Article

Research on Routing Equalization Algorithm of Inter-Satellite Partition for Low-Orbit Micro-Satellites

Hengfei Cheng [1], Zhaobin Xu [2,*], Xiaoxu Guo [1], Jia Yang [1], Kedi Xu [1], Shuqin Liu [1], Zhonghe Jin [2] and Xiaojun Jin [2]

[1] Micro-Satellite Research Center, Zhejiang University, Hangzhou 310007, China; 12124019@zju.edu.cn (H.C.); guoxx@zju.edu.cn (X.G.); 12124068@zju.edu.cn (J.Y.); 22124025@zju.edu.cn (K.X.); 22024081@zju.edu.cn (S.L.)
[2] Micro-Satellite Research Center, Zhejiang Micro-Satellite Research Laboratory, Zhejiang University, Hangzhou 310007, China; jinzh@zju.edu.cn (Z.J.); axemaster@zju.edu.cn (X.J.)
* Correspondence: zjuxzb@zju.edu.cn

Abstract: Low-orbit micro-satellite technology has developed rapidly in recent years due to its advantages of low time delay, low cost and short research period. However, among the existing inter-satellite routing algorithms, the classical flooding and greedy algorithms and their derivatives also have some limitations. The path delay calculated by the flooding algorithm is small but the calculation is large, while the greedy algorithm is the opposite. In this paper, a balanced inter-satellite routing algorithm based on partition routing is proposed. This paper presents the simulation experiments for the following indexes of the classic inter-satellite routing algorithms and the balanced partition routing algorithm: computation complexity, single-node computation pressure, routing path delay, path delay variance (data in Topo table satisfy $\mu = 5, \sigma^2 = 10$). The results reveal that the balanced partition routing algorithm achieves better performance. In this paper, two optimization directions of the balanced partition routing algorithm are simulated under conditions that the data in the Topo table satisfy $\mu = 5, \sigma^2 = 6, \sigma^2 = 10$ and $\sigma^2 = 15$, respectively, when comparing their performance indicators. The experiments show that these two optimization methods can be adapted to various application scenarios and can further reduce the hardware cost of satellite nodes.

Keywords: low orbit; micro-satellite technology; star chain; routing algorithm; partitioning algorithm

check for updates

Citation: Cheng, H.; Xu, Z.; Guo, X.; Yang, J.; Xu, K.; Liu, S.; Jin, Z.; Jin, X. Research on Routing Equalization Algorithm of Inter-Satellite Partition for Low-Orbit Micro-Satellites. *Future Internet* **2022**, *14*, 207. https://doi.org/10.3390/fi14070207

Academic Editor: Min-Shiang Hwang

Received: 24 May 2022
Accepted: 28 June 2022
Published: 4 July 2022

1. Introduction

Due to the advantage of large coverage and little limitation of ground geographical conditions, a micro-satellite network has become an effective means to provide communication service in areas where ground communication facilities are difficult to cover. In recent years, many countries and companies have put forward plans to build a new generation of micro-satellite networks, setting off a boom in satellite network research [1]. For instance, the commercial rocket company SpaceX launched a star-link project to provide Internet service to remote areas [2]. What is more, Tsinghua University has proposed the plan of GRID (Gammy Ray Integrated Detectors) to explore astrophysics.

Different from the ground network, the constant motion of satellites will inevitably bring about frequent switching of links between satellites. These switches result in the constant changes in the topology of the whole satellite network, and the routing information needs to be updated accordingly. Therefore, routing algorithms have a great influence on the efficiency of inter-satellite communication and have become one of the hot topics [3,4]. In satellite networks, the transmission cost of a path is mainly determined by propagation delay and waiting delay. Compared with the ground network, the distance between satellite nodes is longer, so the propagation delay occupies a larger proportion of the total delay. This paper focuses on the end-to-end shortest delay routing algorithm.

At present, mainstream inter-satellite routing algorithms still have their limitations. Current real-time dynamic routing algorithms mainly include flooding and backtracking algorithm and the greedy algorithm. The principle of the flooding and backtracking algorithm is to traverse every inter-satellite node and compare the length of each path to find the shortest path [5]. This algorithm needs to backtrack to find out which one of these paths goes through fewer inter-satellite nodes or if more than two paths have the same path length. Hence, its advantage is that the routing path calculated must be the optimal path. On the other hand, the algorithm has too much calculation and poor scalability. The routing strategy of the greedy algorithm is to find the nearest node under the condition of a local optimal path. The greedy algorithm has a small amount of calculation while the path calculated is usually not the globally optimal path. Both the flooding algorithm and the greedy algorithm have major drawbacks. For these two algorithms, the calculation of the routing path must be completed by a single star node at a time. This means that the computational burden cannot be shared by multiple nodes. When the actual system is running, the inter-star traffic cannot be evenly distributed to each star node, and the calculation pressure of some star nodes is too high. However, it is not known in advance which star nodes will have high computational pressure, so redundancy design can only be carried out to ensure reliability. We know that satellites are powered by solar energy to perform all kinds of calculations in space, and more satellites need to install larger solar panels, which will greatly increase the cost of satellite hardware [6–8].

This paper has the following structure: Section 2 presents related work on academic achievements of predecessors and the main contribution of the paper; Section 3 provides a mathematical analysis of the inter-satellite routing algorithm; two optimization directions of the balanced partition routing algorithm are discussed in Section 4; lastly, Section 5 summarizes the work.

2. Related Works

2.1. The Work of Predecessors

In this field, many experts and scholars have carried out research for many years. Weng Yao, from the Software Institute of the Chinese Academy of Sciences, proposed an on-demand partial topology routing algorithm in LEO satellite networks. The algorithm guarantees the minimum delay and better convergence time of the constructed route, so it is very suitable for those services requiring a higher delay in low orbit satellite networks. However, it is still a flooding algorithm in essence, and the cost of routing detection is still very large. The algorithm only needs to send data to the routing detection, and each node does not need the global topological information, so as to avoid a lot of link-state information exchange, minimize the overhead due to routing protocols, and smaller local topology protocol convergence time is the biggest advantage of this routing algorithm [9]. In 2001, Henderson designed a distributed routing algorithm to realize the minimum delay. Henderson's algorithm is based on terrain factors and believes that a continuous local optimization strategy can make the final path tend to be globally optimal [10]. The datagram routing algorithm introduced by Ekici is applicable to polar satellite constellations. Satellite nodes are regarded as a network structure composed of virtual nodes, and data packets are transmitted in a distributed way in a fixed topology structure [11].

Weng's algorithm can be regarded as a variant of the flooding algorithm, while Henderson and Ekici's algorithms are improvements to the greedy algorithm. These algorithms also have the inherent defects of the flooding algorithm and greedy algorithm. Moreover, the above routing algorithm mainly relies on a one-star node to complete the calculation of each planned route. In the actual inter-satellite system, each star is often not a uniformly distributed workload. This will cause some nodes to bear a lot of computational pressure and consume a lot of energy. The principle of maximum redundancy would require each star to be fitted with a payload with more computing power and a larger capacity of solar panels, which would greatly increase the cost of satellite hardware. This is where we would like to further improve the existing inter-satellite routing algorithm. Therefore, we propose a balanced routing algorithm based on the partition routing idea in the original partition routing algorithm.

2.2. Main Algorithm Ideas and Contributions of the Paper

In this paper, the constellation is assumed to consist of N satellite nodes.

$$\text{Topo} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$$

The Topo table represents the direct distance between two star nodes. For instance, a_{11} represents the distance from the No. 1 satellite to the No. 1 satellite, and the distance of course is 0. a_{ij} represents the direct distance from the No. i satellite to the No. j satellite [12,13].

Inspired by the above research, we propose a balanced routing algorithm based on the partition routing idea. In the original partition routing algorithm, as long as the communication nodes between two partitions have problems, the mutual communication will be affected [14] and the balanced partition routing algorithm determines the external communication nodes between partitions in a dynamic way. If the original optimal communication nodes have problems, our algorithm will consider the distance between the original communication nodes to be infinite. Then, the algorithm will immediately update and calculate the new communication node, so that the destruction resistance will be greatly enhanced [15].

