Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Local search resource allocation algorithm for space-based backbone network in Deep Reinforcement Learning method[☆]

Peiying Zhang [a,b], Zixuan Cui [a], Neeraj Kumar [c], Jian Wang [d], Wei Zhang [b,e], Lizhuang Tan [b,e,*]

[a] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[b] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China
[c] Department of Computer Science and Engineering, Thapar University, Patiala 147004, India
[d] College of Science, China University of Petroleum (East China), Qingdao 266580, China
[e] Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan 250014, China

ARTICLE INFO

Keywords:
Virtual network embedding
Resource allocation
Space-based backbone network
Deep reinforcement learning
Local search

ABSTRACT

With the evolution of Space-based backbone networks, the demand for enhanced efficiency and stability in network resource allocation has become increasingly critical, presenting a substantial challenge to conventional allocation methods. In response, we introduce an innovative resource allocation algorithm for space-based backbone networks. This algorithm represents a synergistic fusion of Deep Reinforcement Learning (DRL) and Local Search (LS) methodologies. It is specifically designed to reduce the extensive training duration associated with traditional policy networks, a crucial aspect in assuring optimal service quality. Our algorithm is structured within a two-stage framework that seamlessly integrates DRL and LS. A distinctive feature of our approach is the incorporation of link reliability into the algorithmic design. This element is meticulously tailored to address the dynamic and heterogeneous nature of space-based networks, ensuring effective resource management. The effectiveness of our approach is substantiated through extensive simulation results. These results demonstrate that the integration of DRL with LS not only enhances training efficiency but also exhibits significant improvements in resource allocation outcomes. Our work represents a noteworthy contribution to the development of practical optimization strategies in space-based networks, merging DRL with traditional methodologies for improved performance.

1. Introduction

The burgeoning advances in space technology have made the construction of space-based backbone networks, leveraging satellites and space vehicles, an inevitable trend [1]. These networks, characterized by their unique attributes, are poised to augment traditional terrestrial networks, offering expansive global coverage and reduced latency in access. In the context of these networks, efficient resource management emerges as a critical component to ensure Quality of Service (QoS), especially given the dynamic nature and specific constraints of such systems.

Fig. 1 illustrates the intricate structure of the space-based backbone network, which is composed of a diverse array of satellites located along different orbital paths. This multifaceted network integrates various types of constellations, including GEO (Geosynchronous Equatorial Orbit), MEO (Medium Earth Orbit), and LEO (Low Earth Orbit) systems. These satellites are interconnected through Inter-Satellite Links (ISL) and Inter-Orbital Links (IOL), forming a comprehensive network. It is worth noting that in the space network structure we are considering, a layered design idea is adopted, similar to the Model-View-Controller (MVC) design pattern in software engineering. This design not only

☆ This work is partially supported by the Natural Science Foundation of Shandong Province under Grants ZR2023LZH017, ZR2022LZH015 and 2023QF025, partially supported by the National Natural Science Foundation of China under Grant 62173345, partially supported by the Open Project of Key Laboratory of Comput- ing Power Network and Information Security, Ministry of Education under Grant 2023ZD010, partially supported by the Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023PX057, and partially supported by the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023RCKY141.

* Corresponding author at: Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China.

E-mail addresses: zhangpeiying@upc.edu.cn (P. Zhang), Z22070047@s.upc.edu.cn (Z. Cui), neeraj.kumar@thapar.edu (N. Kumar), wangjiannl@upc.edu.cn (J. Wang), wzhang@sdas.org (W. Zhang), tanlzh@sdas.org (L. Tan).

https://doi.org/10.1016/j.adhoc.2024.103575
Received 8 January 2024; Received in revised form 10 April 2024; Accepted 6 June 2024

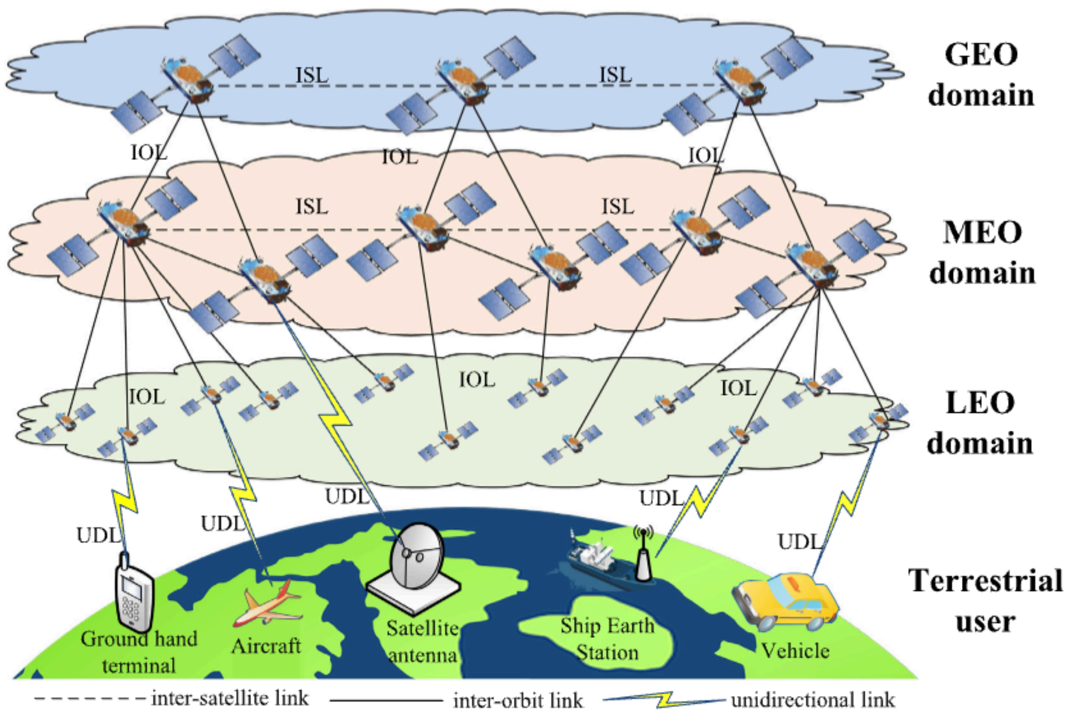

Fig. 1. Space-based backbone network architecture.

reduces the need for direct long-distance links while utilizing the MEO satellites as relays to improve the flexibility and reliability of the network, but also helps to clarify the layering at the space and business levels. The GEO satellites are indirectly connected to the LEO satellites through the MEO satellites, rather than being directly cross-connected through the IOL. Specifically, GEO satellites, MEO satellites and LEO satellites take on different roles and functions respectively; GEO satellites are located in the highest layer and are mainly responsible for providing stable global coverage and long-term services; MEO satellites are located in the middle layer and act as a bridge, responsible for connecting the GEO and LEO layers and providing relay services; LEO satellites are located in the lowest layer and are responsible for providing fast communication services near the ground with their low-latency and high-throughput characteristics fast communication services near the ground. The strategic implementation of a LEO satellite constellation as the core network infrastructure, supported by a cohesive control architecture for satellites in medium and high orbits, brings forth significant advantages. Nevertheless, the dynamic nature and heterogeneity intrinsic to space-based backbone networks pose formidable challenges in the realm of resource management. These challenges are pivotal in ensuring the QoS, a critical aspect in the operation of these networks. Addressing these challenges requires innovative approaches to effectively manage the resources while adapting to the dynamic conditions and varied requirements inherent in space-based network systems.

The development of China's space-based information network [1,2], encompasses a diverse fleet of aircraft in various orbits, each characterized by distinct types and performance capabilities, in addition to related ground facilities and application systems. This network is distinguished by its ability for intelligent acquisition, storage, transmission, processing, integration, and dissemination of information. It is further enhanced by a high level of autonomous operational and management competencies. In this context, the architecture of China's space-based information network involves both Earth Observation (EO) services, for monitoring and analyzing the Earth's surface and atmosphere, and Satellite Communication (SatCom) services, for remote communication and data transmission. Therefore, our solution aims to optimize the allocation of network resources to support the efficient operation of these services. Specifically, the optimization objectives include increasing network throughput, and enhancing service reliability, while taking into account constraints such as satellite transmission capacity and link bandwidth limitations.

In such a context, the task of resource allocation in the space-based information network is a multifaceted challenge, involving multi-source, multi-type dynamic scheduling. The essence of modeling and algorithm development in this domain is underpinned by the concept of resource virtualization. This approach aims to standardize the categorization of resources, facilitating more efficient modeling processes. Consequently, Network Virtualization (NV) presents itself as a viable solution to these challenges. NV technology enables the abstraction, segmentation, and distribution of the foundational physical network infrastructure. Virtual Network Embedding (VNE) plays a pivotal role. VNE is the process of mapping virtual networks (VNs) onto a shared substrate network (SN), also known as a physical network (PN). The goal of VNE is to efficiently allocate the resources of the substrate network to support the demands of multiple virtual networks, thereby maximizing the utilization of the substrate network and ensuring the coexistence of multiple VNs. Therefore, by delving into the realm of multi-domain VNE algorithms [3], it becomes possible to effectively tackle the inherent resource allocation challenges in space-based backbone networks.

