WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 1, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n1.10
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Research on Adaptive Routing Algorithm for Low Earth Orbit Satellite Networks Based on Particle Swarm Optimization

Guanhong Chen, Zexin Li *, Chengjin Zhou, Yizhen Chen

School of Computer Science and Software, Zhaoqing University, Zhaoqing, Guangdong, 526061, China

*Corresponding Author: Zexin Li

ABSTRACT

Aiming at the problems of high dynamic topology and limited on-board resources in Low Earth Orbit (LEO) satellite networks leading to poor performance of traditional routing algorithms, this paper combines the centralized control idea of Software-Defined Networking (SDN) with the collective learning ability of the Particle Swarm Optimization (PSO) algorithm to propose an adaptive potential field routing algorithm based on PSO. This algorithm models the network as a dynamic potential field, defines potential energy based on the cumulative congestion degree to guide data packet transmission; adaptively learns and optimizes link cost weights through the PSO mechanism, achieving dynamic evolution of the path evaluation criterion; and adopts a probability-based multi-path forwarding mechanism to achieve load balancing. Simulation results show that compared to traditional algorithms such as Dijkstra and ACO (Ant Colony Optimization), this algorithm significantly improves performance in terms of average end-to-end delay, network throughput, and load balancing, especially suitable for dynamic traffic scenarios.

KEYWORDS

LEO Satellite Networks; Particle Swarm Optimization; Adaptive Routing; Load Balancing

1. INTRODUCTION

With the rapid development of the 6G integrated space-air-ground information network, Low Earth Orbit (LEO) satellite networks have become a key component in building global seamless connectivity due to their advantages such as wide-area coverage and low transmission delay [1]. However, characteristics like the high dynamicity of the network topology caused by the high-speed movement of satellite nodes, limited on-board computing and storage resources, and frequent changes in link state pose severe challenges to the flexibility, adaptability, and efficiency of routing algorithms [2].

Traditional routing algorithms such as Dijkstra's algorithm, which typically optimize based on a single metric (e.g., hop count or delay), struggle to adapt to the dynamic characteristics of satellite networks and effectively avoid congestion [3]. Bio-inspired optimization algorithms like the Ant Colony Algorithm (ACO) possess certain adaptive capabilities but suffer from issues such as slow convergence speed and susceptibility to local optima [4]. The introduction of the Software-Defined Networking (SDN) architecture, through the separation of control and data planes, provides a global view and centralized control capability for satellite networks [5, 6]. For instance, the minimum hop count routing algorithm proposed in [6] effectively improves routing efficiency, but its path weights are mostly statically set, lacking fine-grained perception and learning capabilities for dynamic services and network states.

To address these issues, this paper proposes an adaptive potential field routing algorithm based on Particle Swarm Optimization (PSO). This algorithm innovatively combines the concept of a physical potential field with swarm intelligence optimization. It constructs a dynamic potential field to reflect the network congestion state and utilizes the PSO mechanism to enable the routing strategy to possess continuous learning and evolutionary capabilities, thereby achieving efficient and reliable data transmission in complex and dynamic satellite network environments [7, 8].

2. SYSTEM MODEL

This paper adopts an SDN-based GEO/LEO two-layer satellite network architecture, as shown in Figure 1. The ground control center acts as the master controller, GEO satellites act as slave controllers forming the control layer, and the LEO satellite network constitutes the data forwarding layer. The control layer collects network status through satellite-ground/inter-satellite links and distributes routing decisions.

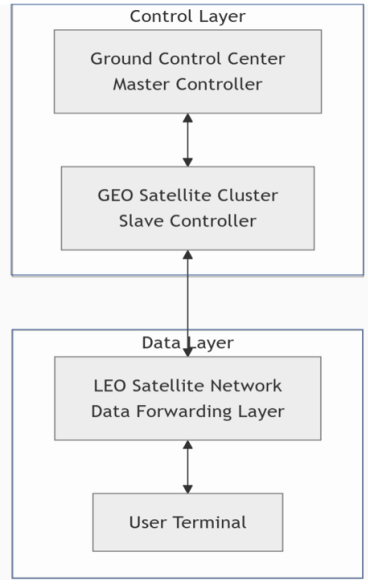

Figure 1. Schematic diagram of the GEO/LEO two-layer satellite network model

This model mainly includes:

Data Forwarding Layer: Composed of the LEO satellite constellation, responsible for receiving and forwarding data packets.