In this paper, N star nodes are divided into three groups. Star nodes in group A are numbered 1 through $\frac{N}{3}$. By parity of reasoning, nodes in group B are numbered $\frac{N}{3} + 1$ through $\frac{2N}{3}$ and nodes in group C are numbered $\frac{2N}{3} + 1$ through N. The direct communication distance and the variance of the distance between star nodes in the same group are smaller. a_{ij} approximately satisfies the following relation:

$$\begin{cases} a_{ij} \sim N(\mu_1, \sigma_1^2),\ i,\ j \subseteq \varphi \\ a_{ij} \sim N(\mu_2, \sigma_2^2),\ i \subseteq \varphi,\ j \nsubseteq \varphi \end{cases}$$

In the above formula, $\mu_1 > \mu_2$, $\sigma_1^2 > \sigma_2^2$, φ refers to one group [16]. The algorithm first calculates the two nearest star nodes between each group, and these two star nodes serve as the gateway communication nodes between each group. For instance, the routing path between node No. i and node No. j needs to be calculated. The specific algorithm is as follows:

(1) If node No. i and node No. j belong to the same group, then we only need to address the route within the group. The routing algorithm within the group performs the partial flooding and greedy algorithm at the same time and selects the shorter one as the final routing path.

(2) In the other case where node No. i and node No. j do not belong to the same group, then we need to calculate the two nearest communication nodes between the two groups that node No. i and node No. j belong to. It is assumed that these two star nodes are node No. i + m and node No. j + n. Finally, the algorithm needs to calculate the shortest routing paths from node No. i to node No. i + m and from node No. j + n to node No. j. Therefore, our final routing path is No. i->No. i + m->No. j + n->No. j.

The simulation results reveal that compared with other algorithms, the proposed balanced partition routing algorithm achieves better performance.

In the latter part of this paper, the balanced partition routing algorithm was further explored. We have yet to explore two different optimization directions on the premise of further reducing the computational pressure of a single node. One way is to divide a large group into smaller groups (which can be interpreted as smaller groups belonging to a larger group), and the other way is to divide nodes into more groups directly. Experimental results show that these two optimization methods can further reduce the complexity of the algorithm. What is more, the hardware cost of nodes is greatly reduced. Finally, these two optimization ideas can make the algorithm adapt to various application scenarios and have better flexibility.

3. Algorithmic Mathematical Analysis

This section analyzes the algorithms mentioned above from a mathematical perspective, including the overall computation complexity of the algorithm, the computational pressure of a single node, and the average path delay of the algorithm.

3.1. Algorithm Complexity Analysis

Greedy algorithm: For the source node, it only needs to find the next jump node with the shortest path in each step until it reaches the target node. At most, $\mathrm{N}-2$ nodes are passed through from the source node to the target node, and the average expectations are $\frac{N-2}{2}$ nodes. Every time the information arrives at one node to find the next node, N nodes need to be traversed to calculate the locally optimal path [17].

Its computational complexity is:

$$\mathrm{G(N)} = \frac{(\mathrm{N}-2)\cdot \mathrm{N}^3}{2} \tag{1}$$

Flooding and backtracking algorithm: Its computational complexity is:

$$\mathrm{F(N)} = \sum_{\mathrm{m}=0}^{\mathrm{N}-2} \frac{(\mathrm{N}-2)!}{(\mathrm{N}-2-\mathrm{m})!} \cdot \mathrm{N}^2 + \delta_1 \tag{2}$$

In this formula, $\frac{(\mathrm{N}-2)!}{(\mathrm{N}-2-\mathrm{m})!}$ is the number of satellite nodes to be traversed when the source node passes m forwarding satellites. It can be deduced that finding the routing path from the source node to the destination node takes $\sum_{\mathrm{m}=0}^{\mathrm{N}-2} \frac{(\mathrm{N}-2)!}{(\mathrm{N}-2-\mathrm{m})!}$ calculations. There are N^2 paths to calculate here. δ_1 is a positive number tending to be infinitesimally small, which represents the backtracking part of the flooding algorithm. Only when more than one path with the same distance is founded during traversal, backtracking is performed to compare the number of nodes that pass through and select the path with fewer nodes. However, this situation is very rare and can almost be ignored [18,19].

Partition routing algorithm: For the source node, the destination node has a chance of $\frac{\frac{N}{3}-1}{N-1}$ being in the same group and a chance of $\frac{\frac{2N}{3}}{N-1}$ being outside the group [20].

For stars nodes within the same group, its computational complexity is:

$$\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-\mathrm{m}\right)!} + \delta_2 \tag{3}$$

For stars nodes outside the group, its computational complexity is:

$$2\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-1} \frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-\mathrm{m}\right)!} + \delta_2\right)$$

$$\delta_2 = \delta_{\mathrm{recall}} + \delta_{\mathrm{greed}}$$

$$\delta_{\mathrm{greed}} = \frac{\left(\frac{N}{3}-2\right)\cdot\frac{N}{3}}{2} \tag{4}$$

Both partial flooding algorithms and greedy algorithms are used to calculate the path within the group. δ_{recall} refers to the overhead of the backtracking part of the flooding algorithm. δ_{greed} represents the overhead of the greedy algorithm. Compared with the huge calculation time cost of the flooding algorithm, the time cost of the greedy algorithm and backtracking can almost be ignored [21].

Hence, the average computational complexity of the total algorithm is:

$$P(N) = (\frac{\frac{N}{3}-1}{N-1}\cdot\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!} + \delta_2 + \frac{\frac{2N}{3}}{N-1}\cdot(\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!} + \delta_2)\cdot 2)\cdot N^2 \quad (5)$$

Balanced partition routing algorithm: In contrast to the partition routing algorithm, a balanced partition routing algorithm adds the process of dynamically calculating communication nodes between the groups.

So its computational complexity is:

$$C(N) = (\frac{\frac{N}{3}-1}{N-1}\cdot\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!} + \delta_2 + \frac{\frac{2N}{3}}{N-1}\cdot((\frac{N}{3})^2 + (\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!} + \delta_2)\cdot 2))\cdot N^2 \quad (6)$$

The computational complexity of the balanced partition routing algorithm and flooding algorithm is compared below:

It is easy to derive from the above equation:

$$C(N) < \left(\left(\frac{N}{3}\right)^2 + \sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\cdot 2\right)\cdot N^2 = C_1(N)$$

$$F(N) = \left[(N-2)! + \sum_{m=\frac{5N}{6}-2}^{N-3}\frac{(N-2)!}{(N-2-m)!} + \sum_{m=\frac{2N}{3}-2}^{\frac{5N}{6}-3}\frac{(N-2)!}{(N-2-m)!} + \sum_{m=0}^{\frac{2N}{3}-3}\frac{(N-2)!}{(N-2-m)!}\right]$$

$$F(N) - C_1(N) = [\left((N-2)! - \left(\frac{N}{3}\right)^2\right) + \left(\sum_{m=\frac{2N}{3}-2}^{\frac{5N}{6}-3}\frac{(N-2)!}{(N-2-m)!} - \sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\right)$$

$$+ \left(\sum_{m=\frac{5N}{6}-2}^{N-3}\frac{(N-2)!}{(N-2-m)!} - \sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\right) + \sum_{m=0}^{\frac{2N}{3}-3}\frac{(N-2)!}{(N-2-m)!}] \quad (7)$$

It can be proved that the balanced partition routing algorithm is much lower than the flooding algorithm in terms of algorithm complexity. A balanced partition routing algorithm adds the process of calculating communication nodes between groups dynamically. Therefore its algorithm complexity is greater than that of the partition routing algorithm.

In terms of the partition routing algorithm and the flooding and backtracking algorithm:

$$C(N) > \left(\frac{1}{2}\cdot\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!} + \frac{1}{2}\cdot\left(\left(\frac{N}{3}\right)^2 + \left(\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\right)\cdot 2\right)\right)\cdot N^2 > \frac{3}{2}\cdot\sum_{m=0}^{\frac{N}{6}-1}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\cdot N^2 = C_2(N)$$

$$C_2(N) - G(N) = \frac{3}{2}\left(\left(\frac{N}{3}-2\right)! + \frac{1}{2}\cdot\left(\frac{N}{3}-2\right)! - \left(\frac{N-2}{2}\right) + \sum_{m=0}^{\frac{N}{6}-3}\frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-m\right)!}\right)\cdot N^2 > 0 \quad (8)$$

Therefore, the computation complexity of the partition routing algorithm is higher than the greedy algorithm. In conclusion, the flooding and backtracking algorithm has the highest computational complexity, followed by the balanced partition routing algorithm and the partition routing algorithm, and the greedy algorithm has the lowest computational complexity.