In recent years, deep reinforcement learning (DRL) has shown great potentials in dealing with dynamic resource allocation problems [4, 5], and attractive results have been achieved on several challenging problems such as Go [6], application recommendation [7] and combinatorial optimization [8–10] are representative problems where DRL has made progress. In a more detailed context, models based on DRL have been introduced as solutions to a variety of NP-hard optimization challenges. These include, but are not limited to, the Maximum Cut Problem(MCP) [11], and the Travelling Salesman Problem (TSP) [12], the vehicle routing problem (VRP) [9,13], the Knapsack Problem, the Bin-Packing Problem, and the Capacitated Facility Location Problem [14,15].

However, there are several challenges in solving the VNE problem with DRL:

- In the realm of space-based communications, signals traverse extensive propagation paths and encounter intricate multi-hop queues. Given the substantial distances between satellites and ground stations, the imperative for reliability is paramount. However, existing algorithms demonstrate a notable deficiency in the quantification of this reliability, a critical aspect in such long-range communication systems.
- The training phase of DRL models is notably time-intensive. Despite the rapidity of DRL inference, the extensive duration required for the training process remains a significant consideration.
- Current DRL models often face challenges in achieving an optimal balance between exploration and exploitation mechanisms. This imbalance can lead to a compromise in the quality of the solutions generated.

To address the aforementioned challenge and secure an optimal solution within a feasible computational timeframe, we propose a two-stage framework for solving node embedding. The node embedding phase is divided into two stages. The first phase is implemented by a policy network for DRL, while the second phase considers local search heuristics.

The specific contributions of this paper are as follows:

- We propose a VNE algorithm that synergistically combines DRL with LS techniques. This integrated framework proficiently addresses the VNE problem, delivering satisfactory solutions while maintaining reasonable computational efficiency.
- In addition to traditional metrics like CPU capacity and link bandwidth, our approach uniquely incorporates considerations of node packet loss and link reliability. This expanded metric set is crucial to fulfill the requirements for reliable link connections within space-based backbone networks.
- A comprehensive series of simulation experiments were undertaken, wherein the key performance metrics of VNE were juxtaposed with those derived from alternative algorithms. The effectiveness of our proposed algorithm is confirmed.

The rest of the paper is organized as follows. We begin by analyzing related work in Section 2. Section 3 introduces the VNE related issues and models the system. In Section 4, the detailed architecture and implementation specifics of the algorithm are delineated. Subsequently, Section 5 provides a comprehensive presentation of experimental outcomes and evaluative findings. Finally, a summary of the work and an outlook for future work is given in Section 6.

2. Related work

In the previous decade, exact and heuristic algorithms have been the mainstream for solving VNE problems, such as Tabu Search algorithm and Genetic algorithm often used as baseline algorithm in other papers. In recent years, the advent of Reinforcement Learning (RL) has marked its ascendancy as a predominant method for addressing NP-hard challenges, such as VNE. In this section, we introduce Exact and Heuristic algorithms and Reinforcement Learning algorithms, In order to show more clearly the differences in technical approaches, optimization objectives and constraints among different research works, we summarize some representative related works in Table 1.

2.1. Exact and heuristic algorithms

VNE has been studied for decades, and researchers have developed a large number of solution methods, including the Exact, the Heuristic and the Meta-Heuristic [27]. Exact algorithms, including linear programming (LP) methods, integer linear programming (ILP) [18] and mixed integer linear programming (MILP) [19], are able to give optimal solutions under exhaustive conditions. However, as the problem size and complexity increase, the computational complexity of the exact algorithms for solving the active optimal solution increases exponentially, requiring a large amount of computational resources to obtain the optimal solution. To reduce the computational effort, it is usually necessary to design approximate or heuristic algorithms that give times these algorithms according to the characteristics of the problem using heuristic techniques or heuristic rules to effectively prune the combinatorial mathematical explosion problem, greatly reducing the search space. Approximate and heuristic algorithms provide feasible and effective methods for solving large-scale complex problems.

In contrast to exact algorithms, heuristic algorithms do not find the absolute optimal solution for each VNE when simulating VNE problems. The heuristic algorithm can give a sub-optimal but good enough VNE solution. This is done by sacrificing absolute optimality [28]. The aim is to improve computational speed. This allows the heuristic algorithm to provide solutions at urgent time intervals. This makes heuristic algorithms well suited for large-scale VNE problems with real-time requirements for evaluation. However, heuristic algorithms have the disadvantage of falling into a local optimal solution while having difficulty in reaching a global optimal solution when solving optimization problems. Local Search (LS) method also known as neighborhood search, are an important class of heuristic algorithms. Local search starts from an initial solution and searches in the space around the initial solution to obtain a better solution. Many well-known heuristic algorithms are derived based on the idea of local search, including Simulated Annealing [20,21], Tabu Search [22] and Variable Neighborhood Search (VNS) [23]. Local search heuristics are widely used in Combinatorial Optimization Problem, Vehicle Routing Problem (VRP) and other fields for solving practical problems due to their flexibility and efficiency. However, when the solution in the neighborhood space does not improve, the search stops and thus falls into a local optimum. Therefore the quality of the initial solution becomes particularly important.

2.2. Reinforcement learning algorithms

RL is another emerging class of methods for solving combinatorial optimization problems. Unlike traditional heuristic algorithms that rely on manual design, RL automatically learns problem-solving strategies through interaction with the environment. Aiming at the characteristics of combinatorial optimization problems, researchers have proposed various algorithmic frameworks for integrating RL. For instance, the Deep Q-Networks (DQN) based RL framework employs deep neural networks for approximating the value function within optimization problems. This approach incrementally converges towards an optimal strategy by means of empirical adjustments [29]. Zhang et al. [24] investigated a node probability-based RL framework for VNE. The paper proposes a VNE algorithm based on node probability using RL. Afifi et al. [25] introduced a RL framework grounded in Q-Learning, which adopts either a greedy Epsilon or an Epsilon decay strategy for exploration purposes. The outcomes derived from these two exploration methodologies were juxtaposed with those obtained from optimization techniques. Empirical evidence demonstrates that the RL framework yields favorable results, particularly in reducing network latency, within a limited number of iterations. Zhang et al. [26] developed an algorithm for VNE that integrates computational and storage resource considerations alongside security constraints. This approach is designed to guarantee both the rationality and security of resource allocation within Industrial Cyber-Physical Systems (ICPS). The proposed algorithm encompasses a robust two-stage RL-VNE framework. It comprehensively incorporates a tri-dimensional resource constraint model, addressing computation, storage, and security aspects simultaneously.

3. Problem statement

In this section, we present the resource allocation problem in space-based backbone networks, including problem description, modeling and formulation.

Table 1
Summary of related work on virtual network embeddinzg.

Reference	Node-Link constraints	Optimization approach	Main contribution
Chowdhury [16](2012)	CPU, link bandwidth	Relaxed LP model, rounding techniques	Model VNE problem by using MILP model for the first time in the literature.
Melo [17](2013)	Node location, CPU, link bandwidth	Pure ILP model	Use pure ILP model to embed proposed VNs.
Yang [18](2016)	CPU, link bandwidth, node location	Pure ILP model	Proposed an exact VNE algorithm (ILP-LC) that considers node location constraints and aims to minimize the cost of substrate network while maximizing VNR acceptance ratio.
Hu [19](2014)	Processing capacity, link bandwidth	Enhanced path-based milp model	Developed a novel VNE framework that decomposes the embedding process into subproblems, leading to a more efficient and scalable solution.
Masti [20](2012)	CPU, link bandwidth	Simulated annealing	Proposed a simulated annealing algorithm for VN reconfiguration to balance the load across the substrate network, thereby reducing peak node and link load.
Zhang [21](2011)	CPU, link bandwidth, node location	Simulated annealing	Proposed FELL, a flexible VNE algorithm with guaranteed load balancing, using simulated annealing to control the trade-off between accuracy and running time.
Diallo [22](2014)	CPU, link bandwidth	Ant Colony Optimization (ACO) and Tabu search	Proposed a hybrid meta-heuristic approach combining ACO and TS for VNE in multi-cloud environments, aiming to minimize the resource supply costs of SPs while improving the performance and QoS of embedded VNR fragments.
Luizelli [23](2017)	CPU, link bandwidth	ILP model and VNS	Proposed a fix-and-optimize heuristic algorithm for large scale VNF placement and chaining, combining ILP and VNS to efficiently generate high-quality solutions.
Zhang [24](2020)	CPU, link bandwidth	Policy network	Proposed a VNE algorithm based on node probability using reinforcement learning, which trains an agent to deduce the mapping probability of each node and ranks the substrate nodes according to this probability for embedding.
Afifi [25](2020)	CPU, link bandwidth	Q-learning	Proposed a reinforcement learning framework for VNE in wireless sensor networks, using Q-learning with Greedy Epsilon or Epsilon Decay for exploration. The framework aims to achieve good results in terms of network delay within a few number of steps.
Zhang [26](2022)	CPU, link bandwidth, node security level	Policy network	Proposed a VNE algorithm with computing, storage, and node security constraints for resource management and security in ICPSs and IoT, using RL to improve performance.

