DRL-Based Service Function Chains Embedding Through Network Function Virtualization in STINs

1st Li Li
School of Information and Electronics
Beijing Institute of Technology
Beijing, China
lililee3397@163.com

2nd Chuhong Yang
School of Information and Electronics
Beijing Institute of Technology
Beijing, China
3120230733@bit.edu.cn

3rd Haoyang Li
School of Information and Electronics
Beijing Institute of Technology
Beijing, China
3120225395@bit.edu.cn

4th Dongxuan He
School of Information and Electronics
Beijing Institute of Technology
Beijing, China
dongxuan_he@bit.edu.cn

2024 IEEE International Conference on Signal, Information and Data Processing (ICSIDP) | 979-8-3315-1566-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSIDP62679.2024.10868090

***Abstract*—Space-terrestrial integrated networks (STINs) are gaining increasing attention for their outstanding benefits in providing seamless connectivity, enhancing network resilience, increasing capacity, and expanding coverage. The combination of network function virtualization (NFV) and deep reinforcement learning (DRL) is a promising candidate to boost the capability of STINs to deliver high-quality services. In this paper, a STIN architecture based on artificial intelligence (AI) algorithms and the process of embedding Service Function Chains (SFCs) into highly dynamic STINs are studied. Then, we propose an enhanced DRL-based SFCs embedding algorithm that initiates with a feature fusion phase of links, followed by aggregating the feature vectors of all nodes and their neighboring nodes to form context information. This context information is then used as part of the input state for the DRL-based SFCs embedding algorithm. Simulation results show that the proposed algorithm can effectively reduce the latency of SFCs embedding when compared with other existing algorithms, thus showing the superiority of our proposed algorithm.**

Keywords—space-terrestrial integrated network, network function virtualization, service function chains embedding, deep reinforcement learning

I. INTRODUCTION

Traditional terrestrial networks often struggle to provide long-distance global communication coverage, particularly in challenging terrains like mountains and oceans. Furthermore, natural disasters frequently destroy these terrestrial network infrastructures, leading to disruptions in end-to-end communication. To address these challenges, space-terrestrial integrated networks (STINs) have been developed. Offering significant benefits such as seamless connectivity, improved network resilience, increased capacity, and extended coverage, STINs are emerging as a compelling solution for next-generation mobile communications [1-4].

The service function chains (SFCs) embedding optimizes network resource utilization, reduces operational complexity, and thus, improves network flexibility and service delivery efficiency, enhancing the STINs performance significantly [5]. One of the key technologies enabling SFCs embedding in STINs is network function virtualization (NFV), which allows network functions to be run on virtual machines instead of dedicated hardware [6]. The use of NFV in STINs brings many benefits. It enables dynamic and flexible allocation of network resources, allowing for efficient use of network capacity and reducing the need for dedicated hardware.

Due to the motion of satellites, the topology of STINs are highly dynamic, thus making the optimization of SFCs embedding intractable. Typically, the SFCs embedding problem can be formulated as a Markov decision process (MDP). Accordingly, deep reinforcement learning (DRL) technology seems to be a suitable method to solve such problem [7-9]. Moreover, since the satellite trajectories are typically pre-designed, referred to as satellite ephemeris, DRL can solve service function chains embedding problem in STINs efficiently. More specifically, DRL learns the STINs' pattern and makes the optimal SFCs embedding policy automatically, leading to improved network performance. Furthermore, DRL has excellent generalization ability in dynamic environment, thus making it extremely suitable to STINs.

Against this background, a DRL-based service function chains embedding method in STINs is proposed. The main contributions of this paper are as follows. We propose the AI-enhanced STINs architecture, which can implement AI algorithms to tackle the network control problem. Then, we model the SFCs embedding process in highly dynamic STINs, which is unprecedented, to the best of our knowledge. Finally, to reduce the latency of SFCs embedding, feature encoding combined with the Proximal Policy Optimization (PPO) algorithm is proposed [10]. Simulation results show that the proposed DRL-based SFCs embedding algorithm outperforms the PPO algorithm and the Deep Q-Network (DQN) algorithm significantly.