3.2. Minimum Computation Force of Star Node

In actual routing, each node only needs to calculate the path to the next node. For both the flooding and backtracking algorithm and the greedy routing algorithm, a single star node needs to calculate the global route. Compared with the traditional algorithm, the partition routing algorithm can allocate the computation force to two or more nodes. In practice, the hardware requirements for inter-satellite nodes are reduced, and the cost is also reduced objectively [22]. The requirements of the above algorithms on the computing capacity of a single node are shown below:

Greedy algorithm:

$$\mathrm{G_{node}(N)} = \frac{(\mathrm{N}-2)\cdot \mathrm{N}^3}{2} \tag{9}$$

Flooding and backtracking algorithm:

$$\mathrm{F_{node}(N)} = \sum_{\mathrm{m}=0}^{\mathrm{N}-2} \frac{(\mathrm{N}-2)!}{(\mathrm{N}-2-\mathrm{m})!} + \delta_1 \tag{10}$$

Partition routing algorithm:

$$\mathrm{P_{node}(N)} = \sum_{\mathrm{m}=0}^{\frac{N}{3}-2} \frac{\left(\frac{\mathrm{N}}{3}-2\right)!}{\left(\frac{N}{3}-2-\mathrm{m}\right)!} + \frac{\left(\frac{\mathrm{N}}{3}-2\right)\cdot\frac{\mathrm{N}}{3}}{2} + \delta_{\mathrm{recall}} \tag{11}$$

Balanced partition routing algorithm:

$$\mathrm{C_{node}(N)} = \sum_{\mathrm{m}=0}^{\frac{N}{3}-2} \frac{\left(\frac{N}{3}-2\right)!}{\left(\frac{N}{3}-2-\mathrm{m}\right)!} + \left(\frac{N}{3}\right)^2 + \frac{\left(\frac{\mathrm{N}}{3}-2\right)\cdot\frac{\mathrm{N}}{3}}{2} + \delta_{\mathrm{recall}} \tag{12}$$

3.3. Algorithm Validity Analysis

It is assumed that the direct distance a_{ij} between any two star nodes satisfies $a_{ij} \sim N(\mu, \sigma^2)$.

It is easy to know that the flooding and backtracking algorithm compares all possible paths between two star nodes and finds the shortest path among them. If there is more than one shortest path, the path with the least number of inter-star nodes will be selected. Therefore, the flooding and backtracking algorithm is definitely optimal [23].

For greedy algorithms, the average expected route path distance is:

$$G_{path}(N) = \frac{N}{2}\cdot\mu\cdot\epsilon\cdot\frac{1}{\sigma^2} \tag{13}$$

ϵ is an undetermined parameter. For greedy algorithms, in principle, the larger σ^2 is, the smaller its expected distance is. The average expected path distance is proportional to μ and inversely proportional to σ^2.

For the balanced partition routing algorithm:

$$\begin{cases} a_{ij} \sim N(\mu_1, \sigma_1^2),\ i,\ j \subseteq \varphi \\ a_{ij} \sim N(\mu_2, \sigma_2^2),\ i \subseteq \varphi,\ j \nsubseteq \varphi \end{cases}$$
$$\mu_1 < \mu < \mu_2,\ \sigma_1^2 < \sigma^2 < \sigma_2^2$$

$$C_{path}(N) = \frac{\frac{N}{3}-1}{N-1}\cdot\mu_1\cdot\epsilon_1\cdot\frac{1}{\sigma_1^2} + \frac{\frac{2N}{3}}{N-1}\cdot\left(2\mu_1\cdot\epsilon_1\cdot\frac{1}{\sigma_1^2} + \mu_2\cdot\epsilon_2\cdot\frac{1}{\sigma_2^2}\right) < C_{path\epsilon} \tag{14}$$

Among them:

$$C_{path\epsilon} = 2\mu_1\cdot\epsilon_1\cdot\frac{1}{\sigma_1^2} + \mu_2\cdot\epsilon_2\cdot\frac{1}{\sigma_2^2} \tag{15}$$

Further, it can be concluded that:

$$G_{path}(N) - C_{path\epsilon} = \frac{N}{2}\cdot\mu\cdot\epsilon\cdot\frac{1}{\sigma^2} - \left(2\mu_1\cdot\epsilon_1\cdot\frac{1}{\sigma_1^2} + \mu_2\cdot\epsilon_2\cdot\frac{1}{\sigma_2^2}\right) \tag{16}$$

The greater the value of N, the more likely the greedy algorithm is to be less efficient than the balanced partition routing algorithm. When the number of inter-star nodes is small, specific variance and average step size should be considered for specific analysis.

3.4. Simulation and Analysis

Firstly, the computational complexity of the above algorithm is simulated.

Figure 1 on the left is an actual simulation, and the data on the right are the logarithmic processing. As can be seen from the left figure, with the increase in the number of satellites, the computational complexity of the flooding and backtracking algorithm increases exponentially, while other algorithms increase slowly. From the figure on the right, in general, the flooding and backtracking algorithm has the highest computational complexity, followed by the two partition routing algorithms, and the greedy algorithm has the lowest computational complexity. When the number of inter-star nodes is not very large, the computational complexity of the balanced partition routing algorithm is slightly higher than that of an ordinary partition algorithm.

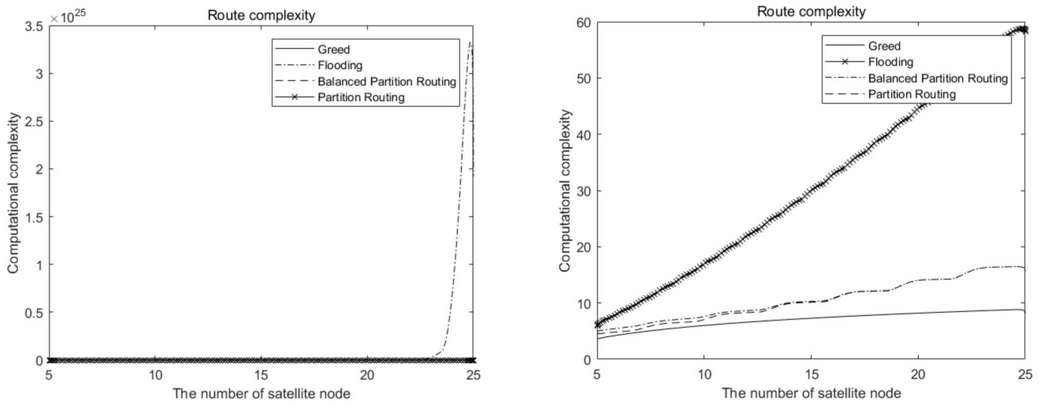

Figure 1. Simulation of computational complexity of three algorithms.

The computation force of a single star node under different algorithms is simulated:

As can be seen from the analysis of Figure 2, with the increase in the number of star nodes in the system, the requirement of the flooding and backtracking algorithm on the computing capacity of a single star node increases rapidly. However, the computing capacity of the two partition routing algorithms increases slowly for a single star node, even approaching the level of a greedy algorithm. This shows that a balanced partition routing algorithm can well distribute computing pressure to multiple star nodes, with stronger destruction resistance and better scalability.

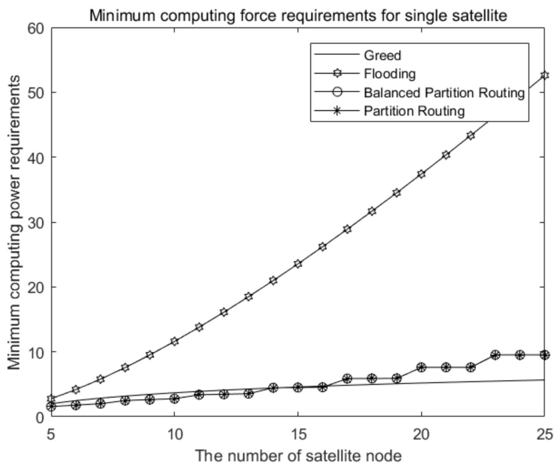

Figure 2. Simulation of calculated pressure for a single node.

Next, the efficiency of the above algorithms is simulated, and the inter-satellite node number is set to 15.

There were a total of 20 simulations. The average path delay of the greedy algorithm is the highest, and the variance of the average delay in each experiment is large. However, the efficiency of the flooding and backtracking algorithm is relatively stable, and the average delay calculated by this algorithm is significantly less than that of the two partition routing algorithms. Figure 3 shows that the average route length of the balanced partition routing algorithm is shorter than that of the classical partition routing algorithm. In other words, the path calculated by the balanced partition algorithm is better than that of the classical partition routing algorithm.

