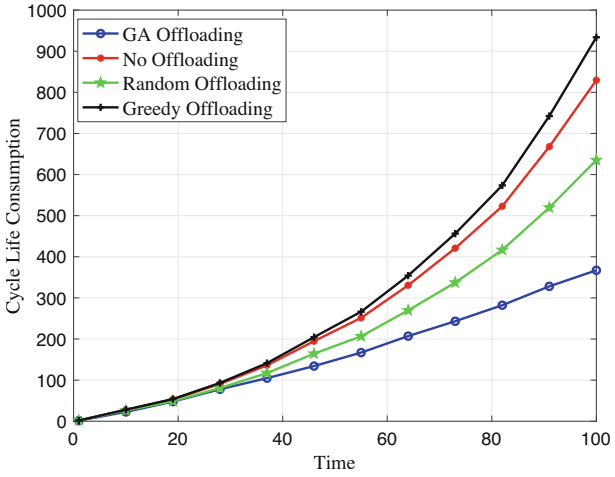

Fig. 4. Variation of cycle life consumption with the number of task executions for different offloading methods.

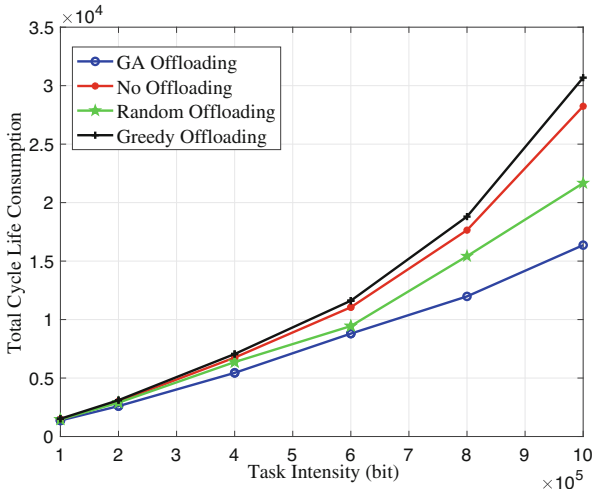

Fig. 5. Variation of total cycle life consumption with the task intensity for different offloading methods.

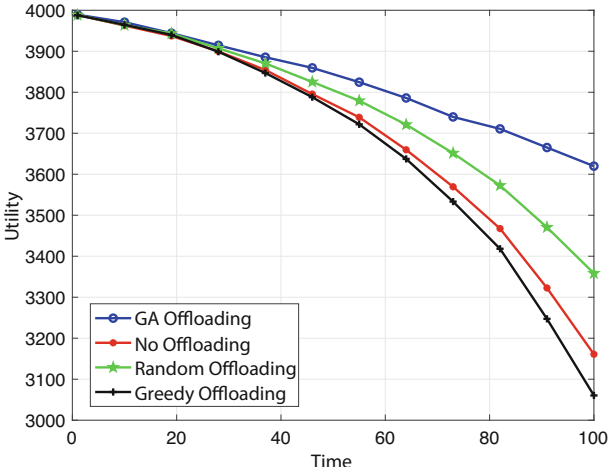

Fig. 6. Variation of utility with the number of task executions for different offloading methods.

be greater, ultimately resulting in lower utility values. GA Offloading exhibits the slowest rate of decline, outperforming the other three methods. When the initial discharge depth is set to 0.5, GA Offloading method demonstrates a utility improvement of 76.4% compared to Random Offloading.

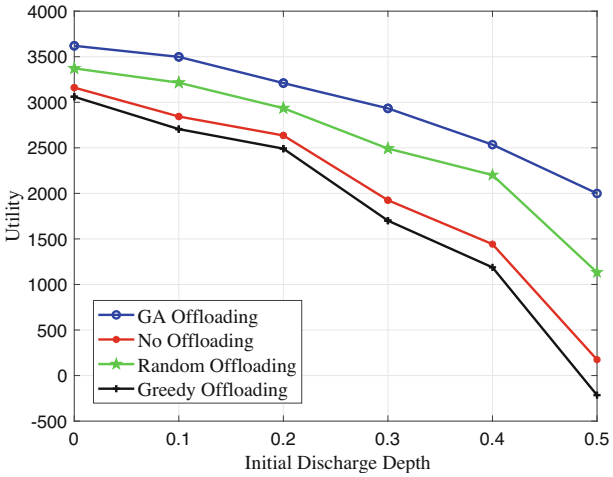

Fig. 7. Variation of utility with the initial discharge depth for different offloading methods.