3.1. Problem description

The space-based backbone network consists of multiple layers of satellites and ground stations providing connectivity services. It can be viewed as a hierarchical network structure consisting of geosynchronous orbit, medium orbit and low orbit, with inter-satellite link (ISL) connecting satellites in the same layer and inter-orbit link (IOL) connecting different orbits. The satellite network has a dynamic topology characterized by changes over time. Distinct segments of the network exhibit varying link delay characteristics at different temporal intervals. Consequently, VNE algorithms, originally tailored for terrestrial networks, are not directly transferable to space-based backbone network. To tackle this challenge, we embrace the concept of Virtual Topology. Virtual Topology refers to an abstract representation of the physical network, where physical resources are mapped onto virtual entities. This abstraction allows for more flexible and efficient resource allocation, as it decouples the physical constraints from the logical network design. For dynamic satellite network architectures, segmenting the perpetually evolving discrete dynamic network topology into discrete time slots can solve the problem of modeling difficulties. We approximate the topology in each time slot t as a static topology. Through this approach, the resource allocation issue within space-based backbone network is effectively reformulated into a Multi-domain VNE problem.

3.2. Network models

In our model, the physical and virtual networks are represented as undirected weighted graphs. This simplifies the abstraction of network topology and resource attributes, allowing for efficient computation and analysis of the VNE problem. The undirected nature suits symmetric communication channels, while weights represent key resource attributes like CPU capacity, link bandwidth, and reliability. This approach facilitates the modeling of resource constraints and requirements for embedding virtual networks onto the substrate network. In the context of the Multi-domain VNE challenge, the physical network is conceptualized as a weighted undirected graph and represented as $G^P = \{N^P, L^P, A_N^P, A_L^P\}$. N^P represents the set of all substrate nodes of the physical network, these substrate nodes include various types of satellites, such as GEO, MEO, and LEO satellites, which form the infrastructure of the network. L^P represents the set of all substrate links of the physical network. The substrate node set N^P contains three types of substrate nodes, namely GEO, MEO and LEO nodes. The substrate link set L^P contains various links including ISLs between three different satellites and IOLs between different orbits. As for A_N^P, we consider it from CPU capacity of substrate nodes CPU_{n^p} and the pocket loss rate PL_{n^p}. For the substrate link property A_L^P, we consider the post-FEC payload data rate between two adjacent nodes as the link bandwidth BW_{l^p}, and the probability of successful data transmission as the link reliability R_{l^p}. Correspondingly, another weighted undirected graph $G^V = \{N^V, L^V, A_N^V, A_L^V\}$, is defined for VN, The symbols N^V and L^V denote the sets of virtual nodes and links, correspondingly. Furthermore, A_N^V and A_L^V are used to represent the sets of attributes requisite for the virtual nodes and links, respectively. The set of attributes required for a virtual node A_N^V contains the CPU requirements CPU_{n^v} in the VNRs. The set of attributes required for a virtual link A_L^V includes the available bandwidth requirements BW_{l^v} and the Reliability requirements R_{l^v}. Specific descriptions of node and link categories are given in Table 2.

With the above conditions, The embedding process of the VNR can be modeled as: $G^V \rightarrow G^P$. Fig. 2 illustrates the procedure of

Table 2
Nomenclature.

Network	Notation	Description	Units
Physical network	G^P	Space-based backbone network	–
	N^P	Substrate nodes (GEO, MEO, LEO satellites)	–
	L^P	Links between physical nodes	–
	CPU_{n^p}	CPU capacity of physical node	GHz
	BW_{l^p}	Bandwidth of physical link	Mbps
	R_{l^p}	Reliability of physical link	Percentage (%)
Virtual network	G^V	Virtual network	–
	N^V	Virtual nodes	–
	L^V	Virtual links	–
	CPU_{n^v}	CPU requirement of virtual node	GHz
	BW_{l^v}	Bandwidth requirement of virtual link	Mbps
	R_{l^v}	Reliability requirement of virtual link	Percentage (%)

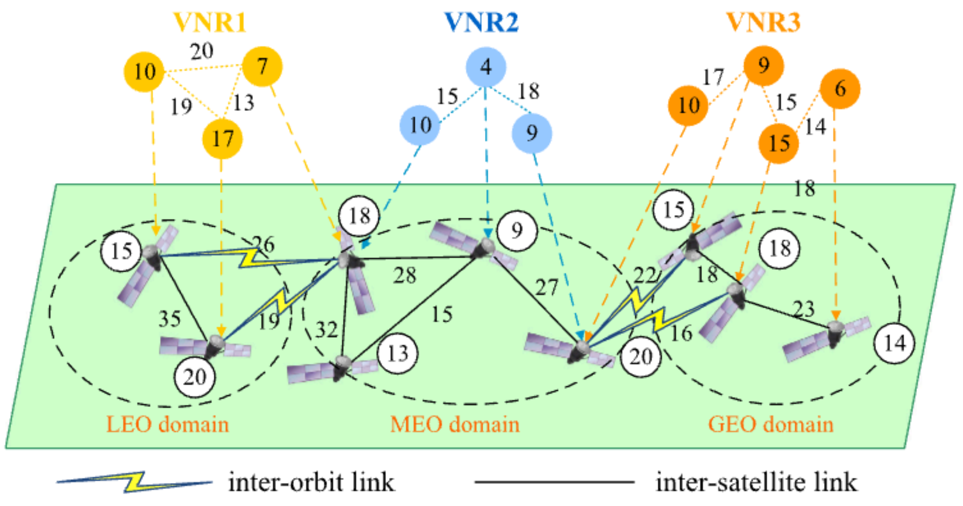

Fig. 2. An example of VNE.

multi-domain VNE in a space-based backbone network. The numerical values depicted on the links represent the requisite bandwidths, while those within the substrate nodes indicate the capacity requirements of each substrate node. For simplicity, the required reliability requirement attribute of the link are also omitted. The embedding procedure is bifurcated into two distinct phases: firstly, the node embedding phase, wherein virtual nodes are mapped onto the substrate nodes; secondly, the link embedding phase, during which virtual links are embedded into the paths traversing the physical network devices. It is noteworthy that two virtual nodes originating from disparate virtual networks may coalesce on a single substrate node. Similarly, two virtual links from different virtual networks can concurrently utilize a substrate link, provided specific conditional constraints are met. The conditional constraints are described in turn below.

We define the node embedding function as $F1 = \{\theta_{ij} | \, n_i^v \in N^V, n_j^p \in N^P\}$, and θ_{ij} can be formulated as:

$$\theta_{ij} = \begin{cases} 1, \; if \; n_i^v \uparrow n_j^p. \\ 0, \; else. \end{cases} \tag{1}$$

Formula (1) stipulates that n the context of the same Virtual Network Request (VNR), each virtual node must be uniquely mapped to a substrate node, but virtual nodes from different VNRs can share the same underlying node. Throughout the node embedding phase, it is imperative that each successfully mapped virtual node possesses a computational resource demand that is either equivalent to or less than the available computational resources of the corresponding substrate node, as shown in formula (2):

$$if \; \theta_{ij} = 1, \; CPU_{n_i^v} \le CPU_{n_j^p}, \tag{2}$$

as for link embedding, we define a embedding $F2 : l_i^v \uparrow L^P$, where l_i^v refers to the ith link in VNR. We represent the embedding process as function $F2 = \{\phi_{ij} | \, l_i^v \in L^V, l_j^s \in L^P\}$, where ϕ_{ij} can be formulated as:

$$\phi_{ij} = \begin{cases} 1, \; if \; l_i^v \uparrow l_j^p. \\ 0, \; else. \end{cases} \tag{3}$$

analogous to the node embedding procedure, each virtual link that is eligible for mapping onto a substrate link should adhere to corresponding criteria, and there are similar constraints in terms of link bandwidth and reliability:

$$if \; \phi_{ij} = 1, \; BW_{l_i^v} \le BW_{l_j^p}, \tag{4}$$

$$if \; \phi_{ij} = 1, \; R_{l_i^v} \le R_{l_j^p}, \tag{5}$$

for link reliability, we consider the packet loss rate at the node level, where each node represents a satellite. This decision is based on empirical observations from a confidential project under a research institute of the China Electronics Technology Group. Specifically, in this project, the satellite exhibited a packet loss rate, despite having a mechanism to repair a 2% packet loss rate. This real-world phenomenon led us to include the node packet loss rate as a factor in our modeling. We suppose a network link consists of the n_i^p to the n_j^p substrate node and the packet loss rate of each substrate node n_i^p is $PL_{n_i^p}$. Then the reliability of the whole link can be expressed as:

$$R_{(n_i^p, n_j^p)} = \prod_{k=i}^{j} (1 - PL_{n_k^p}), \tag{6}$$

any path $p^P \in P^P$ consists of one or more substrate links, each of which is equivalent to constructing a physical path in a serial mode. Then the path reliability is expressed in formula (7).

$$R_{p^P} = \prod_{l^P \in p^P}^{p^P} R_{l^P}. \tag{7}$$

when it is a single-hop link, which means that only one link is in the path, then the path reliability at this point is equivalent to the reliability of that link R_{l^P} which can be expressed by the formula (6).

