electronics

Article

Network Resource Allocation Algorithm Using Reinforcement Learning Policy-Based Network in a Smart Grid Scenario

Zhe Zheng [1], Yu Han [2,*], Yingying Chi [1], Fusheng Yuan [1], Wenpeng Cui [1], Hailong Zhu [3], Yi Zhang [4] and Peiying Zhang [4]

[1] Beijing Smartchip Microelectronics Technology Company Ltd., Beijing 100192, China; zhengzhe@sgchip.sgcc.com.cn (Z.Z.); chiyingying@sgchip.sgcc.com.cn (Y.C.); yuanfusheng@sgchip.sgcc.com.cn (F.Y.); cuiwenpeng@sgchip.sgcc.com.cn (W.C.)
[2] China Mobile Group Shandong Co., Ltd., Jinan 250001, China
[3] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; zhuhl@bupt.edu.cn
[4] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; zhangyi@s.upc.edu.cn (Y.Z.); zhangpeiying@upc.edu.cn (P.Z.)
* Correspondence: hanyu@sd.chinamobile.com

Abstract: The exponential growth in user numbers has resulted in an overwhelming surge in data that the smart grid must process. To tackle this challenge, edge computing emerges as a vital solution. However, the current heuristic resource scheduling approaches often suffer from resource fragmentation and consequently get stuck in local optimum solutions. This paper introduces a novel network resource allocation method for multi-domain virtual networks with the support of edge computing. The approach entails modeling the edge network as a multi-domain virtual network model and formulating resource constraints specific to the edge computing network. Secondly, a policy network is constructed for reinforcement learning (RL) and an optimal resource allocation strategy is obtained under the premise of ensuring resource requirements. In the experimental section, our algorithm is compared with three other algorithms. The experimental results show that the algorithm has an average increase of 5.30%, 8.85%, 15.47% and 22.67% in long-term average revenue–cost ratio, virtual network request acceptance ratio, long-term average revenue and CPU resource utilization, respectively.

Keywords: smart grid; edge computing; resource allocation; multi-domain virtual network; reinforcement learning

Citation: Zheng, Z.; Han, Y.; Chi, Y.; Yuan, F.; Cui, W.; Zhu, H.; Zhang, Y.; Zhang, P. Network Resource Allocation Algorithm Using Reinforcement Learning Policy-Based Network in a Smart Grid Scenario. *Electronics* **2023**, *12*, 3330. https://doi.org/10.3390/electronics12153330

Academic Editors: Ahmed Abu-Siada and Antonio Brogi

Received: 3 July 2023
Revised: 26 July 2023
Accepted: 1 August 2023
Published: 3 August 2023

1. Introduction

Smart grids are based on high-speed communication networks, advanced sensing devices and algorithms which are used to realize low delay, high quality of service, high flexibility, security, and green, reliable and intelligent application of power systems [1]. Among them, calculation determines the execution method of smart grid data analysis and is the basis of smart grid services; the data volume of smart grids is also growing rapidly [2–4]. Edge computing is one of the best ways to solve the power grid data explosion problem and smart power grids are considered as one of the best landing sites for edge computing [5]. Edge computing pushes some computing applications from the center to the edge of the communications network. Unload technology is used to unload part of the computing logic to the edge of the network near the device and the client to perform the real-time data analysis task, and it can meet the demand of large bandwidth, reduce processing load on computing centers, and enhance the scalability and availability of the system [6]. Edge computing has been integrated with smart grids through chip and communication technology, empowering them with ample computing prowess and enabling short-range, low-delay information transfer [7]. This integration has established a

foundation for intelligent production, transmission, distribution and electricity, and simple applications have already been achieved [8,9]. Figure 1 shows the scenario graph of an edge computing assisted smart grid.