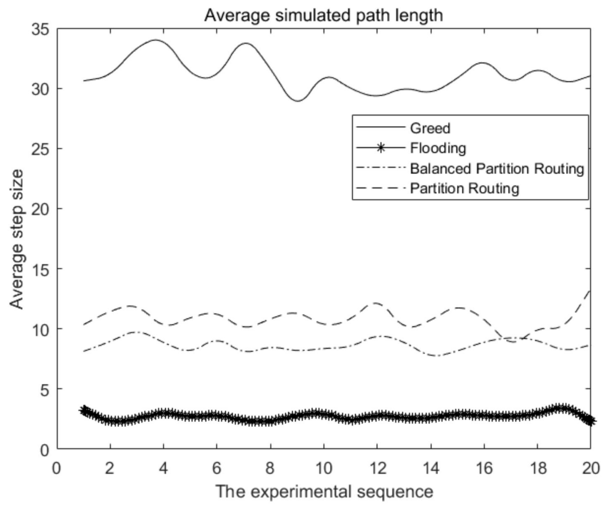

Figure 3. Simulation of average path length for four algorithms.

Twenty simulation experiments were conducted here. It can be seen from Figure 4 that the variance of each shortest path length calculated by the flooding and backtracking algorithm is nearly 0, followed by the two partition routing algorithms. The variance calculated by the greedy algorithm is the largest and the stability is the worst. However, considering that the computation complexity of the balanced partition routing algorithm is much lower than that of the flooding and backtracking algorithm, the balanced partition routing algorithm still has high application significance.

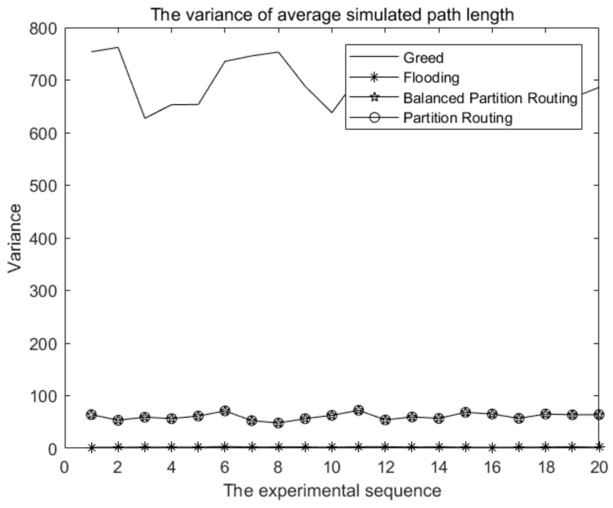

Figure 4. Simulation of the variance of average path length for four algorithms.

3.5. Analysis of Hardware Architecture Cost

The inter-star model is composed of nine star nodes. We randomly conducted 1000 communication missions among these star nodes. We only considered the calculation power consumption of these nodes in calculating the routing path, and the energy consumption of the nodes is positively correlated with the calculation amount. In the actual space environment, the payload of the satellite relies on solar panels to collect solar energy to provide energy.

Table 1 shows the calculation amount for the routing calculation for each node under different algorithms in a simulation experiment. Something that looks like 10^{12} or 10^5 is an order of magnitude. The amount of computation here is expressed only in numbers, not in units. The larger the value is here, the greater the amount of computation performed on this node. This means that this node will consume more power and require higher hardware costs. According to the principle of reliability, since we do not know the calculation pressure of each node in advance, the design index of each node should refer to the node with the largest calculation pressure. For example, in this experiment, the design index of the satellite nodes should refer to the No. 1 star under the flooding algorithm, and the No. 6 star under the greedy algorithm. In other words, the actual hardware design cost of satellite nodes is mainly determined by which node bears the most computational pressure.

Table 1. Amount of computation for each node.

Satellite Number / Types of Algorithms	1	2	3	4	5	6	7	8	9
Flooding (10^{12})	13.04	1.75	2.89	1.28	5.38	8.57	2.25	4.11	9.51
Greedy (10^4)	0.09	2.73	1.00	1.99	1.34	21.9	12.8	2.37	2.62
Partition routing (10^5)	6.11	6.36	5.22	7.31	5.45	6.26	6.21	6.32	6.25
Balanced partition routing (10^5)	7.62	7.70	7.71	7.67	7.88	7.63	7.67	7.64	7.60

In order to increase the reliability of the experiment, we conducted a total of 10 experiments. In Table 2, we omit the computation amount of each star node under various algorithms and only show the numerical value of the node with the most computation in each experiment.

Table 2. Largest amount of computation for single node in each experiment.

Types of Algorithms / Sequence	**Greedy** (10^5)	**Flooding** (10^{12})	**Partition Routing** (10^5)	**Balanced Partition Routing** (10^5)
1	2.19	13.04	7.31	7.88
2	1.21	9.63	6.29	7.99
3	7.41	9.99	6.38	8.06
4	3.53	3.92	6.72	8.09
5	1.75	1.26	6.02	8.08
6	6.44	2.57	5.98	8.16
7	5.32	19.9	5.68	8.14
8	8.01	3.78	6.48	8.18
9	2.11	2.82	6.46	8.28
10	1.07	6.11	6.18	8.22

It can be seen from this table that the computing capacity requirements of the flooding algorithm for the nodes are not in the same order of magnitude as those of the other three algorithms. If the flooding algorithm is used, the hardware cost for the star nodes will be very large. In theory, the average calculation pressure of each node of the greedy algorithm should be far less than that of the partition routing algorithm and balanced partition routing algorithm. However, each route calculation of the greedy algorithm can only be completed by one node independently and cannot be dispersed to multiple nodes. It leads to great calculation pressure on some nodes. According to our simulation results, the node with the greatest computational pressure of the greedy algorithm is the same order of magnitude as that of the two partition routing algorithms, and the hardware cost among them is also close. From this point of view, on the premise of obtaining extremely high algorithm effectiveness, the hardware cost of a balanced partition routing algorithm is much lower than the flooding algorithm and the cost is close to the greedy algorithm and partition routing algorithm, which have good economic benefits.

4. Optimization of Balanced Partition Routing Algorithm

This section mainly explores the two optimization directions of the balanced partition routing algorithm. As shown in Figure 5, in order to further reduce the calculation pressure of a single node, the balanced partition routing algorithm can further optimize the grouping on the basis of the original grouping. One way is to divide the larger group into smaller groups, each of which belongs to the larger group. This article calls this partitioning method 2. Another method is to divide the original large group into several smaller groups. After the separation, the original large group is cleared. We call it partitioning method 3. This section will explore the calculation pressure of a single node, average path length and variance of average path length under conditions of these two optimization directions.

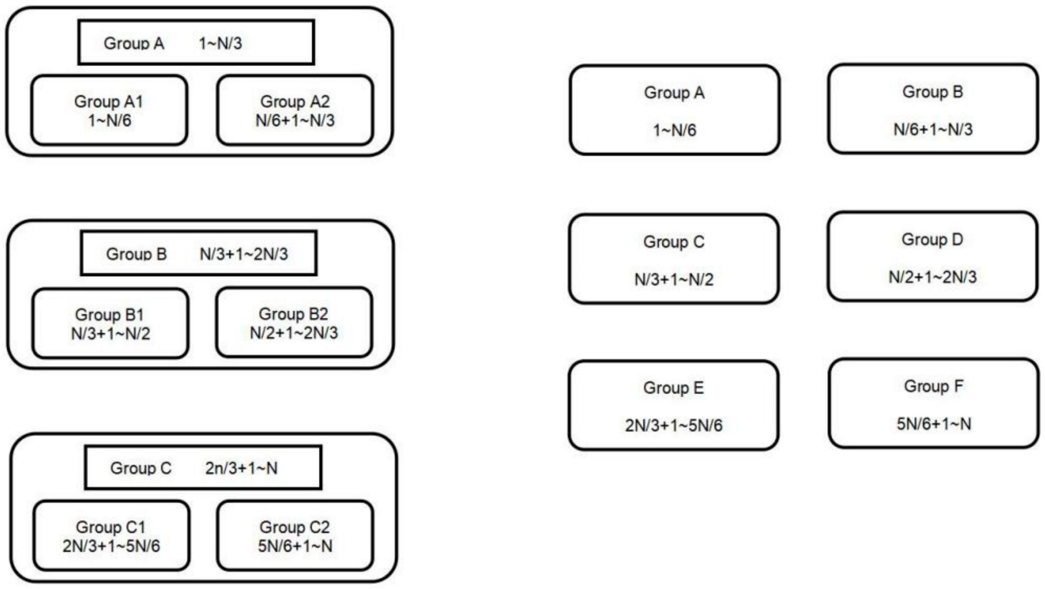

Figure 5. Two kinds of optimized partitioning (method 2 is left, method 3 is right).