3.3. Evaluation indicators

3.3.1. Average node utilization

For evaluating the average utilization of nodes, in order to make the computational model more representative to reflect the overall workload of nodes, this study only considers the core index of CPU computing resource utilization of nodes, expressed as:

$$ANU = \frac{\sum_{n_i^v}^{N^V} CPU_{n_i^v} F1(n_i^v)}{\sum_{n_k^p}^{N^P} CPU_{n_k^p}}. \tag{8}$$

3.3.2. Average link utilization

The average link utilization differs from the node utilization in that it considers both the bandwidth of the link and the packet loss rate. Packet loss rate can be categorized into two types, source packet loss rate and transmission packet loss rate. The packet loss in transmission occupies part of the link bandwidth. Therefore, the average link utilization can be expressed as:

$$ALU = \frac{\mu \sum_{l_i^v}^{L^V} BW_{l_i^v} F2(l_i^v) R_{(l_i^v, l^p)}}{\sum_{l_k^p}^{L^P} BW_{l_k^p}}. \tag{9}$$

in formula (9), to facilitate the calculation, we set the percentage of source packet loss and transmission packet loss for each link to a fixed value. μ denotes the percentage of successfully transmitted packets after considering the source packet loss rate, i.e., the effective transmission rate. $R_{(l_i^v, l^p)}$ denotes the reliability of the substrate link l^p, which is mapped by the ith virtual link l_i^v.

3.3.3. Revenue-cost ratio

The revenue-cost ratio is a classic evaluation metric in VNE problems, which incorporates revenue and cost. The concepts of cost and revenue are not considered separately in our experimental evaluation, as our main focus is on the combined revenue. The revenue-cost ratio is a key measure of our algorithm's performance, which combines information from both cost and revenue dimensions. We use this ratio as an evaluation metric rather than costs and revenues alone because it provides a more comprehensive picture of the economic benefits and resource utilization efficiency of the VNE process.

The revenue generated is intrinsically linked to the CPU and bandwidth demands of the VNR. Consequently, a formula representing this revenue can be derived from the resource requirements of a successfully allocated VNR at a given time point t:

$$Rev(G^V, t) = \alpha \sum_{n_i^v}^{N^V} CPU_{n_i^v} + \beta \sum_{l_j^v}^{L^V} BW_{l_j^v}. \tag{10}$$

in formula (10), α and β are balancing factors between CPU resources and bandwidth resources. In practice, the balancing factor should be set to the service provider's price.

Every deployment of a VN necessitates the allocation of requisite resources from the underlying physical network to support the corresponding bearer. In this context, the present paper defines the cost associated with deploying a VN as the cumulative expense incurred by the physical network in accepting and operating each VNR.

$$Cost(G^V, t) = \alpha \sum_{n_i^v}^{N^V} CPU_{n_i^v} + \beta \sum_{l_j^v}^{L^V} BW_{l_j^v} Hop(l_j^v). \tag{11}$$

$Hop(l_j^v)$ represents the count of hops within the physical path allocated to the underlying physical network by the virtual link. The revenue function only considers the CPU and link bandwidth requirements of the VNR, the cost function also needs to consider the number of hops in the physical path. This reflects the resource consumption required to actually deploy the virtual network in the physical network. It reflects not only the performance of the algorithm in terms of resource utilization efficiency, but also the advantage of the algorithm in terms of economic efficiency. The concept of long-term average revenue is formally articulated as the asymptotic limit of the ratio of total revenue to the time span T, as T approaches positive infinity, expressed as follows:

$$Rev = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} Rev(G^V, t)}{T}. \tag{12}$$

analogous to Rev, the long-term average cost is articulated in the following manner:

$$Cost = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} Cost(G^V, t)}{T}. \tag{13}$$

The expression for the revenue-cost ratio is as follows:

$$R/C = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} Rev(G^V, t)}{\sum_{t=0}^{T} Cost(G^V, t)}. \tag{14}$$

3.3.4. VNR acceptance

In alignment with the definition provided in [30], the acceptance rate for VNRs is defined as follows:

$$AR = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} |VNR|_{acc}}{\sum_{t=0}^{T} |VNR|_{arr}}. \tag{15}$$

in formula (15), $|VNR|\,acc$ represents the quantity of VNRs that have been successfully mapped onto the physical network. $|VNR|\,arr$ denotes the aggregate count of all arriving VNRs.

4. DRL and LS in space-based backbone networks

This section is dedicated to an in-depth exposition of the algorithmic modeling. We have developed a framework centered around a policy network. This framework enables an agent to extract feature matrices from the physical network, encapsulating comprehensive details of the network topology, node attributes, and other pertinent information. Utilizing these features, the policy network is designed to generate the embedding probabilities for the substrate nodes. Concurrently, a local search mechanism is incorporated to enhance the resource allocation strategy. Leveraging the probability distribution generated by the policy network, local search operations are conducted within the immediate neighborhood. This process aims to further refine and optimize the initial output results (see Fig. 3).

4.1. Feature extraction

There are very many attributes of the substrate nodes, for which we have selected a few important attributes as extracted features.

- Node CPU capability: Node CPU capacity refers to the remaining computational capacity of the substrate node itself and is a factor of the node's own capacity. Expressed in formula (16).

$$CPU(n^p) = CPU_{n^p} - \sum_{n_i^v}^{N^V} CPU_{n_i^v} F1(n_i^v, n^p). \tag{16}$$

  where $F1(n_i^v, n^p)$ represents the function value of the virtual node n_i^v embedded in the substrate node n^p.
- Node communication capability: The communication capacity of a substrate node is quantified as the aggregate of the available bandwidth across all substrate links directly interconnected with the substrate node. This metric effectively mirrors the node's communication capability. Define $l^p \oplus n^p$ as the substrate link l^p that is

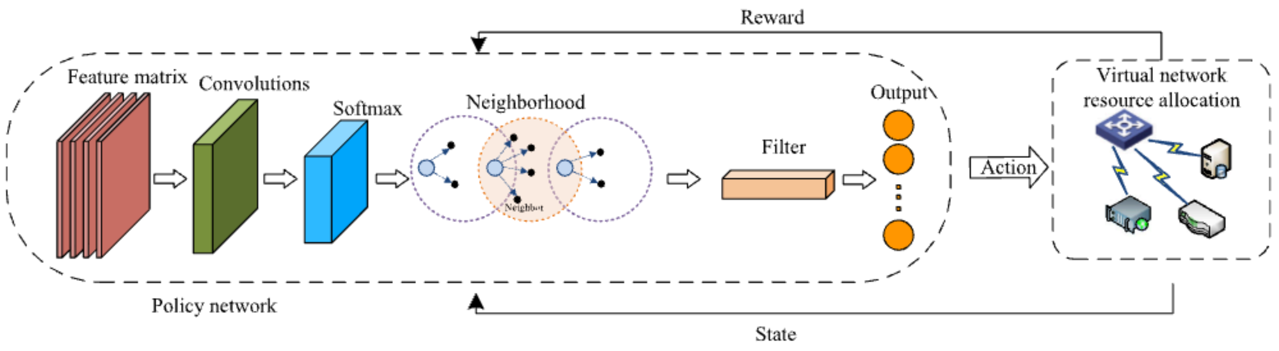

Fig. 3. An example of two-stage training process.

directly connected to the substrate node n^p. The communication capacity of the node is then mathematically represented by the following formula:

$$COM(n^p) = \sum_{\forall l^p \oplus n^p} BW_{l^p} - \sum_{\forall l^p \oplus n^p} \sum_{l_i^v}^{L^V} BW_{l_i^v} F2(l_i^v, l^p). \tag{17}$$

- Node reliability capability: The reliability of individual substrate links is subject to variation. Consequently, the reliability capacity of a node can be defined as the average reliability of all substrate links that are directly connected to it, as detailed in formula (18).

$$REL(n^p) = \frac{\sum_{\forall l^p \oplus n^p} R_{l^p}}{Deg(n^p)} \tag{18}$$

where $Deg(n^p)$ denotes the degree of the substrate node n^p, which is defined as the count of links that are directly connected to this particular substrate node.