Control Layer: Composed of the GEO satellite cluster and the ground control center. GEO satellites act as slave controllers, responsible for collecting status information of LEO satellites within their coverage area; the ground control center acts as the master controller, maintaining the global network view G=(V, E), where V is the set of nodes and E is the set of link attributes (delay, bandwidth, packet loss rate) [9].

Control Interaction: The control layer delivers flow tables to the data layer through protocols like OpenFlow to guide data forwarding [10].

3. THE PROPOSED ROUTING ALGORITHM BASED

The core idea of this algorithm is to perceive global congestion through a dynamic potential field, use Particle Swarm Optimization to adaptively adjust the path selection strategy, and finally achieve load balancing through probability forwarding. The general overview is as follows: The algorithm defines each data flow as a "particle". This particle dynamically evolves its weight vector Wp=[αp,

βp, γp] used to evaluate link quality by continuously learning from the individual historical best experience and the global best experience of the swarm. Nodes calculate link cost based on the dynamically computed "traffic potential" and the evolved weight vector, and select the next hop based on probability, thereby achieving intelligent routing with low delay, high throughput, and strong load balancing capability [11].

3.1. Traffic Potential

The traffic potential Φid is defined as the minimum cumulative cost from the current node i to the destination node d, dynamically reflecting the congestion degree of the path. It is calculated as follows:

$$\phi_i^d = \min_{j \in N(i)} \left(\text{Cost}(i,j) + \phi_j^d \right) \tag{1}$$

Where N(i) is the set of neighbors of node i, and Cost(i, j) is the link cost from node i to neighbor j. The potential calculation process is shown in Figure 2. Data packets always flow from nodes with higher potential to nodes with lower potential [12]. The figure shows the potential distribution from source node S to destination node D, with arrows indicating the forwarding direction of data packets based on the potential gradient (from high to low).

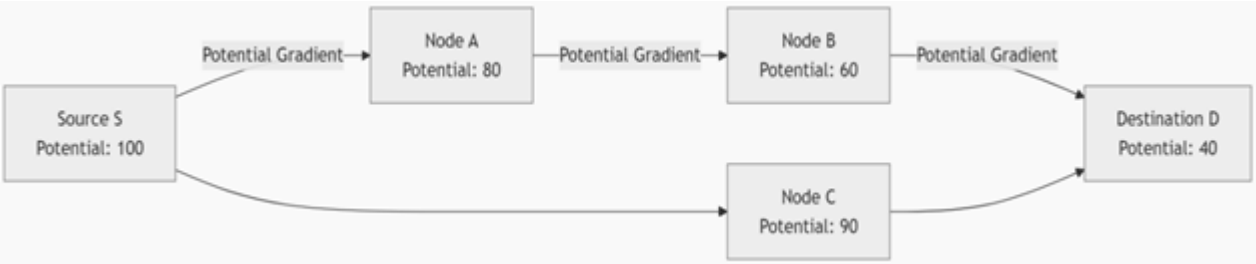

Figure 2. Schematic diagram of traffic potential calculation and data packet flow direction

The figure shows the potential distribution from source node S to destination node D, with arrows indicating the forwarding direction of data packets based on the potential gradient (from high to low).

3.2. Adaptive Link Cost Calculation

The link cost Cost(i,j) is key to the algorithm, comprehensively reflecting the real-time state of the link:

$$Cost_p(i,j) = \alpha_p \cdot \frac{Q_j}{Q_{max}} + \beta_p \cdot \frac{U_{ij}}{U_{max}} + \gamma_p \cdot Hops_{ij} \tag{2}$$

Where:

Qj: The current queue length of neighbor node j.

Q_{max}: The maximum capacity of the node queue.

U_{ij}: The recent utilization of link i -> j.

U_{max}: The maximum theoretical utilization of the link.