4.1. Computational Complexity Analysis under Two Partitioning Methods

Method 1: Method 1 here is the balanced partition routing algorithm mentioned above, and its total computational complexity and computation pressure of a single node are shown in Equations (6) and (12).

Method 2: As shown on the left side of Figure 5, all nodes are divided into three large groups, with two small groups inside each large group. To calculate the routing path from star node numbered i to star node numbered j, there may be three cases to consider.

(1) In the first case, the two star nodes are in the same small group and the routing path is calculated directly in the group.

(2) In the second case, the two nodes are not in the same small group, but in the same large group. Assuming that the two communication nodes belong to small groups,

these two nodes are numbered i + x and j + y, respectively. The communication nodes of each group should be calculated first, and then the paths of these two nodes to their communication nodes should be calculated. Therefore, the total routing path is going to be: i -> i+x -> j+y -> j.

(3) In the third case, the two nodes are not in the same large group. For node i and node j, the first step is to calculate the communication nodes of large groups that these two nodes i, j, respectively, belong to. Assuming that two large communication nodes are numbered i + m and j + n. Assuming that the two small communication nodes in the large group where node i is located are $\mathrm{i}+\mathrm{m}_1$ and $\mathrm{i}+\mathrm{m}_2$. In the same way, we can obtain $\mathrm{j}+\mathrm{m}_1$ and $\mathrm{j}+\mathrm{m}_2$ and then we have to think about whether $\mathrm{i}\ +\ \mathrm{m}$ and i are in the same small group. We also need to consider whether $\mathrm{j}\ +\ \mathrm{n}$ and j are in the same group. There are four possibilities:

(3.1) The first possibility is that node $\mathrm{i}\ +\ \mathrm{m}$ and i are in the same small group and node $\mathrm{j}\ +\ \mathrm{n}$ and j are also in the same small group. The path in this possibility is going to be: $\mathrm{i}->\ \mathrm{i}+\mathrm{m}->\ \mathrm{j}+\mathrm{n}->\ \mathrm{j}$.

(3.2) The second possibility is that node $\mathrm{i}\ +\ \mathrm{m}$ and i are not in the same group while node $\mathrm{j}\ +\ \mathrm{n}$ and j are in the same group. In this possibility, we need to calculate the routing path from i to $\mathrm{i}\ +\ \mathrm{m}$ in the large group firstly and the total routing path is going to be: $\mathrm{i}->\mathrm{i}+\mathrm{m}_1->\ \mathrm{i}+\mathrm{m}_2->\mathrm{i}+\mathrm{m}->\mathrm{j}+\mathrm{n}->\mathrm{j}$.

(3.3) The third possibility is that node $\mathrm{i}\ +\ \mathrm{m}$ and i are in the same group while $\mathrm{j}\ +\ \mathrm{n}$ and j are not in the same group. The total routing path is: $\mathrm{i}->\mathrm{i}+\mathrm{m}->\mathrm{j}+\mathrm{n}->\mathrm{j}+\ \mathrm{j}+\mathrm{m}_1->\ \mathrm{j}+\mathrm{m}_2->\mathrm{j}$.

(3.4) The last possibility is the exact opposite of the first and its routing path is: $\mathrm{i}->\mathrm{i}+\mathrm{m}_1->\ \mathrm{i}+\mathrm{m}_2->\mathrm{i}+\mathrm{m}->\mathrm{j}+\mathrm{n}->\mathrm{j}+\mathrm{m}_1->\ \mathrm{j}+\mathrm{m}_2->\mathrm{j}$.

Minimum computing force requirements for a single node:

$$\mathrm{C}_{\mathrm{method\ 2}_{\mathrm{node}}}(\mathrm{N}) = \left(\sum_{\mathrm{m}=0}^{\frac{N}{12}-1} \frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-\mathrm{m}\right)!} + \delta_2 + \left(\frac{N}{6}\right)^2 + \left(\frac{N}{3}\right)^2\right) \tag{17}$$

Total algorithm complexity:

$$\mathrm{C}_{\mathrm{method\ 2}}(\mathrm{N}) = \mathrm{U}_1 + \mathrm{U}_2$$

$$\mathrm{U}_1 = \frac{\frac{N}{6}-1}{N-1}\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{(\frac{N}{6}-2)!}{(\frac{N}{6}-2-\mathrm{m})!} + \delta_2\right) + \frac{\frac{N}{6}}{N-1}\cdot\left(2\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{(\frac{N}{6}-2)!}{(\frac{N}{6}-2-\mathrm{m})!} + \delta_2\right) + \left(\frac{N}{6}\right)^2\right))$$

$$\mathrm{U}_2 = \frac{\frac{N}{6}}{N-1}\left[\left(2\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{(\frac{N}{6}-2)!}{(\frac{N}{6}-2-\mathrm{m})!} + \delta_2\right) + \left(\frac{N}{3}\right)^2\right) + 2\cdot\left(3\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{(\frac{N}{6}-2)!}{(\frac{N}{6}-2-\mathrm{m})!} + \delta_2\right) + \left(\frac{N}{3}\right)^2 + \left(\frac{N}{6}\right)^2\right) + \left(4\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{(\frac{N}{6}-2)!}{(\frac{N}{6}-2-\mathrm{m})!} + \delta_2\right) + \left(\frac{N}{3}\right)^2 + 2\cdot\left(\frac{N}{6}\right)^2\right)\right]$$

Simplified to obtain:

$$\mathrm{C}_{\mathrm{method\ 2}}(\mathrm{N}) = \frac{\frac{N}{6}-1}{N-1}\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-\mathrm{m}\right)!} + \delta_2\right) + \frac{\frac{N}{6}}{N-1}\cdot\left(14\cdot\left(\sum_{\mathrm{m}=0}^{\frac{N}{6}-2} \frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-\mathrm{m}\right)!} + \delta_2\right) + 3\cdot\left(\frac{N}{3}\right)^2 + 4\cdot\left(\frac{N}{6}\right)^2\right)$$

$$\delta_2 = \frac{\left(\frac{\mathrm{N}}{6}-2\right)\cdot\frac{\mathrm{N}}{6}}{2} + \delta_{\mathrm{recall}} \tag{18}$$

Method 3: As shown in Figure 5, method 3 is similar to method 1. Nodes in the system are evenly divided into more small groups. All nodes are divided into six groups. To calculate the routing path from star node numbered i to star node numbered j, there may be two cases to consider.

(1) In the first case, the two star nodes are in the same small group and the routing path is calculated directly in the group.

(2) In the second case, the two nodes are not in the same group. Assuming that the two communication nodes of the groups these two nodes belong to are numbered i + x and j + y, respectively. Therefore, the total routing path is going to be: i->i+x->j+y->j.

Minimum computing force requirements for a single node:

$$C_{\text{method } 3_{\text{node}}}(N)=\sum_{m=0}^{\frac{N}{12}-1}\frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-m\right)!}+\left(\frac{N}{6}\right)^2+\delta_2 \tag{19}$$

Total algorithm complexity:

$$C_{\text{method } 3}(N)=\left(\frac{\frac{N}{6}-1}{N-1}\cdot\sum_{m=0}^{\frac{N}{12}-1}\frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-m\right)!}+\delta_2+\frac{\frac{5N}{6}}{N-1}\cdot\left(\left(\frac{N}{6}\right)^2+\left(\sum_{m=0}^{\frac{N}{12}-1}\frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-m\right)!}+\delta_2\right)\cdot 5\right)\right)\cdot N^2 \tag{20}$$

Then:

$$C_{\text{method } 2}(N)-C_{\text{method } 3_{\text{node}}}(N)=\left(\frac{N}{3}\right)^2>0 \tag{21}$$

$$C_{\text{method } 2}(N)-C_{\text{method } 2}(N)=\frac{\frac{N}{6}-1}{N-1}\cdot\left(9\cdot\left(\sum_{m=0}^{\frac{N}{6}}\frac{\left(\frac{N}{6}-2\right)!}{\left(\frac{N}{6}-2-m\right)!}+\delta_2\right)+3\cdot\left(\frac{N}{3}\right)^2-\left(\frac{N}{6}\right)^2\right) \tag{22}$$

Therefore, both the computational complexity of a single node and the total computational complexity of method 2 is greater than that of method 3.