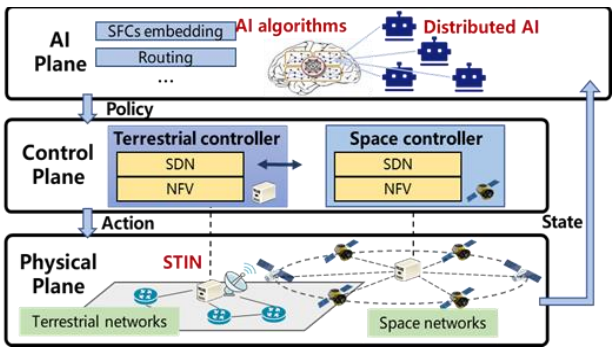

Fig. 1 The diagram of AI-enhanced STINs architecture

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on March 01,2025 at 12:42:52 UTC from IEEE Xplore. Restrictions apply.

II. SYSTEM MODEL

A. AI-enhanced STINs architecture

The proposed AI-enhanced STINs architecture is shown in Fig. 1, which consists of three planes, namely physical plane, control plane, and AI plane.

1) The physical plane represents the substrate network, including both terrestrial networks, such as various ground communication devices and infrastructure, and space networks, such as satellites and other space communication devices. These elements are the basis of STINs, responsible for forwarding, processing, and monitoring data packets.

2) The control plane consists of terrestrial and space controllers that manage their respective networks using software defined network (SDN) and NFV technologies. In particular, SDN enables centralized control of network traffic, while NFV facilitates flexible deployment and management of network functions through virtualization.

3) The AI plane, based on global network states gathered from the physical plane and user demands, can utilize AI algorithms to obtain the optimal network control policies, Typically, distributed AI can be used to boost overall network performance and efficiency through the collaborative efforts of multiple AI algorithms and models. As a result, the key functions of STINs, such as SFCs embedding and routing, can be optimized by AI algorithms accordingly.

These three layers are interconnected via southbound interfaces like OpenFlow, forming an integrated network control and management system. Specifically, the physical plane handles data packet forwarding, processing, and monitoring, the control plane converts AI-derived policies into specific control rules, and the AI plane iteratively learns optimal control policies by interacting with the network.

This tri-plane architecture marks a significant advancement in network management, bringing together the strengths of AI, SDN, and NFV technologies. By harnessing the power of AI for network control, this approach promises to revolutionize the efficiency and adaptability of STINs.

B. Model of SFCs embedding in STINs

SFCs embedding is elaborated through a sample example in Fig. 2. virtual services are designed and then mapped to physical network as SFCs. Considering an STINs consisting of a satellite set N^{sa} and a terrestrial station set N^{te}. The mathematical model of the substrate STINs at snapshot t can be characterized as a weighted directed graph $G_t^S = \{N^S, E_t^S, A^S, D_t^S, W^S\}$ in which $N^S = N^{sa} \cup N^{te}$ denotes the set of substrate nodes, $E_t^S = \{(n^S, m^S) | n^S \in N^S, m^S \in \mathcal{N}(n^S)\}$ represents the set of substrate links connecting the substrate nodes with $\mathcal{N}(n^S)$ denoting the set of substrate nodes connecting to n^S, $A^S = \{a_{n,r}^S \in \{0, \mathbb{R}^+\} | n^S \in N^S, r \in R\}$ is the set of resource capacity of the substrate nodes with R denoting the set of resources, such as CPU and memory, $D_t^S = \{d_{t,n,m}^S \in R^+ | (n^S, m^S) \in E_t^S\}$ is the set of distance of the substrate links, $W^S = \{w_{n,m}^S \in \{0, \mathbb{R}^+\} | n, m \in N^S\}$ is the set of bandwidth assigned to the substrate nodes to transmit packet to the other.

Similarly, the mathematical model of the SFCs can be characterized as a weighted directed graph $G^V = \{N^V, E^V, A^V, B^V, W^V\}$, where N^V represents the set of virtual nodes characterizing the virtual network functions, $E^V = \{(n^V, m^V) | n^V \in N^V, m^V \in \mathcal{N}(n^V)\}$ represents the set of virtual links connecting the virtual nodes with $\mathcal{N}(n^V)$ denoting the set of substrate nodes connecting to n^V, $A^V = \{a_{n,r}^V \in \{0, \mathbb{R}^+\} | n^V \in N^V, r \in R\}$ is the set of demanded resource of the virtual nodes, $B^V = \{b_{n,m}^V \in \{0, \mathbb{R}^+\} | (n^V, m^V) \in E^V\}$ is the set of bit amount of the packets transmitted along the virtual links, $W^V = \{w_{n,m}^V \in \{0, \mathbb{R}^+\} | (n^V, m^V) \in E^V\}$ is the set of demanded bandwidth of the virtual links.