Hops_{ij}: A constant, usually 1 (representing one hop), or can be fine-tuned based on link delay.

α, β, γ: Weight coefficients used to adjust the importance of queue congestion, link utilization, and basic hop count in the decision. To achieve strong load balancing, the weights of α and β should be set larger than that of γ [13].

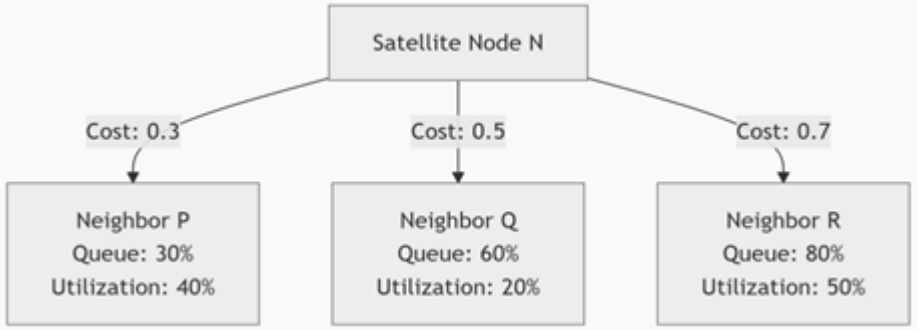

Figure 3. Schematic diagram of the adaptive link cost calculation model

The illustration shows how a satellite node integrates the queue length of its neighbor nodes, link utilization, and hop count to calculate the link cost in different directions.

3.3. Particle Swarm Optimization Mechanism

Each persistent data flow is regarded as a particle p, whose core is to maintain a weight vector for evaluating link cost:

$$\vec{W}_p = (\alpha_p, \beta_p, \gamma_p) \tag{3}$$

The PSO mechanism periodically updates this vector through the following formulas, achieving learning and evolution:

$$\begin{aligned} \vec{V}_p(t+1) &= \omega \cdot \vec{V}_p(t) + c_1 \cdot r_1 \cdot (\vec{P_{best_p}} - \vec{W}_p(t)) + c_2 \cdot r_2 \cdot (\vec{G_{best}} - \vec{W}_p(t)) \\ \vec{W}_p(t+1) &= \vec{W}_p(t) + \vec{V}_p(t+1) \end{aligned} \tag{4}$$

Vp: The velocity vector of weight change.

ω: Inertia weight, balancing global and local search capabilities.

c1, c2: Learning factors, representing the degree of individual and social learning, respectively.

r1, r2: Random numbers in the range [0, 1].

Pbest: The particle's individual historical best weight.

Gbest: The global historical best weight, calculated by the control center after collecting the performance of all particles and broadcasted.

The fitness function of particle p is used to evaluate the quality of its weight vector:

$$Fp = -(Ap + \delta \cdot Lp) \tag{5}$$

Where Ap is the average delay, Lp is the variance of the queue lengths of the nodes along the path, and δ is a balance coefficient. A higher fitness value indicates better performance of the weight vector.

3.4. Probability-based Multi-path Forwarding

To avoid single-path congestion and fully utilize network resources, nodes adopt a probability forwarding mechanism. For each neighbor j, the probability Pj of being selected as the next hop is calculated as follows:

Calculate forwarding weight:

$$W_j = \frac{1}{Cost(i,j) + \Phi_j^d + \epsilon} \tag{6}$$

(ϵ is a very small positive number to prevent division by zero errors.)

Normalize to probability:

$$P_j = \frac{W_j}{\sum_{k \in N(i)} W_k} \tag{7}$$

This mechanism ensures that neighbors with lower total cost (link cost + neighbor potential) have a higher probability of being selected, thereby achieving multi-path traffic distribution and load balancing while pursuing the optimal path.