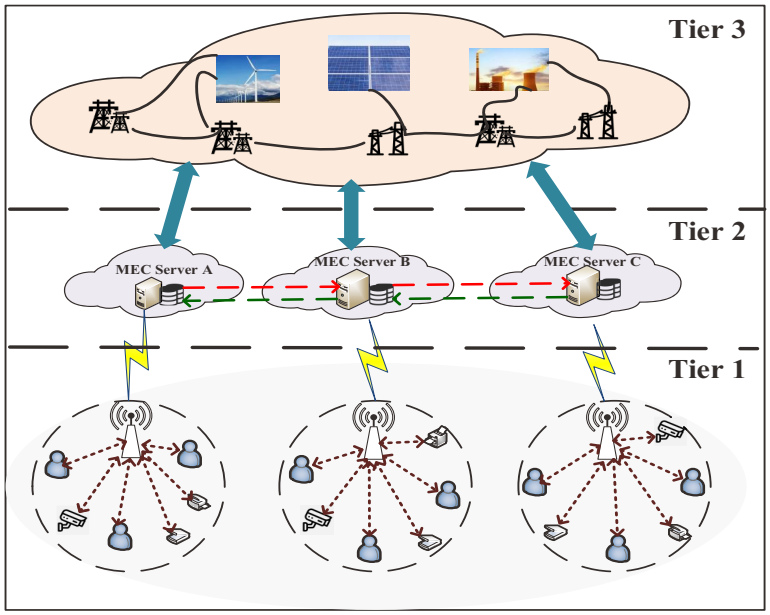

Figure 1. Edge computing assisted smart grid.

However, as a new computing mode, edge computing still has some problems, such as lack of unified programming model, dynamic scheduling method and security standards. Even though edge computing has a great advantage in application, there are also application bottlenecks [10]. Firstly, the high geographic distribution and heterogeneous characteristics of power grid equipment make it difficult to determine uniform standards and interfaces. Secondly, because the smart grid processes so much data per second, it can lead to the offloading of large-scale computing tasks. In addition, research on the edge computing security of smart grids is still in the initial stage, and some achievements have been attained but a complete research system has not been formed [11].

In a virtual network environment, the infrastructure provider (INP) controls the physical network's resources, while the service provider (SP) rents INP resources to construct a virtual network (VN) [12]. VN technology makes it possible to generate several VNs on the same physical network or to connect different physical networks to form a multi-domain virtual network [13]. These VNs can be deployed and managed independently without interfering with each other. Based on a grid-oriented edge computing network background, the bottom network nodes are divided into smart terminal data receiving points, sensor data receiving points and transformer data acquisition points. For the convenience of modeling and simulation, network resources are described as node and link resources [14]. The multi-domain scene of edge computing networks is divided into two problems: VN partition and virtual subnetwork mapping. A virtual subnet is a part of the VN that can be solved using a single-domain mapping algorithm. The multi-domain virtual network mapping algorithm focuses on VN partitioning between multiple domains [15].

As a branch of machine learning, RL aims to maximize the reward signal to optimize the updating algorithm [16]. The RL agent explores continuously in the environment to obtain information, first chooses the corresponding action in a certain state, then interacts with the environment to get the reward of this state; finally, according to the reward, it constantly adjusts its own strategy. RL is generally used to solve strategy optimization problems, including strategy, reward, value function and environmental model [17]. The four elements are as follows. (1) A policy represents a mapping from a given state to the actions taken in that state. (2) Reward is used to judge the behavior of the RL agent.

(3) The value function represents the expectation of a cumulative discount reward. (4) The environment model is the key of RL: the agent acquires the state and takes the action in the environment, thus realizing the interaction with the environment; at the same time, the environment will feed back the corresponding reward signal according to the action that the agent takes. In addition to these four factors, an agent's choice of actions at a given moment can affect not only the reward value at that moment but also its future performance.

The balance between the exploration and utilization of agents remains unresolved. Exploration is the hope that agents try different action choices on the basis of the existing strategies, so as to fully traverse the action space. Utilization is to expect the agent to learn the corresponding strategy on the basis of quickly grasping the existing experience. It can effectively save computing resources and ensure the stability of the algorithm. Exploration can prevent the strategy from falling into a local optimum but it also leads to a slow convergence rate of the algorithms [18]. Therefore, how to improve the agent's exploring ability is a problem worth discussing and studying. In this paper, we improve the existing policy network model from the point of view of policy network optimization, with low cost, high bandwidth and low delay as the goals of optimization for multi-objective allocation of edge computing network resources.