In fact, there are more node attributes that can be extracted, and while extracting more node-related attributes will make the results more accurate, it will also increase the complexity of the program. Our purpose at the beginning is to use the solution obtained through the policy network as the initial solution for local search, so we only need that the quality of the initial solution is not too low, and obviously, the extraction of the three features is enough to satisfy this criterion.

after extracting these feature attributes, we normalize them to be bounded within the range of 0 and 1 using Min-Max Normalization, then get the feature vector v_i of the ith node features:

$$v_i = (CPU(n_i^p), COM(n_i^p), REL(n_i^p))^{\mathrm{T}}. \tag{19}$$

The substrate node feature matrix, denoted as M_f, is formulated by concatenating the feature vectors corresponding to all the substrate nodes:

$$M_f = (v_1, v_2, \dots, v_n)^{\mathrm{T}}. \tag{20}$$

within this matrix, each row corresponds to the feature vector of a specific substrate node.

4.2. Policy network

The policy network employed in our study bears resemblance to the frameworks presented in [31,32], comprising an input layer, a convolutional layer, and a Softmax layer. Our primary objective in this work is to investigate the impact of incorporating a local search mechanism into the VNE process. To isolate the effects of this addition, we opted to maintain the same policy network structure as in our previous work [32]. However, a notable divergence in our approach is the incorporation of filters for substrate nodes within the local search. This modification stems from our hypothesis that certain substrate nodes, despite not meeting the embedding constraints but exhibiting higher embedding probabilities, have a greater degree of correlation with their neighboring substrate nodes. Such substrate nodes, therefore, warrant further exploration in subsequent steps of the algorithm.

- Input layer: Calculate the feature matrix M_f and subsequently transmit it to the policy network.
- Convolutional layer: In the convolutional layer, the convolutional kernels execute convolution operations on the feature matrix, aiming to evaluate the available resources of each substrate node. The activation function corresponding mathematical expression is delineated as follows:

$$h_i = ReLU(\omega_n \cdot v_i + b_n), \tag{21}$$

- Softmax layer: Normalize the vector representing available resources for each substrate node, ascertained via convolution, to deduce the embedding probability of that specific substrate node. This process is mathematically articulated in the formula (22):

$$p_i = \frac{e^{h_i}}{\sum_{j=1}^{n} e^{h_j}}, \tag{22}$$

The probability vector of candidate substrate nodes P is obtained after Softmax layer:

$$P = (p_1, p_2, \dots, p_n). \tag{23}$$

Reward function: The reward function provides a focus for reinforcement learning to evaluate performance, guiding the intelligence to continuously learn and optimize the strategy in the direction of the set goal. In this paper, we use the revenue-cost ratio of mapping a single VNR as the mapping reward for that VNR:

$$Reward = \begin{cases} \frac{Rev(G^v, |VNR|)}{Cost(G^v, |VNR|)} & ,\text{mapped} \\ 0 & ,\text{else} \end{cases} \tag{24}$$

where $Rev(G^v, |VNR|)$ is the revenue in a single VNR, similarly, $Cost(G^v, |VNR|)$ is the cost in a single VNR.

4.3. Neighborhood selection

For the VNRs in our study, the requisite number of substrate nodes is established within a range of 2 to 7. Additionally, we set the initial neighborhood range RA_{init} to 3, defined as the neighborhood $N(n_i^p, 3)$ of substrate node n_i^p as the set of all substrate nodes that are not more than 3 substrate nodes away from n_i^p.

$$N(n_i^p, 3) = \left\{ n_j^v | \; dis(n_j^v, n_i^v) \le 3 \right\} \tag{25}$$

Fig. 4 presents a straightforward example illustrating a substrate node's neighborhood range. In this instance, the neighborhood range is segmented into four distinct layers, with the respective neighborhood

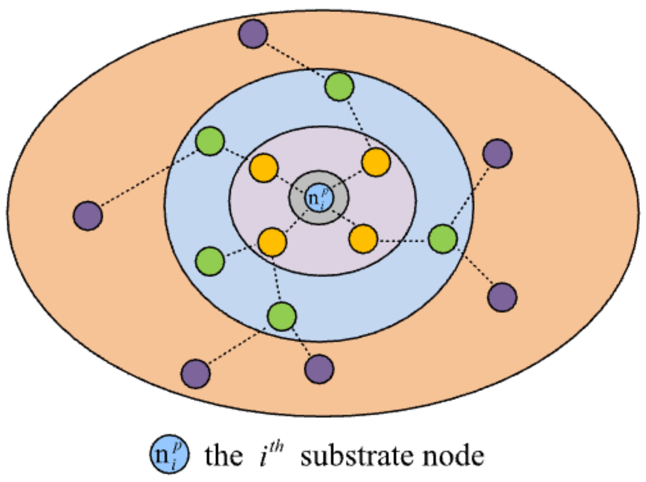

Fig. 4. Four-level neighborhood range of the *i*th substrate node.

ranges RA being 0, 1, 2, and 3, progressing from the innermost to the outermost layer. Practically, it is feasible for a substrate node to concurrently exist within multiple levels of another substrate node's neighborhood.

Should the current neighborhood exceed the number of substrate nodes necessitated by the VNR, an adaptive reduction of the neighborhood is implemented. Moreover, a decay factor ε is established to progressively diminish the neighborhood in proportion to the increasing number of training iterations. This concept is mathematically expressed as follows:

$$RA_{curr} = \lfloor RA_{init} * \varepsilon^{\frac{n-T}{N}} \rfloor \tag{26}$$

in formula (26), ε is between 0 and 1, n and N are the current training number as well as the total training number, respectively. where T is a positive integer indicating that the neighborhood range was kept at RA_{init} for the first T training sessions.

To enhance the quality of the solution, it is posited that the initial solution derived from policy network is of reasonable quality. Consequently, minor adjustments are made to the probabilities of each substrate node within the solution that falls in the neighborhood. The specific mathematical formulation is as follows:

$$p_i^{new} = p_i - \sigma DST(n_i^p) + \eta EC(n_i^p) + \lambda DC(n_i^p) \tag{27}$$

in formula (27), the terms $DST(n_i^p)$, $EC(n_i^p)$, and $DC(n_i^p)$ respectively signify the average distance to other substrate nodes, eigenvector centrality, and degree centrality. The expressions for these parameters are delineated as follows:

$$DST(n^p) = \frac{\sum_{n_i^p}^{N^P} hops(n^p, n_i^p)}{|N(n^p)|+1}, \tag{28}$$

$$EC(n^p) = \frac{\sum_{n_i^p}^{M(n^p)} EC(n_i^p)}{\xi}, \tag{29}$$

$$DC(n^p) = \frac{Deg(n^p)}{m-1}, \tag{30}$$

within this context, $N(n^p)$ refers to the set of virtual nodes embedded onto the substrate nodes n^p. $hops(n^p, n_i^p)$ indicates the number of hops from substrate node n^p to node n_i^p. Furthermore, $M(n^p)$ is defined as the set of nodes that are directly connected to node n^p, and m denotes the total count of substrate nodes.

Upon concluding the node search iterations within the specified neighborhood, a revised candidate node probability vector, denoted as P^{new}, is derived.

4.4. Training process

In this study, we employ a gradient-based strategy to train the node policy network. The output nodes' probability vectors are subsequently refined through local search, aiming to enhance the quality of the solution space. The training methodology is detailed in [Algorithm 1].

Algorithm 1 Training process.

```
Input: $G^P, G^V, \varepsilon, RA_{init}, epochNum$, Training set;
Output: Policy Network $\omega^n$;
1: while $i \le epochNum$ do
2:     for $vnr \in VNRs$ do
3:         for $n^v \in vnr$ do
4:             $M_f \leftarrow \emptyset$;
5:             for $n_i^p \in G^P$ do
6:                 $v_i = (CPU(n_i^p), COM(n_i^p)$
                   $REL(n_i^p))^T$;
7:                 $M_f \leftarrow M_f + v_i$;
8:             end for
9:             Calculate the probability $P$;
10:            Calculate the current range $RA_{curr}$;
11:            Select the top $RA_{curr}$ nodes $N_{RA}$ with
               the highest probabilities;
12:            for $n_i^p \in N_{RA}$ do
13:                for $r = RA_{curr}$ to 0 do
14:                    for $n_j^p \in N(n_i^p, r)$ do
15:                        $p_j^{new} = p_j - \sigma DST(n_j^p) +$
                           $\eta EC(n_j^p) + \lambda DC(n_j^p)$;
16:                    end for
17:                    $r--$;
18:                end for
19:            end for
20:            e-Greedy select node in $P^{new}$;
21:            Update the $G^P$;
22:        end for
23:        BFS link map for $vnr$;
24:        if mapped then
25:            Calculate reward and gradient;
26:        end if
27:        Update parameters in $\omega^n$;
28:    end for
29: $i++$;
30: end while
```

[Algorithm 1] describes the training process of VNE. First, the input and output parameters of the algorithm are defined, including the physical network, the set of virtual network requests, the decay factor, the initial neighborhood range, the number of training rounds and the final output of the trained policy network. For each VNR denoted by vnr, the algorithm iterates over all its virtual nodes n^v. For each virtual node, the algorithm first initializes an empty feature matrix M_f. Then, it iterates over all nodes n_i^p in the physical network, computes the feature vector v_i for each node, and adds it to the feature matrix. Next, the algorithm calculates the embedding probabilities P for each physical node based on the feature matrix and determines the current neighborhood range RA_{curr} based on the current training epoch. The algorithm selects the top RA_{curr} nodes N_{RA} with the highest embedding probabilities and iterates through their neighborhoods, gradually reducing the neighborhood range r, while adjusting the embedding probabilities of the nodes based on their DST, EC, and DC. Finally, the algorithm uses an e-Greedy strategy to select nodes from the updated probability vector P^{new} for embedding and updates the physical network G^P. Afterwards a Breadth-First Search (BFS) is performed on the virtual network request vnr to map the virtual

Table 3
Experimental environment parameters.