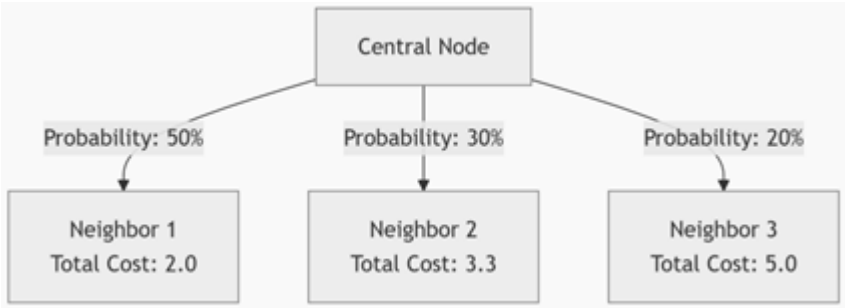

Figure 4. Schematic diagram of probability-based multi-path forwarding

The central node in the figure has multiple neighbors; the thickness of the arrows represents the probability of being selected as the next hop, visually demonstrating how probability forwarding disperses traffic.

4. ROUTING STRATEGY PROCESS

The overall execution flow of the proposed algorithm is shown in Figure 5, clearly illustrating the interaction and execution sequence of the three core modules: distributed potential update, PSO weight learning, and probability forwarding. The specific steps are as follows:

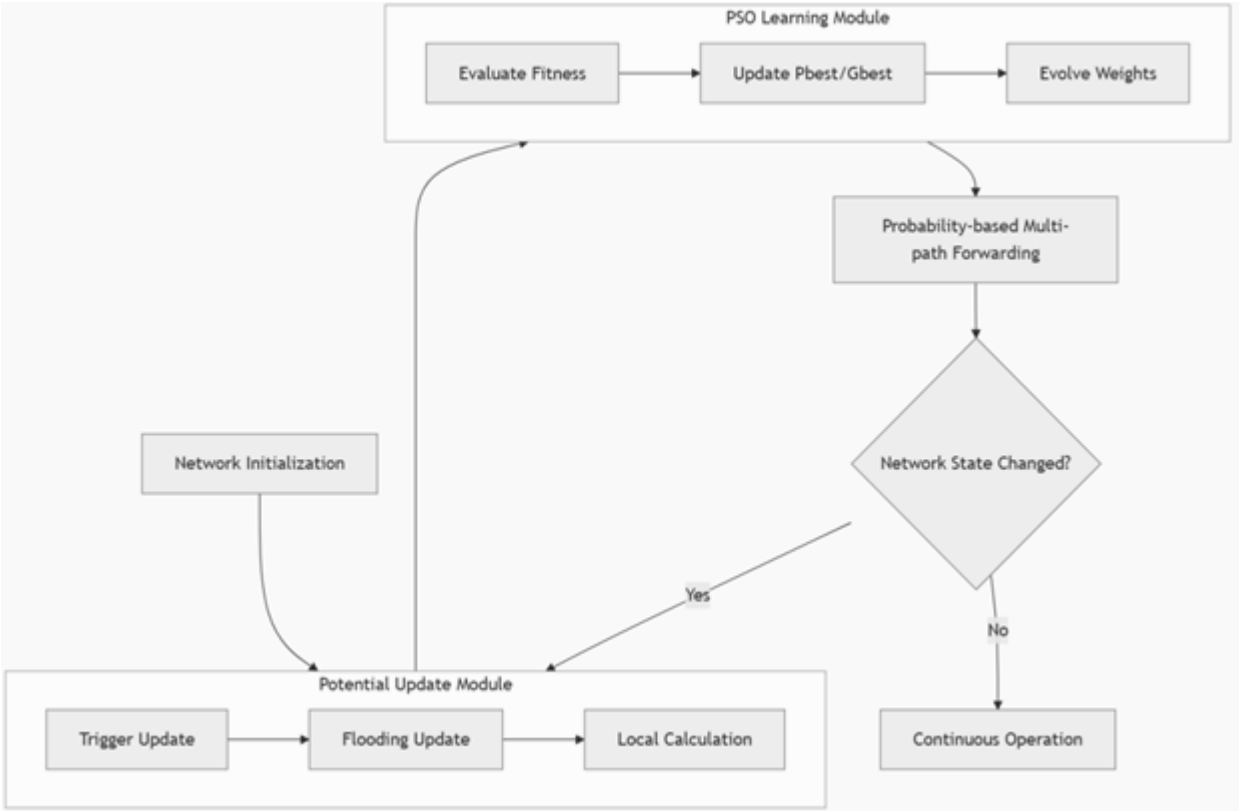

Figure 5. Overall algorithm execution flowchart

(1) Initialization and Information Collection:

LEO satellite nodes establish links with their affiliated GEO controllers and report their own status (queue length, link utilization, etc.). The control center forms a global network view G(V, E).