4.2. Algorithm Efficiency Analysis

The inter-star model we selected is composed of 30 star nodes. The data in the Topo table represent the direct distance between stars and satisfy the distribution of $(\mu,\ \sigma^2)$. We explore the actual performance of two balanced partition routing algorithms under different variances. We first simulate the average routing path delay calculated by the two optimization algorithms under different variances and the μ is 5. The simulated data are shown in Table 3.

Table 3. Data of simulated time average path cost.

Conditions / Sequence	$\mu = 5,\ \sigma^2 = 6$			$\mu = 5,\ \sigma^2 = 10$			$\mu = 5,\ \sigma^2 = 15$		
	Algorithm 1	**Algorithm 2**	**Algorithm 3**	**Algorithm 1**	**Algorithm 2**	**Algorithm 3**	**Algorithm 1**	**Algorithm 2**	**Algorithm 3**
1	6.9727	7.5565	7.7721	8.9871	11.7425	11.1119	13.3474	19.5487	12.4728
2	7.0034	7.1166	5.8530	8.5744	8.9300	7.9212	14.4343	17.5625	15.6544
3	6.3184	6.1110	7.3742	8.6316	9.8857	13.3193	11.9705	15.7141	11.0548
4	6.7229	8.0088	6.3971	9.4518	10.9890	10.8876	12.9407	16.1883	18.7025
5	6.8002	7.8050	6.9728	8.0303	12.5435	11.8771	14.0550	20.2276	12.5364
6	6.3598	6.6342	9.6275	10.0427	13.6140	8.8530	15.1775	18.2937	14.5067
7	6.4300	10.3892	8.0600	9.33365	15.3480	9.5072	11.2382	14.7445	10.6957
8	6.5192	7.7034	6.9728	9.0030	14.7181	10.7664	12.3992	11.9626	15.5931
9	6.5477	8.1679	7.8820	9.5419	12.8866	11.8589	15.3509	21.4360	21.0220
10	6.4357	7.7889	7.7813	8.1095	11.3909	10.6629	13.1029	12.5283	9.7106
average	6.611	7.728	7.469	8.971	12.205	10.677	13.402	16.821	14.195

Notation: The units of the metrics in this table are second(s).

It can be seen from the above table that we carried out 10 simulation experiments. According to the average value of data obtained from the 10 simulations, the routing path delay calculated by algorithm 1 is the best, algorithm 3 is the second, and algorithm 2 is the worst. In order to eliminate chance, we performed another 20 simulations and drew a simulation graph of the average routing path delay.

Figures 6–8 are the simulation diagrams of the actual average route path length calculated by two optimization algorithms with different variances. As can be seen from the figures, an overall trend is that with the increase in data variance in the Topo table, the average path length calculated by the three algorithms increases. However, in terms of internal comparison, the average path length obtained by algorithm 1 is the shortest, while that obtained by algorithm 2 is the longest.

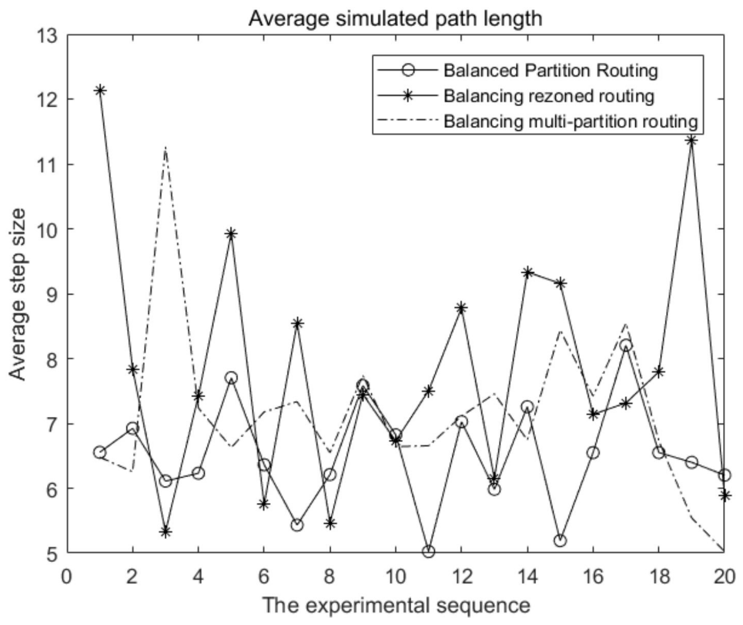

Figure 6. Simulation of average path length ($\mu = 5, \sigma^2 = 5$).

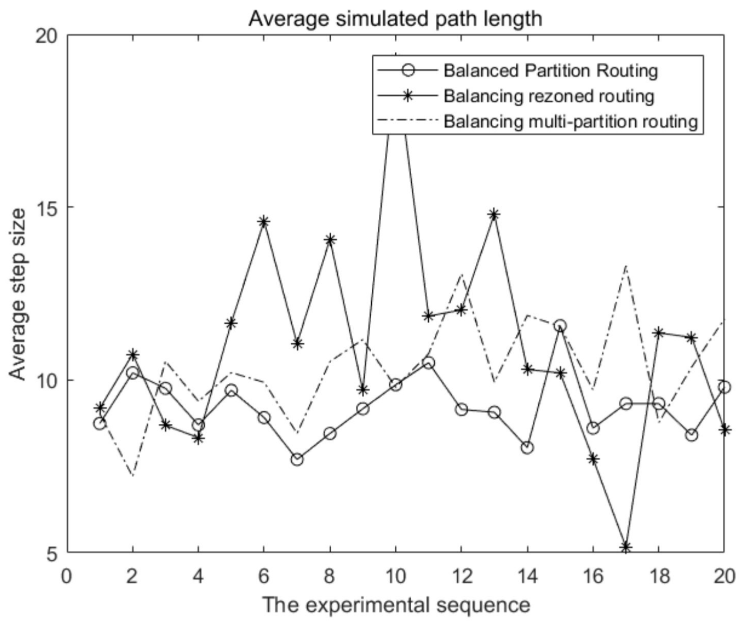

Figure 7. Simulation of average path length ($\mu = 5, \sigma^2 = 10$).

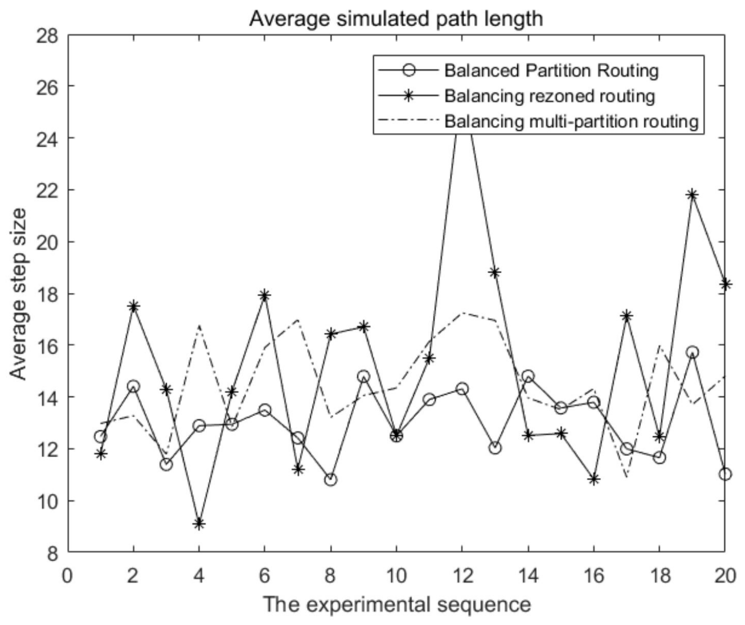

Figure 8. Simulation of average path length ($\mu = 5, \sigma^2 = 15$).

As can be seen from the Tables 3 and 4, as the variance of data in the Topo table increases, so does the variance of the average path length calculated by each algorithm. Internally, under the same conditions, the variance of the average path length calculated by algorithm 2 is the smallest, indicating that the path length calculated by algorithm 2 for each routing path is relatively close, and the efficiency of the algorithm is the most stable. Figures 9–11 also reflect the same results. In practical applications, such algorithms are more resistant to destruction. If the lengths of different paths calculated differ greatly, some star nodes will be overloaded while others are idle, which wastes resources and reduces the efficiency of inter-star communication [24].