In the face of limited network resources, it is necessary to reasonably manage and allocate resources in the smart grid to improve the utilization of network resources to meet the needs of different application scenarios. Therefore, in this paper, we study the problem of resource management and scheduling to improve the utilization of network resources in smart grids. Specifically, we transform the resource management problem into a virtual network embedding (VNE) problem and then propose an RL-based VNE algorithm. The resources involved in the VNE process to be allocated are shown in Table A1. Simulation results show that the algorithm significantly improved resource utilization in edge computation networks, reduced task processing delays and reduced costs. The major contributions of this article are as follows:

1. We introduce edge computing into smart grids and transform the resource allocation of edge computing-assisted smart grids into a multi-domain VNE problem.
2. We propose a VNE algorithm based on dual RL, which uses a self-constructed policy network in the node mapping phase and link mapping phase, and designs a reward function with multi-objective optimization to achieve resource allocation.
3. Comparing the algorithm in this paper with three other VNE algorithms, it was found that this algorithm outperformed the other algorithms.

The remainder of this paper is organized as follows. We review related work on edge computing, multi-domain virtual networks and VNE algorithms in Section 2, and discuss network system models and related concepts in Section 3. In Section 4, we explain our proposed method in detail. Performance evaluation results and conclusion are described in Sections 5 and 6, respectively.

2. Related Works

2.1. Edge Computing Network

The scenarios and requirements of the distribution network have been changed by the development of the energy Internet, and the network structure is more complex and variable. There is an urgent need for cutting-edge technologies such as edge computing, software defined network and Internet of Things.

At this stage, scholars have conducted in-depth research on cloud-edge collaboration techniques. Ren et al. [19] proposed a two-tier multi-cloud center collaboration paradigm to effectively execute the complex computing requirements of mobile customers by using the computational collaboration between the upper cloud center and the edge cloud. A fiber-optic wireless access network architecture was proposed by Li et al. [20], and they used an approximate collaborative computation offloading algorithm and game theory to achieve joint offloading between cloud and mobile edge computing. A new framework of global management based on the distributed integration form of central and edge clouds

was explored by Deng et al. [21] and they explored nine application scenarios of cloud-edge collaboration. Three different implementations of edge computing were discussed by Dolui et al. [22], namely fog computing, Cloudlet and mobile edge computing; they compared their characteristics, introduced edge computing to the power industry, and proposed the application prospect of edge computing based on home energy management systems and power demand response businesses. Lin et al. [23] analyzed the development potential of cloud-edge collaboration and proposed a multi-dimensional approach to solve the collaboration problem. Su et al. [24] constructed a "city brain" based on the cloud-side collaborative urban vision computing platform using rapidly developing artificial intelligence technology. The "city brain" is used in urban target recognition, urban event perception, urban traffic management, urban digital modeling, etc. to support a rich variety of business scenarios in urban management such as traffic, public security, municipalities, education and medical care. Cloud-edge collaboration will become an important trend in the future development of intelligent industry technology, so that cloud computing and edge computing interact with each other to make up for the shortcomings in different application scenarios.

Edge-side collaboration is mainly to solve the contradiction between the resource demand of intelligent algorithms and the resource limitation of edge devices, and to balance the application service quality and privacy protection. Wang et al. [25] proposed an edge computing collaboration framework to support collaborative processing of latency-sensitive multimedia IoT tasks on resource-rich mobile devices, where the key is to assign video blocks to appropriate mobile edge video processing groups for processing. Tan et al. [26] optimized a collaborative service framework for edge computing systems based on the privacy trust and security assurance issues in edge computing, with full consideration of user quality of experience (QoE) for the user application requirements characteristics. Wang et al. [27] integrated edge computing into a low-power WAN, and the energy of two base stations is used to accomplish the computing tasks collaboratively. Deng et al. [28] studied the interaction and cooperation between the edge layer and the core cloud system to meet the low latency and high speed services, weighed the power consumption and delay in the edge and cloud computing system, and resolved the original problem into three sub-problems of the corresponding subsystem using the approximate method.