5 Conclusion

In this study, we conducted research on task offloading in satellite networks. We proposed a GA-based offloading method and compared it with other existing methods. Through simulation experiments and performance analysis, we have drawn the following conclusions:

Firstly, our experimental results demonstrate that the proposed GA Offloading method outperforms other methods in terms of cycle life consumption and utility function values. It effectively reduces the battery's discharge depth during the offloading decision-making process, thereby extending the lifespan of the satellite network.

Furthermore, the research findings reveal the impact of task intensity, initial discharge depth, and other factors on the performance of offloading methods. By further analyzing and optimizing these influencing factors, it is possible to enhance the performance and energy utilization efficiency of satellite networks.

Acknowledgment. This work is supported in part by the Beijing Natural Science Foundation under Grant L202003, the National Natural Science Foundation of China under Grant 61971069 and 62271065, and the Open Research Project of the State Key Laboratory of Media Convergence and Communication, Communication University of China (SKLMCC2021KF009).

References

1. Cui, G., Li, X., Xu, L., Wang, W.: Latency and energy optimization for MEC enhanced sat-IoT networks. IEEE Access **8**, 55915–55926 (2020)
2. Tang, Q., Fei, Z., Li, B., Han, Z.: Computation offloading in LEO satellite networks with hybrid cloud and edge computing. IEEE Internet Things J. **8**(11), 9164–9176 (2021)
3. Wang, Y., Zhang, J., Zhang, X., Wang, P., Liu, L.: A computation offloading strategy in satellite terrestrial networks with double edge computing. In: 2018 IEEE International Conference on Communication Systems (ICCS), pp. 450–455. IEEE (2018)
4. Fang, H., Jia, Y., Wang, Y., Zhao, Y., Gao, Y., Yang, X.: Matching game based task offloading and resource allocation algorithm for satellite edge computing networks. In: 2022 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–5. IEEE (2022)
5. Gao, X., Wang, J., Huang, X., Leng, Q., Shao, Z., Yang, Y.: Energy-constrained online scheduling for satellite-terrestrial integrated networks. IEEE Transactions on Mobile Computing **22**(4), 2163–2176 (2021)
6. Wang, Y., Li, J., Chen, M., Chai, R.: Joint route selection and time-slot allocation for energy consumption optimization in satellite communication systems. In: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), pp. 1–5. IEEE (2021)
7. Gost, M.M., Leyva-Mayorga, I., Pérez-Neira, A., Vázquez, M.Á., Soret, B., Moretti, M.: Edge computing and communication for energy-efficient earth surveillance with leo satellites. In: 2022 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 556–561. IEEE (2022)
8. Gao, X., Liu, R., Kaushik, A., Zhang, H.: Dynamic resource allocation for virtual network function placement in satellite edge clouds. IEEE Trans. Netw. Sci. Eng. **9**(4), 2252–2265 (2022)
9. Foust, J.: Spacex's space-internet woes: despite technical glitches, the company plans to launch the first of nearly 12,000 satellites in 2019. IEEE Spectr. **56**(1), 50–51 (2018)
10. Yang, Y., Xu, M., Wang, D., Wang, Y.: Towards energy-efficient routing in satellite networks. IEEE J. Sel. Areas Commun. **34**(12), 3869–3886 (2016)
11. Cheng, L., Feng, G., Sun, Y., Liu, M., Qin, S.: Dynamic computation offloading in satellite edge computing. In: ICC 2022-IEEE International Conference on Communications, pp. 4721–4726. IEEE (2022)
12. Zhou, D., Sheng, M., Wang, X., Xu, C., Liu, R., Li, J.: Mission aware contact plan design in resource-limited small satellite networks. IEEE Trans. Commun. **65**(6), 2451–2466 (2017)