C. Model of SFCs queue

Consider a set of SFCs $H = \{h_k\}$. Here, H represents a set of SFCs, and h_k denotes the k-th specific SFC in the set. In STINs, the asynchronous SFC arrivals and departures are modeled as follows. The k -th SFC h_k needs to be implemented with the time interval $[t_k^{\mathrm{ar}}, t_k^{\mathrm{de}}]$, where t_k^{ar} is the arrival time and t_k^{de} is departure time. The duration of h_k is denoted by $\tau_k = t_k^{\mathrm{de}} - t_k^{\mathrm{ar}}$. Without loss of generality, we assume that $t_1^{\mathrm{ar}} \le t_2^{\mathrm{ar}} \le \cdots \le t_K^{\mathrm{ar}}$ and $t_1^{\mathrm{ar}} = 0$. The SFC embedding and resetting process are triggered when a SFC arrives or an embedded SFC departures. Upon the arrival of a SFC, i.e., $t = t_k^{\mathrm{ar}}$, the AI plane has to decide whether to accept the SFC. If the SFC is accepted, the substrate network must allocate the demanded substrate network resource to the SFC according to the embedding policy released by AI plane. Then, h_k is added to the embedded SFC set $\bar{H}$. Similarly, upon the termination of an embedded SFC, i.e., $t = t_k^{\mathrm{de}}$ with the $h_k \in \bar{S}$, the allocated substrate network resources are released.

D. Network metrics

Since the STINs is dynamic due to the motion of satellites and the process of virtual service function chains embedding, the network metrics can be defined as follows, which are variant with respect to time t.

- Channel capacity

Given the demanded bandwidth $w_{n,m}^V$ of the virtual link (n^V, m^V), the variant channel capacity of the substrate link (n^S, m^S) at snapshot t is defined as

$$c_{t,n,m}^S = w_{n,m}^V log_2(1 + \frac{\eta p_{n,m} (d_{t,n,m}^S)^{-2}}{\sigma^2}) \qquad (1)$$

where η and $p_{n,m}$ denote the channel gain and the transmit power assigned to the substrate link, respectively.

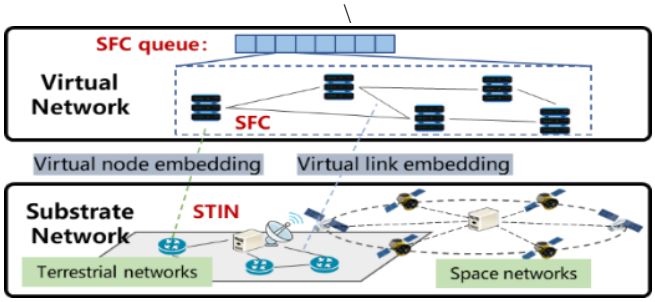

Fig. 2 Model of SFCs embedding in STINs

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on March 01,2025 at 12:42:52 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 DRL-based SFCs embedding process

Input: The state of substrate graph $\mathcal{S}$, The node set of the substrate graph $\mathcal{N}_s$, a series of SFC request $\mathcal{R} = \{r_i = (\mathcal{G}_{v_i}, h_i, t_{s_i}, t_k^{ar}, t_k^{de}) | i = 1, 2, 3, \dots, N, t_{s_1} < t_{s_2} < \cdots < t_{s_n}\}$, the PPO agent PPO_θ

Output: The result of embedding $\mathcal{E}$.

Initialize: The substrate node feature $node\_feature$ $\mathcal{N}_s \leftarrow \{\}$, $\mathcal{E} \leftarrow []$, buffer $\mathcal{B} \leftarrow []$

for $node$ **in** $\mathcal{N}_s$ **do**

$\mathcal{S}[node][link\_feature] \leftarrow \sum_{v \in \mathcal{N}(node) \cap \{node\}} \mathcal{S}[v][link\_feature]$

// Encode node features by aggregating the neighbor link information such as link delay.

for r **in** $\mathcal{R}$ **do**

$\mathcal{S} \leftarrow Update(\mathcal{S}, r, None)$

$node\_result, reward_n \leftarrow PPO_\theta(\mathcal{S}, r)$ // Virtual node embedding stage: Perform node embedding operation using PPO model and get the node embedding result and reward.