4.3. Analysis of Hardware Architecture Cost

The inter-star model is composed of 18 star nodes. Other experimental details and methods are exactly the same as the simulation experiment in 3.5. Table 5 shows the calculation amount of routing calculation for each node under three algorithms in a simulation experiment.

Table 4. Data of variance of simulated time average path cost.

Conditions / Sequence	$\mu = 5, \sigma^2=6$			$\mu = 5, \sigma^2=10$			$\mu = 5, \sigma^2 = 15$		
	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 1	Algorithm 1
1	17.8395	19.1632	14.8941	28.2624	42.7909	57.4118	89.3170	63.3746	72.3029
2	20.1273	18.5325	28.2469	39.2474	19.0063	39.1743	91.8820	79.7508	104.3368
3	16.1741	16.3519	26.0751	47.5303	36.6430	51.3725	85.5774	52.7638	124.6169
4	14.2612	12.0631	21.4181	44.6129	36.2415	48.4760	80.2696	55.0811	84.7636
5	18.3008	12.6547	15.6712	46.5585	32.7720	63.5157	94.9057	89.1324	85.5036
6	26.0488	17.1340	21.1393	41.6227	23.0475	35.6929	84.2496	52.8075	102.9034
7	13.5860	14.0287	20.7693	30.3912	34.6619	50.8909	82.9925	97.9406	131.0594
8	19.6410	15.0383	17.7038	34.5207	23.5419	66.3233	96.1250	56.6428	96.4693
9	17.2942	23.9701	22.5576	40.6512	48.1517	44.1655	54.0130	73.5422	108.1908
10	19.7617	14.4214	33.9848	54.4098	43.6303	50.8788	90.8565	49.9056	126.0783
average	18.303	16.336	22.460	40.781	34.049	50.790	85.019	67.095	103.623

Notation: The units of the metrics in this table is s^2.

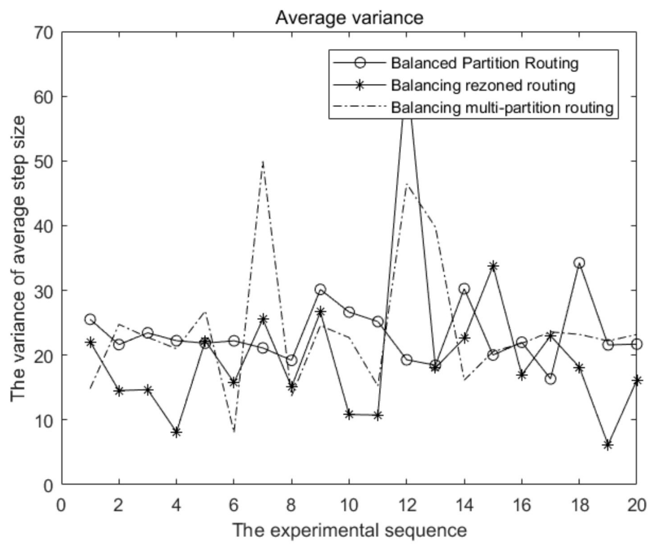

Figure 9. Simulation of the variance of average path length ($\mu = 5, \sigma^2 = 5$).

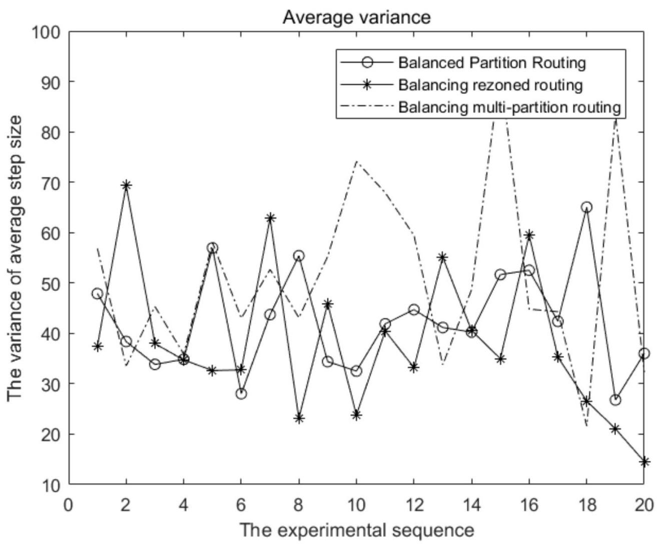

Figure 10. Simulation of the variance of average path length ($\mu = 5, \sigma^2 = 10$).

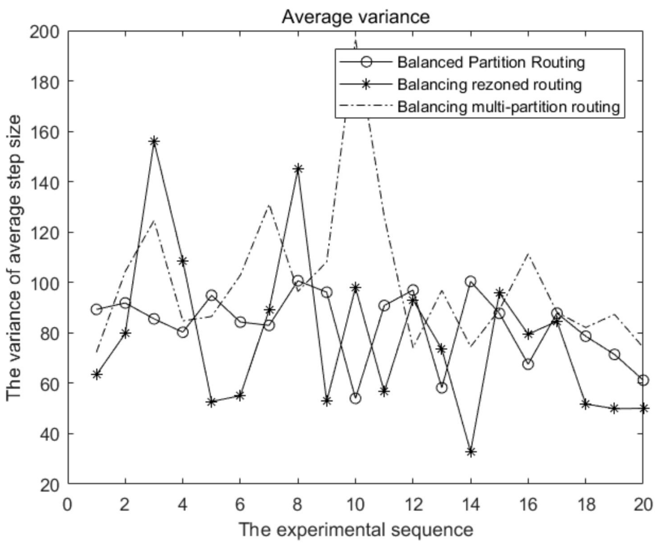

Figure 11. Simulation of the variance of average path length ($\mu = 5, \sigma^2 = 15$).

Table 5. Amount of computation for each node.

Types of Algorithms \ Satellite Number	**1** / **10**	**2** / **11**	**3** / **12**	**4** / **13**	**5** / **14**	**6** / **15**	**7** / **16**	**8** / **17**	**9** / **18**
Method1 (10^6)	1.83	1.85	1.90	1.93	1.82	1.96	1.93	1.87	1.85
	1.84	1.90	1.84	1.86	1.90	1.88	1.90	1.88	1.85
Method2 (10^5)	4.74	4.94	5.12	5.32	5.37	4.99	4.95	5.18	5.11
	5.17	5.17	4.72	5.12	5.00	4.86	4.87	5.27	4.85
Method3 (10^5)	6.05	6.19	6.37	6.17	6.20	6.08	6.34	6.18	6.23
	6.08	6.26	6.21	6.06	6.18	6.13	6.18	6.15	6.15

Table 5 shows the calculation power consumption of each star node under various algorithms in the first experiment. Table 6 reflects the node with the highest calculated pressure in the 10 experiments. It can be seen from the two tables that the computational pressure borne by each node under these three algorithms is relatively average, which means that each node was fully utilized. To be specific, the average and maximum computational pressures borne by each node of the two algorithms after optimization are indeed significantly lower than those before optimization, which proves that the two optimization directions can further reduce hardware costs.

Table 6. Largest amount of computation for single node in each experiment.

Sequence \ Types of Algorithms	**Method** 1 (10^6)	**Method** 2 (10^5)	**Method** 3 (10^5)
1	1.96	5.37	6.37
2	2.02	5.46	6.34
3	2.07	5.48	6.39
4	2.06	5.23	6.51
5	2.10	5.30	6.14
6	2.04	5.38	6.20
7	2.08	5.26	6.41
8	2.08	5.47	6.30
9	2.02	5.87	6.34
10	2.06	5.30	6.22

5. Conclusions

This paper proposes a new inter-satellite balanced partition routing algorithm. Firstly, the total computation complexity and single node's pressure of the algorithm and other traditional algorithms are analyzed from the point of view of mathematics. The simulation model is built to simulate the performance of the four inter-satellite routing algorithms, including average path propagation delay, the variance of the path propagation delay, and the calculated pressure of a single node. Simulation results reveal that compared with other algorithms, the proposed balanced partition routing algorithm achieves better performance.

