A Fast Percolation-Dijkstra SPF Method for Mega-Constellation Satellite Network Routing

Shenshen Luan, Member, IEEE, Luyuan Wang, Yepeng Liu, Ninghan Sun, Ran Zhang

***Abstract*—Real-time routing for satellite communication of mega-constellations is a challenging task due to the large scale of network nodes, especially on devices with limited computation, such as onboard embedded systems. Hence, this paper proposes a fast-SPF method for mega-constellation backbone network routing. Firstly, inspired by the regularity and sparse characteristics of mega-constellations, the 4-degree percolation theory is proposed to describe the node search process. Then, dynamic minimum search and mapping methods are used to narrow down the traversal range. The proposed method performs equally well as the heap-optimized Dijkstra algorithm but with less memory space and dynamic access. The experimental results show that the proposed method can significantly reduce routing computation time, especially onboard, edge-computing, or other computation-limited devices.**

***Index Terms*—mega-constellation routing, 4-degree percolation, routing algorithm, satellite network, Dijkstra algorithm.**

I. INTRODUCTION

In recent years, satellite Internet built on the low-Earth orbit(LEO) mega-constellations has developed rapidly[1-2], such as Starlink, Kuiper, and Oneweb. In the space segment, the satellite network mainly comprises of mega-constellation backbone networks containing many satellite nodes. For example, there are 72x22 satellites in the Starlink K1 layer and 34x34 in the Kuiper K1 layer. When the scale of the network nodes significantly increases, it usually leads to slower routing calculation time [3], which is particularly severe on embedded devices with limited computing resources, such as onboard computers. Therefore, improving the efficiency of large-scale network routing calculation on the onboard computers has become a critical problem that must be solved.

The Shortest-Distance Path (SPD) algorithm is a classic network routing calculation method used in the mega-constellations networks [4-6]. Besides, the Dijkstra algorithm adopts a greedy search strategy to traverse all nodes in the network for shortest path calculation and search. However, traversing without considering the topology imposes poor real-time performance in solving sparse-graph routing problems. Thus, Eramo V et.al.[7] proposed a heap-optimization method to improve the Dijkstra algorithm and reduce time complexity in sparse graphs. However, the heap operation and complex data structures increase computational costs. Zhang et al. [8] employed using Dijkstra to calculate the minimum hop routing in satellite and ground-integrated networks in the architecture of large-scale network routing algorithms. For multi-layer networks, Ma et.al. [9] used Dijkstra to obtain the optimal routing table for MEO satellites and achieved domain clustering of LEO satellites. Dijkstra has also addressed advantageous switching sequences in graph-based user-satellite switching frameworks in LEO satellite networks [10]. However, due to complex data structures and traversing operations, the above methods do not meet the real-time requirements of network communication on embedded platforms or systems. To overcome that concern, Chen et al. [11-12] developed an analytical model-based shortest path routing algorithm considering the mega-constellation network topology characteristics, thereby effectively reducing routing computation time. Pan et al. [13] introduced an Orbit Prediction Shortest Path First (OPSPF) algorithm for elastic LEO satellite networks using the terrestrial routing protocol OSPF. Besides, Li et al. [14] described a time-varying topology for random transmission requirements and random packet generation/arrival. They proposed an effective Net-grid based Shortest Path Routing (NSPR) algorithm for large-scale satellite networks. However, NSPR's high complexity makes it unsuitable for the operation of onboard embedded devices.

In addition to large-scale nodes, the limited computing resource of the onboard embedded computer in the satellite network is also a key consideration in designing routing algorithms. Inspired by the regularity and sparsity characteristics, a Percolation Dijkstra algorithm including four-degree percolation and dynamic min-search, is proposed to accelerate the routing calculation of the mega-constellation backbone network.

The remainder of this paper is organized as follows: Section II introduces the proposed method and its detailed procedures. Section III compares the proposed methods' time complexity with other algorithms. Section IV presents the results on embedded platforms and discusses the findings. Finally, Section V highlights the contributions and concludes this paper.

Manuscript received XX March 2024; revised XX April 2024; accepted XX XXX 2024. Date of publication XX XXX 2024. (*Corresponding author: Luyuan Wang*).

Shenshen Luan is with Electronic and Information Engineering School, Beihang University, Beijing, 100191, China (e-mail: luanshenshen@buaa.edu.cn).

Luyuan Wang is with Department of Electronic Engineering and Information Science, University of Science and Technology of China, 230022, (email: wang_luyuan@sina.com).