$edge\_result, reward_e \leftarrow EdgeEmbedding(\mathcal{S}, virtual\_edge)$

// Routing stage: Perform edge embedding operation by applying Dijkstra's Algorithm. The result will be $None$ if no possible substrate link is found.

$result \leftarrow (node\_result, edge\_result)$

$\mathcal{E}.append(result)$

$\mathcal{S}' \leftarrow Update(\mathcal{S}, r, result)$ // Update substrate network. During the process the availability of nodes corresponding links will be updated.

$\mathcal{S} \leftarrow \mathcal{S}'$

$\mathcal{B}.append((\mathcal{S}, node\_result, edge\_result, reward_n, reward_e))$ // Save the data for batch training in PPO algorithm.

if $IsFull(\mathcal{B})$:

$PPO_{\theta'} \leftarrow UpdateAgent(PPO_\theta, \mathcal{B}, \mathcal{N}_s)$ // Update the parameters of PPO agent, where $\theta' = \theta + \Delta\theta$, and $\Delta\theta$ is the parameter update derived from the optimization process.

$\mathcal{B} \leftarrow []$

end for

- Available substrate network resource

The available resource amount of type r of the substrate node n^{S} at snapshot t is defined as the remaining amount of resource, i.e.,

$$a_{t,n,r}^A = a_{n,r}^S - \sum_{n^V \in F_t(n^S)} a_{n,r}^V \tag{2}$$

where $F_t(n^{\mathrm{S}}) \in N^{\mathrm{S}}$ denotes the set of virtual nodes embedded to substrate node n^{S} at snapshot t.

Moreover, the available bandwidth of the substrate link $(n^{\mathrm{S}}, m^{\mathrm{S}})$ at snapshot t is defined as the remaining bandwidth, i.e.,

$$w_{t,n,m}^A = w_{n,m}^S - \sum_{(n^V, m^V) \in F_t(n^S, m^S)} w_{n,m}^V \tag{3}$$

where $F_t(n^S, m^S) \in E^{\mathrm{V}}$ denotes the set of virtual links implemented onto the substrate link (n^S, m^S) at snapshot t.

- Transmission delay

In the context of SFC embedding in integrated service networks, the delay caused by electromagnetic wave propagation is significantly lower than the delay incurred during packet transmission. Therefore, in this scenario, the total transmission delay is primarily attributed to the packet transmission process. For virtual network functions embedding, the transmission delay $\tau_{n,m}^{\mathrm{V}}$ for implementing substrate link $(n^{\mathrm{V}}, m^{\mathrm{V}})$ onto substrate link $(n^{\mathrm{S}}, m^{\mathrm{S}})$ is defined as follows

$$b_{n,m}^V = \int_{t_0}^{t_0 + \tau_{n,m}^V} c_{t,n,m}^S \, dt \tag{4}$$

Where $b_{n,m}^V$ is the total number of bits transmitted, and t_0 is the starting time for transmission.

- Multi-objective optimization

The virtual node n^{V} is allowed to embedded to a substrate node n^{S} only if

$$a_{n,r}^V \le a_{t,n,r}^A \tag{5}$$

Similarly, the virtual link $(n^{\mathrm{V}}, m^{\mathrm{V}})$ is allowed to embedded to a substrate node $(n^{\mathrm{S}}, m^{\mathrm{S}})$ only if

$$w_{n,m}^V \le w_{t,n,m}^A \tag{6}$$

The optimization problem is to find a SFCs embedding policy under the above constrains that maximizes the success ratio of the SFCs embedding and minimizes the average transmission delay. Therefore, such problem can be transformed into a multi-objective optimization problem. Heuristically, we use an equivalent transmission delay $\tilde{\tau}_k$ for the SFCs rejected. Then the optimization can be formulated as minimizing the equivalent average transmission delay

$$min \sum_{h_k \in H} \left(\tilde{\tau}_k + \sum_{(n^{V,k}, m^{V,k}) \in E^{V,k}} \tau_{n,m}^{V,k} \right) \tag{7}$$

where k denotes the SFC index.