In the latter part of this paper, two further optimization directions of a balanced partition routing algorithm are analyzed. Firstly, experimental results show that these two optimization methods can further reduce the complexity of the algorithm. Moreover, the hardware cost of nodes is greatly reduced. Finally, the further optimization of the balanced partition routing algorithm can adapt to various scenarios and has good mobility. This algorithm could not only be used as an alternative scheme in the routing planning of low-orbit micro-satellite constellations and star links but also be used as a general scheme for decentralized data fusion in cooperative systems such as unmanned aerial vehicle formations.

In our following work, we intend to study the grouping of balanced partition routing algorithms. In our branch of experimental study, we found that the average path delay calculated by different grouping methods is huge for the balanced partition routing algorithm. The good grouping method will greatly improve the efficiency of the algorithm. On the other hand, the processing delay of data within the nodes will also be considered in our future research [1].

Author Contributions: Conceptualization, Z.X., Z.J. and X.J.; Formal analysis, Z.X. and Z.J.; Methodology, H.C. and X.G.; Resources, J.Y. and K.X.; Software, H.C. and X.G.; Validation, H.C. and S.L.; Writing—original draft, H.C.; Writing—review and editing, H.C. and Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62073289.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

N	Number of satellite nodes in the system
Topo	A table storing direct distances between stars
No.	Satellite identification
a_{ij}	Direct distance from the No. i to the No. j satellite
μ	Average of the data in Topo table
σ^2	Variance of the data in Topo table
φ	A group
G	C omputational complexity of Greedy algorithm
F	C omputational complexity of Flooding algorithm
δ_1	Cost of backtracking part of the Flooding algorithm
P	C omputational complexity of Partition routing algorithm
δ_2	Cost of $\delta_{\text{recall}} + \delta_{\text{greed}}$
δ_{recall}	Cost of backtracking part of Partial Flooding algorithm
δ_{greed}	Overhead of the greedy algorithm in the group
C	C omputational complexity of Balanced partition routing algorithm
G_{node}	Calculated pressure of a node in Greedy algorithm
F_{node}	Calculated pressure of a node in Flooding algorithm
P_{node}	Calculated pressure of a node in Partial Flooding algorithm
C_{node}	Calculated pressure of a node in of Balanced partition routing algorithm
G_{path}	Average path length of Greedy algorithm
F_{path}	Average path length of Flooding algorithm
P_{path}	Average path length of Partial Flooding algorithm
C_{path}	Average path length of Balanced partition routing algorithm
$C_{method\ 2_{node}}$	Calculated pressure of a node in of Balanced partition routing algorithm (2)
$C_{method\ 3_{node}}$	Calculated pressure of a node in of Balanced partition routing algorithm (3)
$C_{method\ 2}$	Average path length of Balanced partition routing algorithm (2)
$C_{method\ 3}$	Average path length of Balanced partition routing algorithm (3)

References

1. Alishahi, M.; Hajiabolhassan, H. On chromatic number and minimum cut. *J. Comb. Theory Ser. B* **2019**, *139*, 27–46. [CrossRef]
2. Balas, E.; Yu, C.S. Finding a Maximum Clique in an Arbitrary Graph. *SIAM J. Comput.* **1986**, *15*, 1054–1068. [CrossRef]
3. Bondy, J.A.; Murty, U.S.R. *Graph Theory*; Springer: New York, NY, USA, 2008.
4. Bonomo, F.; Cornaz, D.; Ekim, T.; Ries, B. Perfectness of clustered graphs. *Discret. Optim.* **2013**, *10*, 296–303. [CrossRef]
5. Kergosien, Y.; Lenté, C.; Billaut, J.-C.; Perrin, S. Metaheuristic algorithms for solving two interconnected vehicle routing problems in a hospital complex. *Comput. Oper. Res.* **2013**, *40*, 2508–2518. [CrossRef]

6. Chen, J.; Liu, S.; Zhang, Y. Inter-Satellite Token Ring Ad Hoc Network Technology for Micro-Nano Satellite Cluster Collaboration. In Proceedings of the 2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication (CTISC), Shanghai, China, 23–25 April 2021; pp. 125–131. [CrossRef]
7. Dong, C.; Xu, X.; Liu, A.; Liang, X. Load balancing routing algorithm based on extended link states in LEO constellation network. *China Commun.* **2022**, *19*, 247–260. [CrossRef]
8. Chen, Q.; Giambene, G.G.; Yang, L.; Fan, C.; Chen, X. Analysis of Inter-Satellite Link Paths for LEO Mega-Constellation Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2743–2755. [CrossRef]
9. Weng, Y.; Liu, L. On-demand partial topology routing algorithm in LEO satellite networks. *Comput. Eng. Appl.* **2007**, *43*, 148–150.
10. Henderson, T.H.; Katz, R.H. On Distributed and Geographic-Based Packet Routing for LEO Satellite Networks. In Proceedings of the Globecom '00—IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137), San Francisco, CA, USA, 27 November–1 December 2000; Volume 2, pp. 1119–1123.
11. Ekici, E.; Akyildiz, I.F.; Bender, M.D. A Distributed Routing Algorithm for Data gram Traffic in LEO Satellite Networks. *IEEE J. Sel. Areas Commun.* **2004**, *22*, 272–286.
12. Woo, S.; Ohara, M.; Torrie, E.; Singh, J.; Gupta, A. The SPLASH-2 programs: Characterization and methodological considerations. In Proceedings of the 22nd annual international symposium on Computer architecture—ISCA '95, Santa Margherita Ligure, Italy, 22–24 June 1995; pp. 24–36.
13. Bienia, C.; Kumar, S.; Singh, J.P.; Li, K. The Parsec Benchmark Suite: Characterization and Architectural Implications. In Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, Toronto, ON, Canada, 25–29 October 2008; pp. 72–81.
14. Bienia, C.; Kumar, S.; Li, K. PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors. In Proceedings of the IEEE International Workshop/Symposium on Workload Characterization, Seattle, WA, USA, 14–16 September 2008; pp. 47–56. [CrossRef]
15. Patel, A.; Ghose, K. Energy-Efficient Mesi Cache Coherence with Pro-Active Snoop Filtering for Multicore Microprocessors. In Proceedings of the 13th international symposium on Low power electronics and design, Bangalore, India, 11–13 August 2008; pp. 247–252.
16. Zeng, G.; Zhan, Y.; Pan, X. Failure-Tolerant and Low-Latency command in Mega-Constellations: The Redundant Multi-Path Routing. *IEEE Access* **2021**, *9*, 34975–34985. [CrossRef]
17. Martin, M.M.K.; Sorin, D.J.; Beckmann, B.M.; Marty, M.R.; Xu, M.; Alameldeen, A.R.; Moore, K.E.; Hill, M.D.; Wood, D.A. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *ACM SIGARCH Comput. Arch. News* **2005**, *33*, 92–99. [CrossRef]
18. Yang, J.; Li, D.; Jiang, X.; Chen, S.; Hanzo, L. Enhancing the Resilience of Low Earth Orbit Remote Sensing Satellite Networks. *IEEE Netw.* **2020**, *34*, 304–311. [CrossRef]
19. Zheng, B.; Yen, K.; Peng, X.; Zhang, R. Angle Partition-Based TDMA for VDES Satellite Multiuser Downlink Communications. In Proceedings of the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
20. Yang, K.; Zhang, B.; Guo, D. Controller and Gateway Partition Placement in SDN-Enabled Integrated Satellite-Terrestrial Network. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019; pp. 1–6.
21. Ebrahimi, M.; Daneshtalab, M.; Farahnakian, F.; Plosila, J.; Liljeberg, P.; Palesi, M.; Tenhunen, H. HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks. In Proceedings of the 2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP), Austin, TX, USA, 2 June 2013; pp. 19–26. [CrossRef]
22. Dehyadegari, M.; Daneshtalab, M.; Ebrahimi, M.; Plosila, J.; Mohammadi, S. An adaptive fuzzy logic-based routing algorithm for networks-on-chip. In Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), San Diego, CA, USA, 6–9 June 2011; pp. 208–214. [CrossRef]
23. Ebrahimi, M.; Daneshtalab, M.; Liljeberg, P.; Plosila, J.; Tenhunen, H. Exploring partitioning methods for 3D Networks-on-Chip utilizing adaptive routing model. In Proceedings of the Fifth IEEE/ACM International Symposium on Networks on Chip, Pittsburgh, PA, USA, 1–4 May 2011; pp. 73–80. [CrossRef]
24. Wang, L.; Jin, Y.; Kim, H.; Kim, E.J. Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip. In Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, Washington, DC, USA, 10–13 May 2009; pp. 64–73. [CrossRef]