(2) Distributed Potential Table Calculation and Update:

Triggered Update: When the link cost Cost(i,j) or the node's own queue length changes beyond a threshold, an update is triggered.

Flooding Update: The node floods a Link State Advertisement (LSA) to its neighbors, containing its updated potential Φid.

Local Calculation: Neighbor node k, upon receiving the LSA, recalculates its own potential according to the formula

$$\Phi_k^{d(new)} = \min_{j \in N(k)} \left(Cost(k,j) + \Phi_j^d \right) \tag{8}$$

If the potential changes, it continues to trigger flooding updates until the network stabilizes.

(3) Particle Swarm Optimization:

After a data flow (particle) transmission is completed, its fitness Fp is calculated based on the transmission delay Ap and node queue variance Lp.

Update the particle's individual historical best Pbest.

The control center collects all particles' Pbest, calculates the current global best Gbest, and broadcasts it to the network.

Each particle updates its own weight vector Wp according to the PSO formula.

(4) Routing Request and Decision:

When an LEO satellite receives a data packet and the flow table entry is missing, it sends a request to the control plane. The control plane, combining the latest potential information and the evolved weight vector corresponding to the requesting data flow, calculates the cost of multiple paths from the source to the destination.

(5) Probability-based Multi-path Forwarding:

The control plane delivers the calculated probabilities for each path to the source node. The source node and intermediate nodes along the path randomly select the next hop for data packet forwarding according to the probability distribution Pj.

(6) Route Update and Maintenance:

GEO satellites continuously monitor the LEO network status, triggering new rounds of potential updates and PSO learning, enabling the routing strategy to continuously adapt to network changes.

5. SIMULATION ANALYSIS

5.1. Algorithm Complexity Analysis

The time complexity of this algorithm mainly comes from potential updates and PSO optimization.

Potential Update: Triggered flooding is used. In the worst case, the complexity is O(|V|·|E|), but in practice, due to local updates, the average complexity is lower.

PSO Optimization: The complexity is related to the number of particles m, the number of iterations T, and the weight dimension D (3 in this paper), which is O(m·T·D). Since m and T are controllable and the calculation is performed on the control plane, the burden on the satellites is small.

A comparison of the computation time of the proposed algorithm with Dijkstra and ACO algorithms as the number of nodes increases is shown in Table 1:

Table 1. Algorithm Time Complexity Comparison Table

Algorithm Name	Time Complexity	Description
Proposed Algorithm	O(|V|*|E| + O(m*t T*t D)	Potential Update (Worst-case) + PSO Optimization
Dijkstra Algorithm	O(|E| + |V|log|V|	Priority Queue-based Implementation
ACO Algorithm	O(m* T*|V|^2	Ant Count × Iterations × Node Count^2

5.2. Simulation Parameter Settings

Using STK and MATLAB for co-simulation, the simulation experiment adopts an SDN-based GEO/LEO two-layer satellite network architecture. The main parameter settings are as follows:

The LEO satellite orbit altitude is 800 km, with 6 orbital planes in total, each deploying 11 satellites, totaling 66 satellites. The orbital inclination is 86.4°, and each satellite establishes 4 inter-satellite links.

The GEO satellite orbit altitude is 35786 km, composed of 3 satellites, with an orbital inclination of 0°, and each satellite establishes 2 inter-satellite links.

The network link bandwidth is set to 2 Mbps, and the node queue capacity is 40 data packets. The parameters for the Particle Swarm Optimization algorithm are set as: number of particles 50, number of iterations 100. These parameters provide a complete simulation environment basis for evaluating algorithm performance.