III. METHODOLOGY

A. Markov decision process

The SFCs embedding problem can be modeled as a Markov Decision Process (MDP) that aims to learn a policy function $\pi(a \mid s)$, which takes action a (which substrate node the virtual node is embedded to) based on the state s to maximize long-term rewards, i.e.,

$$\pi = argmax_\pi E_{\pi,p}\left[\sum_t \gamma^{\tilde{t}-1} r_{\tilde{t}}\right] \tag{8}$$

where r_t is the reward function and $p(s', r|s, a)$ is Markov transition dynamics. The design of $r_{\tilde{t}}$ is essential for the performance of SFCs embedding, as one can let the network obtain higher rewards by taking actions as wanted.

B. Proximal policy optimization

The Proximal policy optimization (PPO) algorithm which is a modified Actor-Critic algorithm uses two main networks: an Actor network for policy optimization and a Critic network for Q-value estimation. The Actor network adjusts parameters θ through policy gradient updates to maximize expected return by enhancing expected rewards for action selection via gradient ascent. Simultaneously, the Critic network adjusts

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on March 01,2025 at 12:42:52 UTC from IEEE Xplore. Restrictions apply.

parameters φ by minimizing mean squared error (MSE) to accurately estimate Q-values for state-action pairs, aligning them with actual cumulative rewards. To ensure training stability, PPO introduces target policy and target Q networks. Target network parameters θ and φ gradually converge with the online network parameters using soft update strategies controlled by τ. This approach facilitates smooth parameter transitions, enhancing algorithm stability and convergence speed. Additionally, an experience replay pool N stores historical data for random sampling during parameter optimization, thereby improving training effectiveness. The discount factor γ influences the Critic network's estimation of future rewards, shaping the algorithm's learning behavior to emphasize long-term rewards. The PPO has an actor network and a critic network, as show in figure 3. The theory and steps of PPO algorithm is given in [10]. For simplicity, the PPO network characterized by θ is defined as $\mathrm{PPO}(s|\theta)$, where s is the STIN state. The Output of PPO is the substrate node, where the virtual node is embedded.

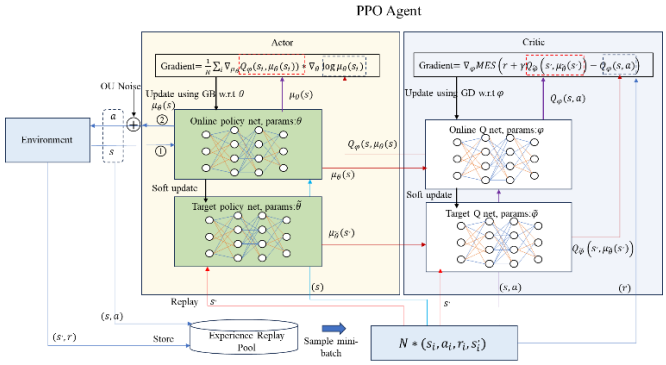

Fig.3 The PPO algorithm

When embedding the SFC h_k, in order to learn the STIN pattern fully, the STIN state s at snapshot $\tilde{t}$ is consisted of $\{a^{\mathrm{A}}_{\tilde{t},n,r}\}$, $\{w^{\mathrm{S,A}}_{\tilde{t},n,m}\}$, $\{a^{\mathrm{V}}_{n,r}\}$, $\{w^{\mathrm{A}}_{n,m}\}$, $\{b^{\mathrm{V}}_{n,m}\}$, a one-hot vector indicating the last virtual node is embedded to which substrate node, a one-hot vector indicating the next virtual node to be embedded and a weighted adjacency matrix given by

$$[\Lambda]_{n,m} = \int_{t_k^{\mathrm{ar}}}^{t_k^{\mathrm{de}}} w^{V}_{n,m}\, log_2\Big(1 + \frac{\eta p_{n,m} (d^{S}_{\tilde{t},n,m})^{-2}}{\sigma^2}\Big)\, dt \tag{9}$$

where $[\Lambda]_{n,m}$ is proportional to integral of channel capacity.