Network	Parameter	Value
Physical network	Substrate nodes	100
	Substrate links	550
	CPU resource	50 ~ 80
	Node pocket loss rate	10e−2 ~ 10e−4
	Link bandwidth resource	50 ~ 80
VNRs	The number of VNR	2000
	Training set	1000
	Testing set	1000
	Virtual nodes	2 ~ 7
	Node connection probability	0.5
	CPU resource requirement	1 ~ 30
	Bandwidth resource requirement	1 ~ 30
	Reliability requirement	47 ~ 99

links. If the VNR is successfully mapped, the algorithm calculates the reward and gradient and updates the parameters of the policy network ω^n. The entire training process continues until the specified number of training epochs is reached.

5. Experimental results and analysis

This section commences with an introduction to the experimental environment. Subsequently, a detailed description of the comparison algorithm is provided. The section concludes with an analysis of the numerical results.

5.1. Environment and parameter setting

In this study, the GT-ITM tool [33] is utilized for generating a randomized network topology, facilitating the construction of a network environment comprising 100 nodes and 550 links. Table 3 provides a comprehensive breakdown of the attribute information for both nodes and links. The dataset employed in our research encompasses a total of 2000 VNRs, evenly divided between training and testing samples. Each VNR incorporates 2 to 7 nodes, featuring a 50% probability of inter-node linkage. For detailed insights into the physical network and VNR characteristics, refer to Table 3.

5.2. Comparison algorithms

Regarding the choice of comparison algorithms, we chose three algorithms to compare with ours.

- RLVNE [31]: A Reinforcement Learning-based VNE algorithm (RLVNE) that optimizes node and link mappings using policy gradients. This research represents the inaugural effort to optimize VNE by leveraging historical request data in conjunction with a policy network-based reinforcement learning approach. The methodology employs policy gradients and backpropagation techniques to train policy networks, utilizing historical request data as the foundational training input.
- GCN [34]: A VNE algorithm that utilizes a Graph Convolutional Network (GCN) to capture the topological features of the network. This study introduces a VNE algorithm that uniquely integrates DRL with an innovative neural network architecture grounded in GCN.
- HPSO [35]: A Hybrid with Particle Swarm Optimization approach (HPSO) for VNE. This approach presents a VNE method that innovatively incorporates simulated annealing into the Particle Swarm Optimization algorithm. It features a distinctive particle initialization distribution strategy to effectively disperse the particles.

5.3. Experimental results analysis

5.3.1. Training results

Validation of training results: In order to validate the appropriateness of the reward configuration within the local search mechanism, an initial uniform weight of 0.05 for each attribute in formula (27) was set, along with a uniform learning rate of 0.005, an attenuation factor ε of 0.9, and an initial step size T of 300. We documented the Revenue to Cost (R/C) values across iterations, employing varying initial ranges for the local search (3, 0). This comparative analysis aims to ascertain the validity of the reward settings and to determine if the local search contributes to expediting the agent's training process.

Fig. 5 depicts the fluctuation of R/C values for different initial ranges throughout the training period. In the early stages of training, due to the stochastic nature of network parameters and the agent's initial unfamiliarity with the network environment, the policy network's initial solutions are of moderate quality and exhibit considerable variance, even with local search intervention. As training epochs progress, the agent gradually acclimates to the network, leading to enhanced quality of initial solutions and reduced amplitude in local searches. This results in a more rapid attainment of stability compared to scenarios without local search. Subsequently, as the neighborhood range narrows, the disparity between the two approaches diminishes, culminating in eventual stabilization. Importantly, the overall trend of R/C values is an upward trajectory amidst the fluctuations, indicating an improvement in the efficiency of resource utilization. As the neighborhood range narrows with further training, the disparity between the approaches with and without local search diminishes, culminating in eventual stabilization of the R/C values. The experimental findings demonstrate that local search not only accelerates the agent's training but also enhances the stability of the training outcomes, ultimately leading to a steady improvement in the revenue-cost ratio as the training epochs increase.

5.3.2. Comparative experimental results

Experiment 1: Comparison of long-term average node utilization

Fig. 6 illustrates a detailed comparative analysis of average node utilization utilizing four distinct methodologies. The long-term average node utilization of the four algorithms shows an overall increasing trend with time. Initially, all the algorithms have a low utilization rate, but over time, the utilization rate gradually increases and approaches a smooth state. The two-stage algorithm we proposed shows marked superiority in the realm of node mapping optimization. Its initial phase, characterized by a fixed neighborhood range, offers an optimal exploratory trajectory, especially in scenarios where the network environment is not yet well-understood, thus facilitating a rapid increase in node utilization. A comprehensive quantitative evaluation indicates that our method achieves a higher node utilization rate than GCN by 20.11%, RLVNE by 37.14%, and HPSO by 37.19%. Experiment 2: Comparison of long-term average link utilization

Fig. 7 provides a comparative evaluation of average link utilization utilizing four different methodologies. The long-term average link utilization of the four algorithms also shows an increasing trend over time. It can be seen that our algorithm shows the most significant increase and reaches a relatively high point in the second half of the graph, showing a more stable high utilization. Our method, which prioritizes link reliability during the mapping process, demonstrates a notably higher link utilization rate. This improvement can be attributed to the superior quality of our node mapping approach. A thorough analysis of the data reveals that our approach achieves an increase in link utilization by 5.94% compared to GCN, 9.85% over RLVNE, and 15.33% over HPSO. Experiment 3: Comparison of long-term revenue-cost ratio

Fig. 8 delineates the trend in revenue-cost ratios for the four evaluated methods, where each exhibits a declining trajectory prior to achieving stability. The trends of the long-term Revenue-Cost Ratio (R/C) of the algorithms over time show some differences. Our algorithm

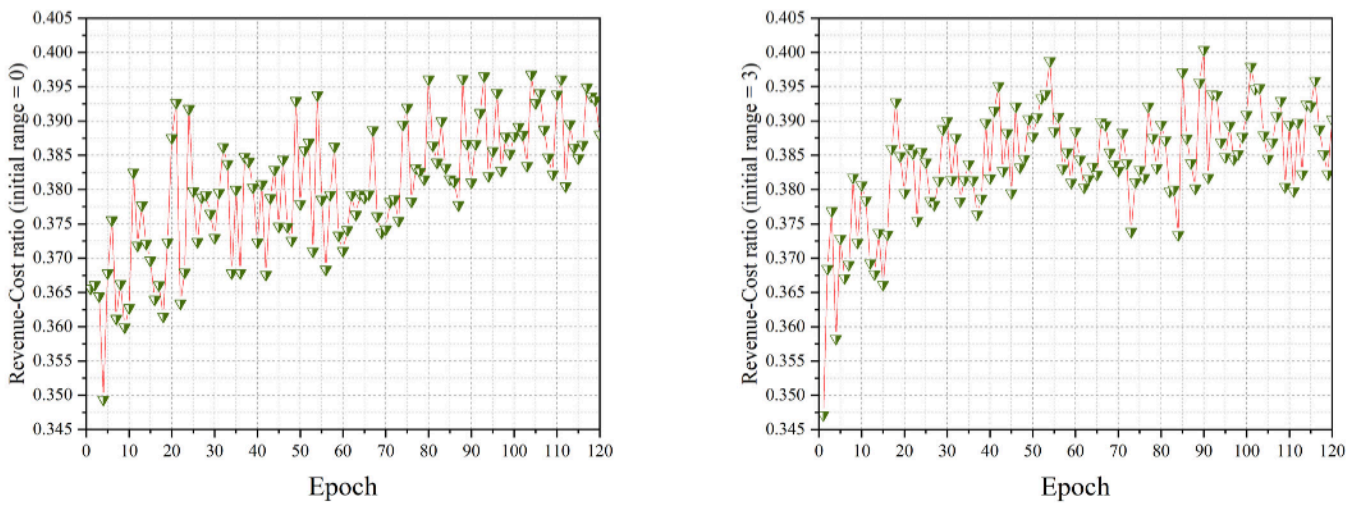

Fig. 5. Revenue-Cost ratio in different initial ranges.