Yepeng Liu, Ninghan Sun and Ran Zhang are with School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (email: LiuYepeng2020@gmail.com, sunnh@bupt.edu.cn, zhangran@bupt.edu.cn.).

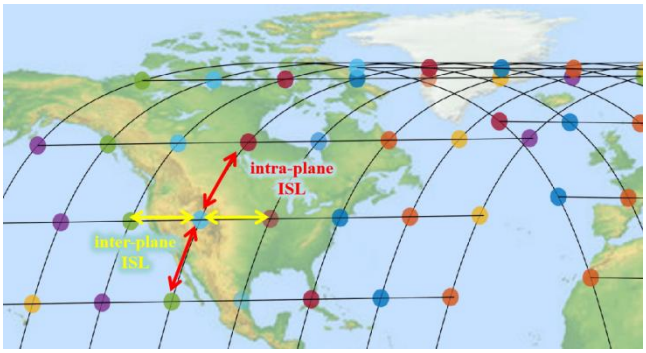

Fig. 1. Mesh-like topology of the mega-constellation backbone network and the cross-shaped unit's inter/intra plane ISL

II. Methodology

Usually there are two parameters, link costs and hops, for computing the network SPF routing strategy. Here, the link cost generated by random function is chosen for routing computation.

A. Adjacent Matrix of Mega-Constellation

The graph-based routing algorithm typically describes the network topology with nodes and edges. In the proposed method, an adjacency matrix describes the network topology and serves as an input variable for programming implementation. For a mega-constellation composed of M orbits and N satellites in each orbit, the number of rows and columns of its adjacency matrix is M*N. The element values of the adjacency matrix represent the cost between any two satellite nodes, where diagonal element 0 represents the satellite node itself, floating-point numbers from 0 to 1 represent link cost, and NULL represents the inability of the two satellites to connect.

Fig. 2 depicts an adjacent matrix of a 3*5 topology, which means 3 orbits and 5 satellites in each orbit. The cost is set as 1 for convenience. In Fig. 2, the elements with a gray, blue, and orange background denote the inter-plane ISL link cost, the intra-plane ISL link cost and the link cost between the first and last planes ISL, respectively. It can be seen that the link cost is distributed along the diagonal line of the matrix, and there are 6 main link-cost lines in the matrix, regardless of the size of the matrix. This is the sparse characteristic of the mega-constellation. Although the inter-plane link will be connected or disconnected during the satellite motion, the link cost in the adjacent matrix only changes its values instead of changing its positions, which is the regularity characteristics.

In addition, a percolation array, a visited array, and a distance array will be set up to record the percolation results, visited node index, and current shortest distance, respectively, similar to the Dijkstra algorithm. The initial value of the percolation array is set to null. Additionally, the percolation array is length-varying with the input/output of the visited nodes during the iteration of the percolation, which can help reduce the traversing times of the minimum search.

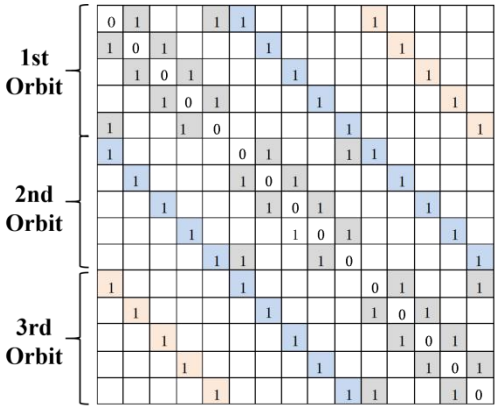

Fig. 2. Adjacent matrix of a mega-constellation with link cost of different ISLs.

B. 4-degree Percolation

The Dijkstra-based algorithm usually needs to traverse all remaining nodes when searching for neighboring nodes, resulting in a computation equal to the number of nodes N. According to the 4-degree sparse feature of the backbone network, the search should percolate along a regular path instead of traversing all nodes. In the adjacency matrix, searching for the connected nodes of the i-th satellite only requires calculating and manipulating the values of the matrix element A[i, j], where j belongs to [i-1, i+1, i+N, i-N]. A[i, j] is the adjacent matrix and N is the satellite number of each orbit. If $distance(\mathrm{i}) + \mathrm{w}(\mathrm{i},\mathrm{j}) < distance(j)$, the satellite j is the next shortest-distance node. Add the index j to the percolation array and update the distance to the corresponding position in the distance array.