2.2. Multi-Domain Virtual Network

The key technology to conquer the rigidity of the power grid architecture is network virtualization, which can adapt to increasing business needs. The existing multi-domain VNE strategy can be classified into distributed mapping and centralized mapping according to whether there is an intermediary role between the SP and the InPs. In the distributed mapping strategy, the SP directly requests information from each InP, and then determines the final mapping result of VN through repeated comparison and negotiation.

Chowdhury et al. [29] forwarded the part of the VN that cannot be mapped by a single InP to the adjacent InPs to reduce the cost of cross-domain mapping, controlled the mapping process through signaling in the designed protocol and introduced an intermediate role to obtain global information, reducing the huge cost of negotiation. Dietrich et al. [30] study the problem of multi-domain VN mapping under the condition that INP only knows limited information such as boundary nodes and inter-domain links. They use the idea of integer programming to solve the virtual network request (VNR) partition problem. Although they obtain the optimal mapping cost scheme, they do not further consider the efficiency of VNR partition.

2.3. VNE Based AI

Academia has noticed that AI strategies can be used to solve the problem of VNE with the continuous development of artificial intelligence (AI) and machine learning. Yao et al. [31] proposed a VNE algorithm based on the policy network, which uses the historical request data of the VN to develop the RL agent, and finds the optimal embedding

using the policy gradient descent method. Zhang et al. [32] used graph convolution neural network (GCNN) based on RL to automatically extract dynamic physical network features to optimize VNE strategy. With the goal of optimizing the industrial Internet of Things architecture, Zhang et al. [33] measures the security of physical nodes using social attribute perception, standardizes resource constraints using resource knowledge description, trains deep RL agents to generate embedding probability and embeds virtual nodes according to probability.

The greatest challenge for edge computing lies in how to deploy computing and storage capabilities dynamically, and how to collaborate and connect cloud-edge devices efficiently and seamlessly. Multi-domain mapping does not require INP as a broker to coordinate mapping, but allows each InP to conduct distributed mapping based on common agreements. However, the heuristic mapping method has difficulty coping with the dynamically changing power grid resource allocation requirements and requires an artificial intelligence based embedding strategy to adapt to complex power grid scenarios.

3. Network Structure and Problem Description

Network Models

Considering network control and management, a large-scale heterogeneous network is usually divided according to geographical, functional and other factors. The multi-domain virtual network structure constructed in this paper takes three domains, namely the intelligent terminal domain, the sensor data receiving domain and the transformer data acquisition domain. In the multi-domain network environment, the resources of the physical network and the topological information of the infrastructure are the key factors to complete the mapping. In this paper, we group the physical network nodes based on the multi-domain network environment into three primary types: intelligent terminal data receiving nodes, sensor data receiving nodes and transformer data acquisition nodes. When considering the multi-domain VNE problem, the physical network can be represented as an undirected weighted graph in a modeling approach $G^P = \{N^P, L^P, R^P\}$. Therefore, N^P is represented as $N^P = \{N_I^P, N_S^P, N_T^P\}$.

Considering the variations in link connections among different network segments, we characterize them individually: $L^P = \{L_I^P, L_S^P, L_T^P, L_{IS}^P, L_{ST}^P, L_{IT}^P\}$. As for R^P, we break it down into four aspects: computational capability of nodes, bandwidth, security level of nodes and delay of links. Therefore, the physical network resource attributes are defined as $R^P = \{CPU^P, SL^P, BW^P, D^P\}$. The security level of nodes is defined as $SL^P = \{SL_I^P, SL_S^S, SL_T^P\}$. The bandwidth of links is defined as $BW^P = \{BW_I^P, BW_S^P, BW_T^P, BW_{IS}^P, BW_{ST}^P, BW_{IT}^P\}$. And $D^S = \{D_I^P, D_S^P, D_T^P, D_{IS}^P, D_{ST}^P, D_{IT}^P\}$. The VNR can be represented as an undirected weighted graph $G^V = \{N^V, L^V, R^V\}$, N^V represents the sets of virtual nodes and L^V represents the sets of the VNR's virtual link. And $R^V = \{CPU^V, SR^V, BW^V, D^V\}$. We summarize the main symbols and their explanations in Table 1.