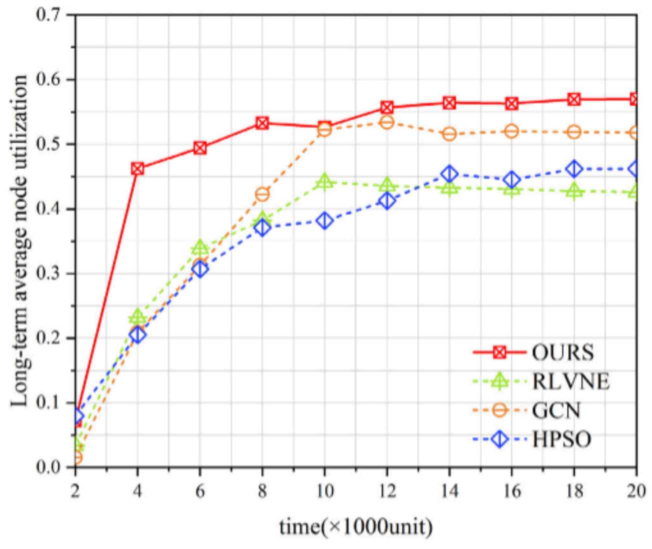

Fig. 6. Long-term node utilization ratio comparison.

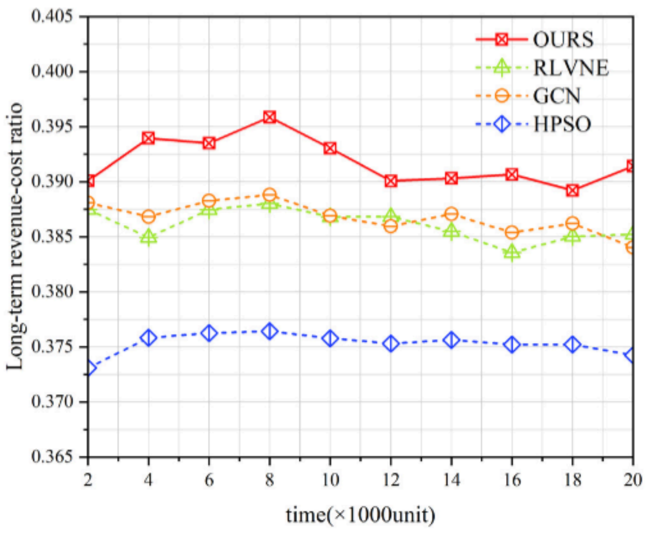

Fig. 8. Long-term revenue-cost ratio comparison.

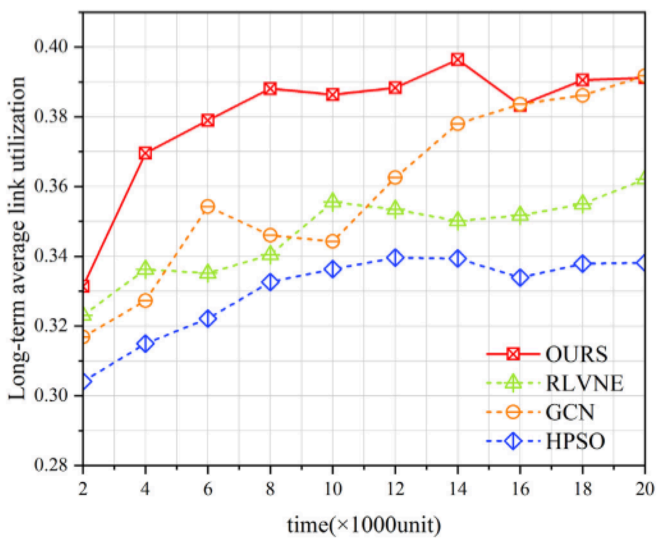

Fig. 7. Long-term link utilization ratio comparison.

shows significant stability and superiority, especially in the middle and late stages of training, where the R/C ratio increases and stabilizes. In contrast, the R/C ratio of the RLVNE algorithm fluctuates but generally stays in a more stable interval. The GCN algorithm rises faster in the initial stage and then stabilizes, while the R/C ratio of the HPSO is always at a low level with little change. Notably, our methodology registers a marginally superior revenue-cost ratio, surpassing GCN by 1.31%, RLVNE by 1.48%, and HPSO by 4.40%. Experiment 4: Comparison of VNR acceptance ratio

Fig. 9 showcases the trend in VNR acceptance ratios across all algorithms under consideration. We can observe an overall decreasing trend in the VNR acceptance rate of various algorithms. Nonetheless, our algorithm consistently maintains a relatively high acceptance rate and outperforms the other algorithms even if it declines in the long-term trend. In contrast, the RLVNE algorithm suffered a significant decline over time, which may point to the instability of its performance in specific cases. The other two algorithms-GCN and HPSO-although the decline trend is less drastic, their acceptance rates are significantly lower than those of our algorithm. It is observed that our method and RLVNE demonstrate comparable acceptance rates, both outstripping GCN by 2.34% and HPSO by 4.45%.

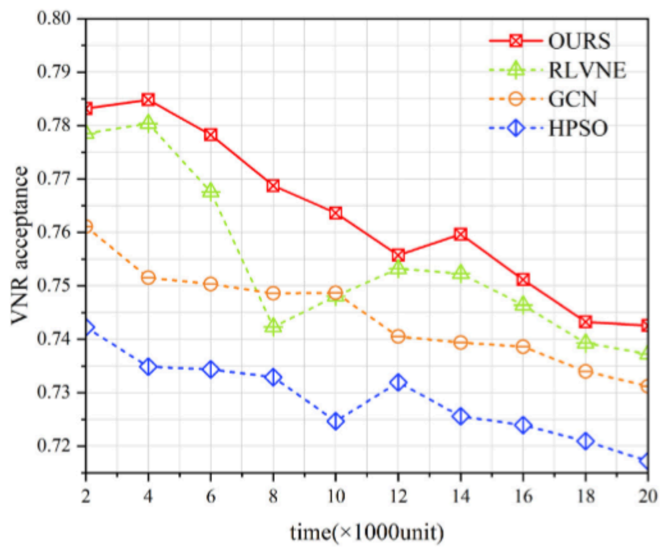

Fig. 9. VNR acceptance ratio comparison.

6. Conclusion and future work

This research introduces a novel two-stage algorithm that synergizes DRL with LS for the purpose of VNE. Our findings highlight the significance of integrating lightweight learning intelligence into traditional methodologies as a critical strategy for crafting robust and scalable network optimization algorithms. It is, however, pertinent to acknowledge that neighbor selection and definition are inherently problem-specific, leading to solutions that may not be universally applicable. Looking ahead, our research will pivot in two principal directions. Firstly, we aim to adapt this method for more sophisticated neighborhood configurations, addressing more complex network challenges like modulation formats and spectrum allocation in Elastic Optical Networks, and the optimization of Service Function Chains. Secondly, we endeavor to explore the potential of utilizing a DRL agent to supplant manual neighborhood scoring rules. This would enable the agent to autonomously determine the direction and assessment of neighborhoods, thereby striving to develop a methodology with broader applicability.

CRediT authorship contribution statement

Peiying Zhang: Writing – review & editing. **Zixuan Cui:** Writing – original draft. **Neeraj Kumar:** Supervision. **Jian Wang:** Data curation. **Wei Zhang:** Resources. **Lizhuang Tan:** Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

[1] Zhenhua Liu, Chuanwen Lin, Gang Chen, Development and trend of space-based information network, in: Journal of Physics: Conference Series, Vol. 1544, IOP Publishing, 2020, 012180.

[2] N.T. Zhang, K.L. Zhao, G.L. Liu, Thought on constructing the integrated space-terrestrial information network, J. China Acad. Electron. Inf. Technol. 10 (3) (2015) 223–230.

[3] Peiying Zhang, Chao Wang, Chunxiao Jiang, Neeraj Kumar, Qinghua Lu, Resource management and security scheme of ICPSs and IoT based on VNE algorithm, IEEE Internet Things J. 9 (22) (2021) 22071–22080.

[4] Wai-xi Liu, Jun Cai, Qing Chun Chen, Yu Wang, DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks, J. Netw. Comput. Appl. 177 (2021) 102865.

[5] Nicola Di Cicco, Emre Furkan Mercan, Oleg Karandin, Omran Ayoub, Sebastian Troia, Francesco Musumeci, Massimo Tornatore, On deep reinforcement learning for static routing and wavelength assignment, IEEE J. Sel. Top. Quantum Electron. 28 (4: Mach. Learn. in Photon. Commun. and Meas. Syst.) (2022) 1–12.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.

[7] Zhihao Shen, Kang Yang, Wan Du, Xi Zhao, Jianhua Zou, Deepapp: a deep reinforcement learning framework for mobile application usage prediction, in: Proceedings of the 17th Conference on Embedded Networked Sensor Systems, 2019, pp. 153–165.

[8] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, Samy Bengio, Neural combinatorial optimization with reinforcement learning, 2016, arXiv preprint arXiv:1611.09940.