Fig. 3 illustrates an example of finding a mesh network's shortest-distance path from node 1 to node 9. Every column of the network topology is in the same orbit plane of the mega-constellation. When starting node 1 searches for the next shortest-distance node, it just needs to check the link costs with its neighbor 4 nodes rather than all 25. It works like percolation rather than flooding water on the network topology. According to the shortest-distance principle, node 2 is the next starting node. With this strategy, the shortest-distance path can be found after 4 times percolation and the shortest path is 1-2-5-8-9.

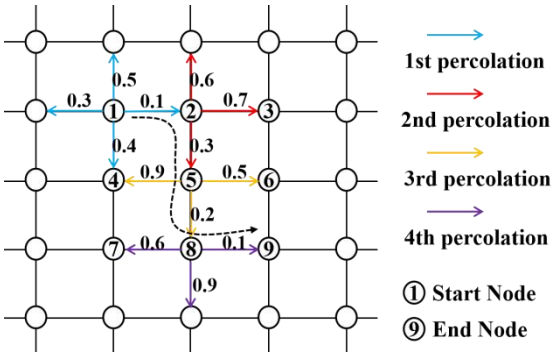

Fig. 3. Example of finding the shortest-distance path on the mesh topology using 4-degree percolation .

C. Dynamic Min-Search

After updating the percolation array, the shortest distance node should be searched and set as the next node of 4-degree percolation. Fig. 3 presents a traverse of all the elements in the percolation array and a search for the corresponding minimum value in the distance array. Then, we record the min-value's node index α and set the visited array value of the node α to 1. Finally, we set α as the next node to perform 4-degree percolation and remove α from the percolation array. As iterations continue, the number of elements in the percolation array dynamically changes. Fig. 4 illustrates the flowchart of the proposed method.

Algorithm 1 Dynamic Min-Research

Input: Percolation_array
Output: Percolation_array, Next_node, Visited_array

```
SourceNode = 0;
Min = INF;
FOR I = 1 : length(Percolation_array)
    Percolation_node = Percolation_array(i);
    IF Min > distance_array(Percolation_node)
        Min = distance_array(Percolation_node);
        SourceNode = Percolation_node;
    END
END
IF SourceNode != 0
    Visited_array(SourceNode) = 1;
    Next_node = SourceNode;
    Remove Percolation_mode from
Percolation_array
END
```

III. Complexity Analysis

To verify the proposed method's effectiveness theoretically, we analyzed its time complexity and compared it with the Dijkstra algorithm.

For the 4-degree percolation method, the calculation times of each node are reduced from N to 4 due to the limitation of the maximum node number. Due to the dynamic number of elements in the percolation array in the dynamic minimum search method, a quantitative description of the calculation times X cannot be determined. But X can be estimated averagely using the Monte Carlo algorithm. Based on the experimental results presented in Section IV, it can be determined that the value of X is approximately N/7.5 after 1000 experiments. Therefore, the total computation times of the proposed method is (4+X) * N. By comparing the computation times of the Dijkstra algorithm with $2*N^2$, the algorithm's computation complexity has been effectively improved. The efficiency of the proposed method is compared with Dijkstra as expressed below, and it can be concluded that the maximum efficiency can reach 15 when the node number N tends to positive infinity theoretically.

$$\eta = \frac{(4+X)*N}{2*N^2} = \frac{4+X}{2*N} \quad (1)$$

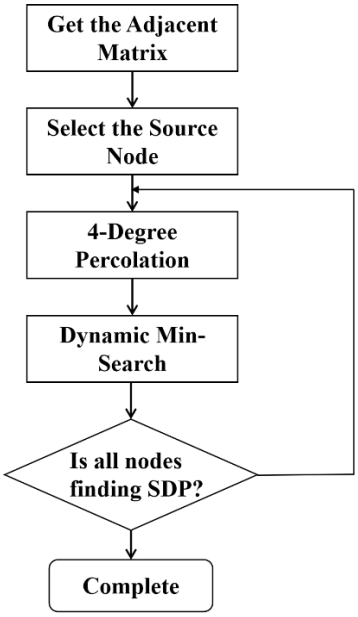

Fig. 4. Flowchart of the proposed method.