Three service types are set, and their initial weight factors are shown in Table 2 (initial values, subsequently adjusted by PSO):

Table 2. Service Types and Initial Weight Factors

Service Type	Description	α (Queue)	β (Utilization)	γ (Hops)
A	Low-latency service	0.60	0.20	0.20
B	High-throughput Service	0.20	0.60	0.20
C	High-reliability Service	0.33	0.33	0.33

5.3. Performance Analysis of Simulation Results

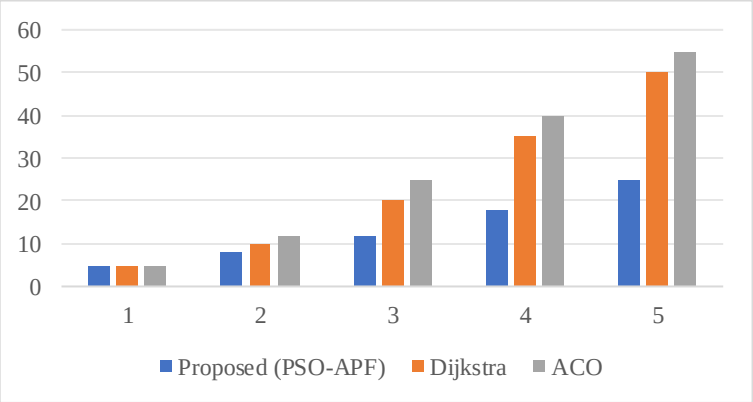

Figure 6. Average End-to-End Delay Comparison

Average End-to-End Delay: As shown in Figure 6, under low load, the delays of all algorithms are similar. As the data sending rate increases and network congestion occurs, the proposed algorithm, due to its ability to dynamically perceive congestion and avoid high-delay paths through PSO learning, and with the help of the GEO layer for long-distance detours, its average delay is significantly lower than that of the Dijkstra and ACO algorithms.

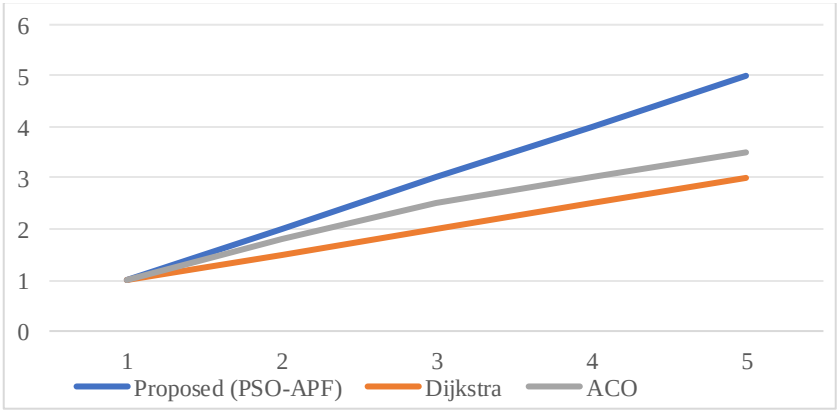

Figure7. Network Throughput Comparison

Network Throughput: As shown in Figure 7, through effective multi-path load balancing, the proposed algorithm avoids the formation of network bottlenecks, allowing the overall network throughput to grow steadily with increasing load and consistently remain higher than that of the comparison algorithms.