[9] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, Martin Takác, Reinforcement learning for solving the vehicle routing problem, Adv. Neural Inf. Process. Syst. 31 (2018).

[10] Hanjun Dai, Bo Dai, Le Song, Discriminative embeddings of latent variable models for structured data, in: International Conference on Machine Learning, PMLR, 2016, pp. 2702–2711.

[11] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, Le Song, Learning combinatorial optimization algorithms over graphs, Adv. Neural Inf. Process. Syst. 30 (2017).

[12] Wouter Kool, Herke Van Hoof, Max Welling, Attention, learn to solve routing problems!, 2018, arXiv preprint arXiv:1803.08475.

[13] J.Q. James, Wen Yu, Jiatao Gu, Online vehicle routing with neural combinatorial optimization and deep reinforcement learning, IEEE Trans. Intell. Transp. Syst. 20 (10) (2019) 3806–3817.

[14] Alvaro H.C. Correia, Daniel E. Worrall, Roberto Bondesan, Neural simulated annealing, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 4946–4962.

[15] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi, Exact combinatorial optimization with graph convolutional neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).

[16] Mosharaf Chowdhury, Muntasir Raihan Rahman, Raouf Boutaba, Vineyard: Virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Netw. 20 (1) (2011) 206–219.

[17] Marcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, Jorge Carapinha, Optimal virtual network embedding: Node-link formulation, IEEE Trans. Netw. Serv. Manag. 10 (4) (2013) 356–368.

[18] Zeheng Yang, Yongan Guo, An exact virtual network embedding algorithm based on integer linear programming for virtual network request with location constraint, China Commun. 13 (8) (2016) 177–183.

[19] Qian Hu, Yang Wang, Xiaojun Cao, Virtual network embedding: An optimal decomposition approach, 2014, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2014.6911723.

[20] Sarang Bharadwaj Masti, Serugudi V. Raghavan, Simulated annealing algorithm for virtual network reconfiguration, in: Proceedings of the 8th Euro-NF Conference on Next Generation Internet NGI 2012, IEEE, 2012, pp. 95–102.

[21] Sheng Zhang, Zhuzhong Qian, Song Guo, Sanglu Lu, FELL: A flexible virtual network embedding algorithm with guaranteed load balancing, in: 2011 IEEE International Conference on Communications, ICC, IEEE, 2011, pp. 1–5.

[22] Marieme Diallo, Alejandro Quintero, Samuel Pierre, An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers, IEEE Trans. Cloud Comput. 9 (3) (2019) 896–909.

[23] Marcelo Caggiani Luizelli, Weverton Luis da Costa Cordeiro, Luciana S Buriol, Luciano Paschoal Gaspary, A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining, Comput. Commun. 102 (2017) 67–77.

[24] Peiying Zhang, Chao Wang, Gagangeet Singh Aujla, Xue Pang, A node probability-based reinforcement learning framework for virtual network embedding, in: 2020 IEEE 21st International Symposium on "a World of Wireless, Mobile and Multimedia Networks", WoWMoM, 2020, pp. 421–426.

[25] Haitham Afifi, Holger Karl, Reinforcement learning for virtual network embedding in wireless sensor networks, in: 2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob, 2020, pp. 123–128.

[26] Peiying Zhang, Chao Wang, Chunxiao Jiang, Neeraj Kumar, Qinghua Lu, Resource management and security scheme of ICPSs and IoT based on VNE algorithm, IEEE Internet Things J. 9 (22) (2022) 22071–22080.
[27] Haotong Cao, Shengchen Wu, Yue Hu, Yun Liu, Longxiang Yang, A survey of embedding algorithm for virtual network embedding, China Commun. 16 (12) (2019) 1–33.
[28] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, Introduction to Algorithms, MIT Press, 2022.
[29] Xiaoyuan Fu, F. Richard Yu, Jingyu Wang, Qi Qi, Jianxin Liao, Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach, IEEE Trans. Wireless Commun. 19 (1) (2020) 507–519.
[30] Peiying Zhang, Chao Wang, Neeraj Kumar, Weishan Zhang, Lei Liu, Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning, IEEE Internet Things J. 9 (12) (2021) 9389–9398.
[31] Haipeng Yao, Xu Chen, Maozhen Li, Peiying Zhang, Luyao Wang, A novel reinforcement learning algorithm for virtual network embedding, Neurocomputing 284 (2018) 1–9.
[32] Peiying Zhang, Yuanjie Li, Neeraj Kumar, Ning Chen, Ching-Hsien Hsu, Ahmed Barnawi, Distributed deep reinforcement learning assisted resource allocation algorithm for space-air-ground integrated networks, IEEE Trans. Netw. Serv. Manag. (2022).
[33] Ellen W. Zegura, Kenneth L. Calvert, Samrat Bhattacharjee, How to model an internetwork, in: Proceedings of IEEE INFOCOM'96. Conference on Computer Communications, Vol. 2 (1996) 594–602.
[34] Zhongxia Yan, Jingguo Ge, Yulei Wu, Liangxiong Li, Tong Li, Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks, IEEE J. Sel. Areas Commun. 38 (6) (2020) 1040–1057.
[35] Peiying Zhang, Yanrong Hong, Xue Pang, Chunxiao Jiang, VNE-HPSO: Virtual network embedding algorithm based on hybrid particle swarm optimization, IEEE Access 8 (2020) 213389–213400.

Peiying Zhang is currently an Associate Professor with the College of Computer Science and Technology, China University of Petroleum (East China). He received his Ph.D. in the School of Information and Communication Engineering at University of Beijing University of Posts and Telecommunications in 2019. He has published multiple IEEE/ACM Trans./Journal/Magazine papers since 2016, such as IEEE TII, IEEE T-ITS, IEEE TVT, IEEE TNSE, IEEE TNSM, IEEE TETC, IEEE Network and etc. He served as the Technical Program Committee of AAAI'24, AAAI'23, IEEE ICC'23, IEEE ICC'22, and INFOCOM Wireless-Sec 2023. He is the Leading Guest Editor of Drones, Mathematics, Electronics, Wireless Communications and Mobile Computing, and etc. He is the editorial board of Drones, CMC-Computers, Materials & Continua, Mobile Information Systems, International Journal of Computational Intelligence Systems and Artificial Intelligence and Applications (AIA). His research interests include semantic computing, future internet architecture, network virtualization, and artificial intelligence for networking.

Zixuan Cui is currently studying for a master's degree in the College of Computer Science and Technology, China University of Petroleum (East China). His research interests include network virtualization and artificial intelligence for networking.

Neeraj Kumar (Senior Member, IEEE) is currently working as a Full Professor with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India. He has published more than 400 technical research papers, which are cited more than 43637 times by researchers across the globe with a current H-index of 114. His broad research areas are green computing and network management, IoT, big data analytics, deep learning, and cyber-security. He has also edited/authored ten books with international/national publishers. He is a highly-cited researcher from WoS from 2019 to 2021. Prof. Kumar has organized various special issues of journals of repute from IEEE, Elsevier, and Springer. He has been the Workshop Chair at IEEE GLOBECOM 2018, IEEE INFOCOM 2020, and IEEE ICC 2020, and the track Chair. He is serving as an Editor of ACM Computing Surveys, IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, IEEE Network Magazine, IEEE Communications Magazine, Journal of Networks and Computer Applications (Elsevier), Computer Communications (Elsevier), and International Journal of Communication Systems (Wiley).

Jian Wang is currently a Professor and serves as the Head of the Laboratory for Intelligent Information Processing with the College of Science, China University of Petroleum (East China). His research interests include computational intelligence, differential programming, clustering, fuzzy systems, evolutionary computation. Prof. Wang serves as an Associate Editor for the IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Emerging Topics in Computational Intelligence, Information Sciences, International Journal of Machine Learning and Cybernetics, and Journal of Applied Computer Science Methods. He also serves on the Editorial Board for the Neural Computing & Applications and Complex & Intelligent Systems.

Wei Zhang is currently a Professor with Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences). He received the B.E. degree from Zhejiang University in 2004, the M.S. degree from Liaoning University in 2008, and the Ph.D. degree from Shandong University of Science and Technology in 2018. He has published more than 50 journal or conference papers, such as IEEE TCC, IEEE TMM, IEEE TKDE, IEEE TNSM, ELSEVIER FGCS, ELSEVIER CN, IEEE INFOCOM, etc. He holds more than 10 US patents and Chinese patents. He is a member of IEEE and CCF. His research interests include future generation network architectures, edge computing and edge intelligence.

Lizhuang Tan is currently an Assistant Professor with Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences). He received his Ph.D. degree from School of Electronic and Information Engineering, Beijing Jiaotong University in 2022. He has published more than 20 journal or conference papers, such as USENIX NSDI, IEEE TNSM, ELSEVIER CN, APNOMS, etc. His research interests include network measurement, management and optimization, especially software-defined networking and data center networking.