IV. Experiments

The topology of the mega-constellation backbone network is similar to that of the mesh grid, and the network scale can be expanded by increasing the number of constellation orbits. This paper increases the constellation scale by extending the orbit number from 3 to 36 with 18 satellites per orbit. The proposed algorithm, Dijkstra algorithm, and heap-optimized Dijkstra algorithm were compared to verify the effectiveness of the proposed method. The experimental setup involved an 11th Gen Intel(R) Core (TM) i5-11300H @ 3.10GHz and 16GB RAM. incorporated in an Icore4T embedded edge computing platform. All algorithms were implemented in C/C++ , and the corresponding results are illustrated in Fig. 5. The results highlight that the proposed algorithm has dramatically improved with the increase in the constellation scale compared to the Dijkstra algorithm. Fig. 6 compares the proposed method and the heap-optimized Dijkstra algorithm as a ratio of their calculation time. The proposed method outperforms when the network scale is less than 334. When the network scale exceeds 334, the calculation time of the proposed method is a little longer and stabilizes as the network scale increases.

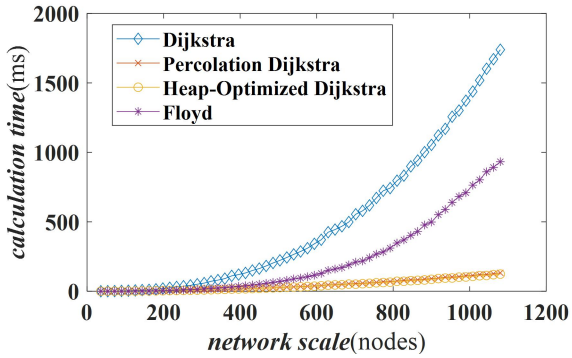

Fig. 5. Calculation time of algorithms with increasing network scale by 18 satellites per orbit.

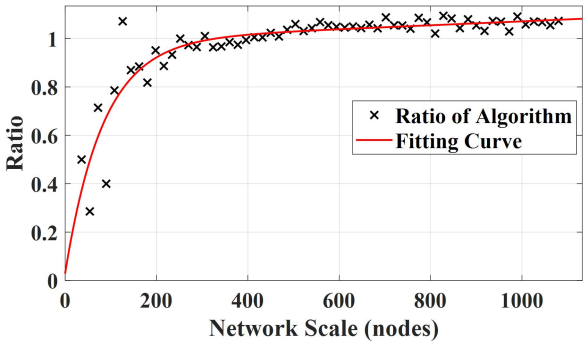

Fig. 6. The ratio of the calculation time between the proposed method and the heap-optimized Dijkstra algorithm.

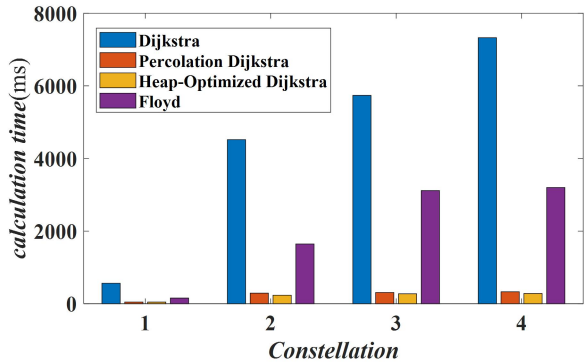

Fig. 7. Calculation time of typical constellation topology. 1 for Oneweb, 2 for Kuiper, 3 for Starlink-a layer, and 4 for Starlink-b layer.

We also conducted experiments on typical constellation topologies, such as Oneweb (18x36), Kuiper (34x34), Starlink-a (72x22), and Starlink-b (24x66). Fig. 7 reveals that the proposed method can perform the same as the heap-optimized Dijkstra algorithm while avoiding the massive memory access and space with a heap or prior queue data structure. In practice, no large network scale usually reaches nearly 1000 nodes for communication networks. The proposed network can performs best within 334 nodes, depicted in Fig. 6.

Experiments on the onboard computing platform with SPARC v8 CPU @133MHz with no operation system environment verified the effectiveness of the proposed method on the embedded devices. The results in Fig. 8 illustrate that the proposed method is consistent with the analysis in Section III and can significantly promote real-time performance without introducing complex data structures.

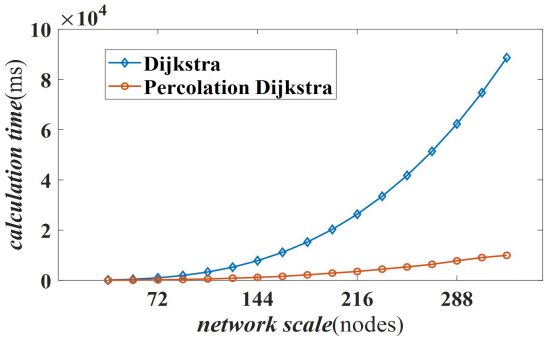

Fig. 8. The calculation time on onboard device.