C. PPO-based SFCs embedding

The DRL-based SFCs embedding can be decomposed into four stages in Algorithm 1.

1) Feature aggregation encodes the features of all nodes (primarily link features) into feature vectors. Assuming that a node can access the information of its adjacent one-hop neighbors, the feature vectors of all nodes and their neighbor nodes are aggregated to form context information, which serves as a part of the input state for PPO.

2) Virtual node embedding determines the current virtual node is embedded to which substrate node that satisfies constraint (6) by PPO. If no substrate node satisfies the constraint, the SFC is rejected.

3) Routing: Once a virtual node is embedded, a weighted Dijkstra algorithm on the weighted adjacency matrix Λ is used to find the substrate link with the maximum cumulative channel capacity, on which the virtual link is embedded. If no substrate link satisfies the constraint (6), the SFC is rejected. If not, do select the next virtual and repeat the first stage.

4) PPO: Once the routing stage is finished, the state is updated. Then, the transition (s, a, s', r) is stored to the buffer for PPO. Once the two and three stages are repeated a certain times, optimize the parameter θ of the network $\mathrm{PPO}(s|\theta)$ by utilizing the transitions sampled from the buffer.

IV. Simulation Results

In this section, we analyze the performance of the proposed DRL-based SFCs embedding algorithm. We consider a STIN consisting of a satellite constellation with 24 satellites and 10 terrestrial stations. The link parameters of the STIN are generated by Satellite Tool Kit (STK).

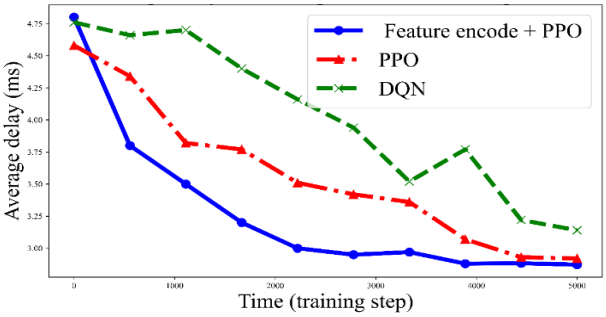

Fig. 4 The average delay of different algorithms when the number of SFC requests is 20.

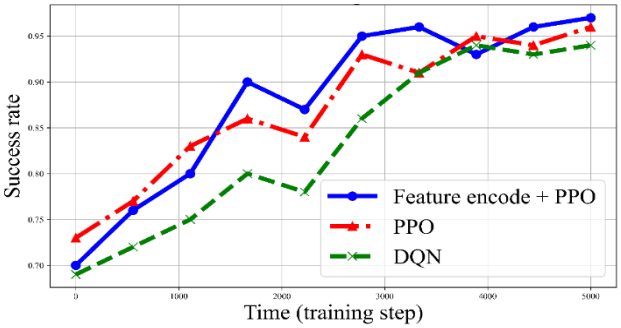

Fig.5 The success rate of different algorithms when the number of SFC requests is 20.

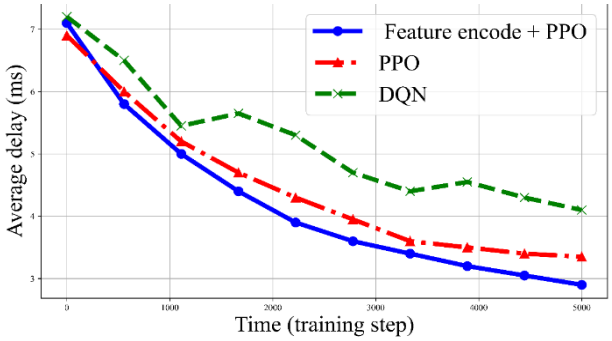

Fig.6 The average delay of different algorithms when the number of SFC requests is 100.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on March 01,2025 at 12:42:52 UTC from IEEE Xplore. Restrictions apply.

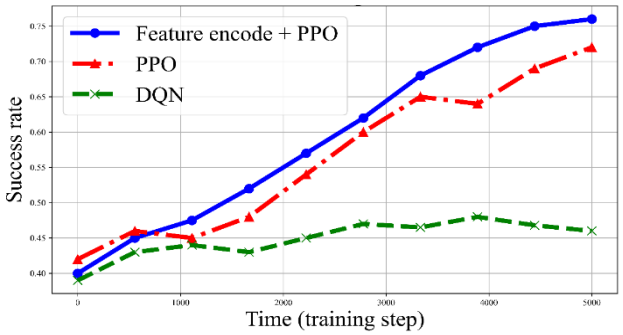

Fig.7 The success rate of three algorithms when the number of SFC requests is 100.