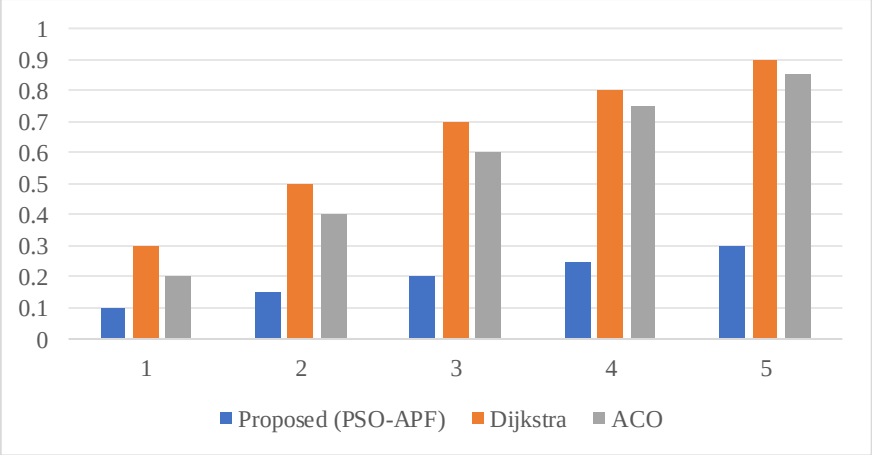

Figure 8. Load Balancing Comparison

Load Balancing: As shown in Figure 8, using the variance of node queue lengths as the measurement indicator. The curve of the proposed algorithm is the flattest and has the lowest value, proving its effectiveness in dispersing traffic to various nodes and avoiding local congestion.

6. CONCLUSION

Aiming at the high dynamic and resource-constrained characteristics of Low Earth Orbit (LEO) satellite networks, this paper proposed an adaptive potential field routing algorithm based on Particle Swarm Optimization (PSO). This algorithm constructs a dynamic potential field to realistically reflect the network congestion state, introduces a PSO mechanism to enable the routing strategy to possess continuous learning and collective evolution capabilities, and finally achieves efficient load balancing through probability forwarding. Simulation results show that this algorithm outperforms traditional routing algorithms in key performance metrics such as delay, throughput, load balancing, and packet loss rate, demonstrating good adaptability and robustness.

Future research work will focus on: 1) Introducing more QoS metrics such as energy consumption and link lifetime into the fitness function; 2) Validating the algorithm on more realistic satellite network simulation platforms (e.g., NS3) or testbeds; 3) Exploring integration with advanced AI methods such as deep learning to further enhance the intelligence level and predictive capability of routing decisions.

ACKNOWLEDGEMENTS

This work was supported by the Innovation Training Program of Zhaoqing University under Grant X202510580107 and the Provincial Innovation Training Program under Grant S202510580055.

REFERENCES

[1] Smith J, Wang L. "A Survey on LEO Satellite Networks for Global Connectivity," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1227--1269, 2020.

[2] Johnson M, Brown K. "SDN-Based Architecture for Satellite Networks," International Journal of Satellite Communications and Networking, vol. 37, no. 5, pp. 487--502, 2019.

[3] Lee S, Kim H. "Dynamic Routing in LEO Satellite Networks Using Machine Learning," IEEE Transactions on Aerospace and Electronic Systems, vol. 57, no. 4, pp. 2431--2445, 2021.

[4] Zhang Y, Liu Q. "Particle Swarm Optimization for Wireless Sensor Networks Routing," Journal of Network and Computer Applications, vol. 112, pp. 1--15, 2018.

[5] Chen X, Li W. "Load Balancing in Software-Defined Networks: A Review," Computer Networks, vol. 215, p. 109281, 2022.

[6] Kennedy J, Eberhart R. "Particle Swarm Optimization," in Proceedings of the International Conference on Neural Networks, 1995, pp. 1942--1948.

[7] Anderson T, White S. "Adaptive Routing Algorithms for Dynamic Networks," IEEE/ACM Transactions on Networking, vol. 28, no. 3, pp. 1325--1338, 2020.

[8] Roberts M, Green P. "QoS-Aware Routing in Satellite Networks," International Journal of Communication Systems, vol. 32, no. 14, p. e3742, 2019.

[9] Turner J, King B. "Multi-Path Routing for Load Balancing in Computer Networks," Computer Communications, vol. 127, pp. 1--12, 2018.

[10] Evans R, Scott D. "Energy-Efficient Routing in Satellite Networks," IEEE Wireless Communications Letters, vol. 9, no. 8, pp. 1234--1237, 2020.

[11] Hill J, Adams N. "Software-Defined Networking for Space Communications," IEEE Communications Magazine, vol. 59, no. 5, pp. 112--118, 2021.

[12] Morgan K, Baker T. "Particle Swarm Optimization: Advances and Applications," Swarm Intelligence, vol. 12, no. 3, pp. 211--234, 2018.

[13] Cook L, Bell M. "Deep Learning for Predictive Routing in Satellite Networks," Neural Networks, vol. 145, pp. 256--268, 2022.