V. Conclusion

This article proposes a fast SPF algorithm to accelerate the routing calculation of a large-scale backbone network of a mega-constellation. Specifically, a 4-degree percolation algorithm was designed based on the mega-constellation's sparsity and regularity characteristics. Additionally, a dynamic minimum search algorithm was proposed to limit the range of traversal calculations. The experimental results show that compared to conventional algorithms, the proposed method can significantly reduce routing computation time and is suitable for satellite computers with limited computing resources.

References

[1] R. De Gaudenzi, M. Luise and L. Sanguinetti, "The Open Challenge of Integrating Satellites into (Beyond-) 5G Cellular Networks," in IEEE Network, vol. 36, no. 2, pp. 168-174, March/April 2022, doi: 10.1109/MNET.011.2100116.

[2] Y. Hu, X. Wang and W. Saad, "Distributed and Distribution-Robust Meta Reinforcement Learning (D ^{2} -RMRL) for Data Pre-Storage and Routing in Cube Satellite Networks," in IEEE Journal of Selected Topics in Signal Processing, vol. 17, no. 1, pp. 128-141, Jan. 2023, doi: 10.1109/JSTSP.2022.3232944.

[3] J. Shuai, Y. Liu and Y. Wang, "Energy Efficient Maximal Throughput Resource Scheduling Strategy in Satellite Networks," in IEEE Wireless Communications Letters, vol. 12, no. 2, pp. 312-316, Feb. 2023, doi: 10.1109/LWC.2022.3224770.

[4] A. U. Chaudhry and H. Yanikomeroglu, "When to Crossover From Earth to Space for Lower Latency Data Communications?," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 3962-3978, Oct. 2022, doi: 10.1109/TAES.2022.3156087.

[5] Z. Lin et al., "Systematic Utilization Analysis of Mega-Constellation Networks," *2022 International Wireless Communications and Mobile Computing (IWCMC)*, Dubrovnik, Croatia, 2022, pp. 1317-1322, doi: 10.1109/IWCMC55113.2022.9824193.

[6] X. Qin, T. Ma, Z. Tang, X. Zhang, H. Zhou and L. Zhao, "Service-Aware Resource Orchestration in Ultra-Dense LEO Satellite-Terrestrial Integrated 6G: A Service Function Chain Approach," in *IEEE Transactions on Wireless Communications*, vol. 22, no. 9, pp. 6003-6017, Sept. 2023, doi: 10.1109/TWC.2023.3239080.

[7] Eramo V , Listanti M , Caione N ,et al. “Optimization in the Shortest Path First Computation for the Routing Software GNU Zebra,” *IEICE Transactions on Communications*, 2005.

[8] S. Zhang and K. L. Yeung, "Scalable routing in low-earth orbit satellite constellations: Architecture and algorithms", *Comput. Commun.*, vol. 188, pp. 26-38, Apr. 2022.

[9] T. Ma, B. Qian, X. Qin, X. Liu, H. Zhou and L. Zhao, "Satellite-Terrestrial Integrated 6G: An Ultra-Dense LEO Networking Management Architecture," in *IEEE Wireless Communications*, vol. 31, no. 1, pp. 62-69, February 2024, doi: 10.1109/MWC.011.2200198.

[10] M. Hozayen, T. Darwish, G. K. Kurt and H. Yanikomeroglu, "A Graph-Based Customizable Handover Framework for LEO Satellite Networks," *2022 IEEE Globecom Workshops (GC Wkshps)*, Rio de Janeiro, Brazil, 2022, pp. 868-873, doi: 10.1109/GCWkshps56602.2022.10008514.

[11] Q. Chen, L. Yang, Y. Zhao, Y. Wang, H. Zhou and X. Chen, "Shortest Path in LEO Satellite Constellation Networks: An Explicit Analytic Approach," in *IEEE Journal on Selected Areas in Communications*, doi: 10.1109/JSAC.2024.3365873.

[12] Q. Chen, G. Giambene, L. Yang, C. Fan and X. Chen, "Analysis of Inter-Satellite Link Paths for LEO Mega-Constellation Networks," in *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2743-2755, March 2021, doi: 10.1109/TVT.2021.3058126.

[13] T. Pan, T. Huang, X. Li, Y. Chen, W. Xue, and Y. Liu, “OPSPF: Orbit Prediction Shortest Path First Routing for Resilient LEO Satellite Networks,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[14] J. Li, H. Lu, K. Xue, and Y. Zhang, “Temporal Netgrid ModelBased Dynamic Routing in Large-Scale Small Satellite Networks,” in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6009–6021, Jun. 2019.