For training, the Adam optimizer is used for gradient backpropagation, the learning rate lr is set to be 0.02. In PPO, the discount factor γ=0.99, and the clipping parameter ε=0.2, The training results are presented in Figures 4-7. Fig. 4 and Fig. 5 respectively show the performance of Feature encode + PPO, PPO, and DQN in terms of average delay and success rate when the number of SFC requests is 20. It can be observed that, by applying PPO to the virtual network embedding (VNE) task, lower delay, higher success rate, and faster convergence speed can be achieved when compared to the traditional DQN algorithm. Our proposed four-phase VNE method, Feature encode + PPO, outperforms methods only using PPO or DQN, which demonstrates that the feature encode phase effectively extracts neighborhood information of nodes, thus enhancing the effectiveness of training phase.

To further illustrate the superiority of our proposed algorithm, we increased the number of SFC requests within the same time interval to 100 and placed the corresponding results in Fig. 6 and Fig. 7. It can be observed that, the increase of the number of requests raises the network's load, resulting in performance degradation. When only DQN method is used, there is a noticeable fluctuation in training performance, and the success rate improves slightly. Although the PPO method alone performs better, it still does not converge as quickly as our proposed method.

CONCLUSIONS

This paper presents an innovative embedding SFCs within STINs by integrating Network Function Virtualization (NFV) and Deep Reinforcement Learning (DRL). The dynamic nature of STINs, largely due to satellite movements, poses a challenge for effective SFC embedding. To solve this problem, the proposed DRL-based algorithm initiates with a feature fusion phase that integrates link and node attributes, which then forms the contextual basis for the embedding decisions. The proposed architecture leverages NFV and AI to dynamically manage network resources, aiming to enhance service quality and network resilience. The simulation results demonstrate the effectiveness of the proposed algorithm in both reducing latency and improving the rate of successful SFC embeddings. Specifically, the simulation results show that the DRL-based algorithm outperforms traditional methods with lower latency and higher success rates in embedding SFCs, which is critical for maintaining efficient and reliable network operations in STINs.

REFERENCES

[1] H. Yao, et. al, "The space-terrestrial integrated network: An overview," IEEE Commun. Mag., vol.56, pp. 178-185, 2018.

[2] S. Zhang, D. Zhu, Y. Wang, "A survey on space-aerial-terrestrial integrated 5G networks for future mobile communications," Computer Netw., no.174, pp.1-18, 2020.

[3] Bi Y, Han G, Xu S, et al., "Software defined space-terrestrial integrated networks: Architecture, challenges, and solutions" IEEE Netw., pp.22-28, 2019.

[4] K. Guo, Y, Ren, Y. Zhang, Y. Liu, W. Li, "Challenges and Opportunities for Terrestrial Network Operators in the Air-Space-Ground Integrated Network," Int. Conf. Wireless Commun. Signal Process. (WCSP), 2020, pp. 1-6.

[5] N. Toumi, M. Bagaa, S. Ksentini, "On using Deep Reinforcement Learning for Multi-Domain SFC placement," IEEE Global Commun. Conf. (GLOBECOM), 2021, pp. 1-6.

[6] Y. Bi, G. Han, S. Xu, et al. "Software defined space-terrestrial integrated networks: Architecture, challenges, and solutions," IEEE Network, vol.33, no.1, pp. 22-28, 2019.

[7] S. Richard, B. Andrew, Reinforcement Learning: An Introduction, MIT Press, pp.291-312, 1998.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, et al. "Playing Atari with Deep Reinforcement Learning," Computer Science, pp. 1-9, 2013.

[9] P. Sollich, A. Krogh, "Advances in neural information processing systems," Biochem. Biophys. Res. Commun., 2003.

[10] J. Schulman,, et al. "Proximal Policy Optimization Algorithms," Computer Science, pp. 1-12, 2017.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on March 01,2025 at 12:42:52 UTC from IEEE Xplore. Restrictions apply.