Our paper centers on addressing the resource allocation challenge in smart grids to improve the utilization of network resources by transforming it into a multi-domain virtual network resource allocation puzzle. To achieve this, we break down the process into two vital stages: node embedding and link embedding. Specifically, we denote the node embedding function as $F1 = \{\theta_{ij} |\ n_i^v \in N^P, n_j^p \in N^P\}$ and have the constraint:

$$F1(n^v) = n^p, \forall n^p \in N^P, \exists\ unique\ n^v \in N^V. \tag{1}$$

Formula (1) indicates that a physical node n_j^p can only allow a unique virtual node n_i^v to be embedded, so θ_{ij} can be represented as:

$$\theta_{ij} = \begin{cases} 1,\ if\ n_i^v \to n_j^p. \\ 0,\ else. \end{cases} \tag{2}$$

and the embedding of the nodes should satisfy the following constraints:

$$\sum_{j=1}^{N^P} \theta_{ij} = 1, \forall n_i^v \in N^V, n_j^p \in N^P. \tag{3}$$

Formula (3) indicates that each virtual node can only be mapped to a single physical node in the current VNR. For another process, we defined a link embedding as $F2 : l_i^v \rightarrow L^P$, where l_i^v represents the i^{th} link in the VNR. The process of embedding can be represented as a function denoted by $F2 = \{\phi_{ij} | \, l_i^v \in L^V, l_j^s \in L^P\}$, where ϕ_{ij} can be formulated as:

$$\phi_{ij} = \begin{cases} 1, \; if \; l_i^v \rightarrow l_j^p. \\ 0, \; else. \end{cases} \tag{4}$$

But unlike node embedding, virtual network links can often be embedded in multiple physical network links. The process of link embedding can be expressed as

$$\sum_{j=1}^{N^P} \phi_{ij} \geq 1, \forall l_i^v \in L^V, l_j^p \in L^P. \tag{5}$$

In addition to the above constraints, we also need to take into account the constraints on the resource properties of the link. They are the computational capacity, security level, bandwidth and delay that were mentioned above. The following resource constraints specific to the edge computing network are set [34].

Table 1. Notation.

Notation	Definition
G^P	Physical networks of smart grid
N^P	Physical nodes set of smart grid
L^P	Physical links set of smart grid
R^P	Physical network resource attribute of smart grid
CPU^P	Computational capability of nodes in smart grid
SL^P	Security level of nodes in smart grid
BW^P	Bandwidth of links in smart grid
D^P	Delay of links in smart grid
G^V	VNR
N^V	Set of virtual nodes in VNR
L^V	Set of virtual links in VNR
R^V	Resource requirement in VNR
CPU^V	Computational capacity needed by the virtual nodes
SR^V	Security level requirements for virtual nodes
BW^V	Bandwidth requirements for the virtual links
D^V	Delay of the virtual links

1. Computational capacity: If n_i^v has been embedded on node n_j^p, the computational capacity resource requirement of the virtual network node does not exceed the computational capacity resource of the physical node, which can be given by the following formula:

$$if \; \theta_{ij} = 1, \; CPU_{n_i^v}^V \leq CPU_{n_j^p}^P. \tag{6}$$

In Formula (6), $CPU_{n_i^v}^V$ represents the computational capacity needed by the virtual node n_i^v and $CPU_{n_j^p}^P$ represents the computing resource available from the physical node n_j^p. The total computing resource requirements of all virtual nodes embedded in physical node n_j^p should not exceed the computational capacity of n_j^p.

$$\sum_{i=1}^{|VNR|} \sum_{n_i^v \rightarrow n_j^p} CPU_{n_i^v}^V \leq CPU_{n_j^p}^P. \tag{7}$$

Due to the fact that the nodes are divided into three defined types and have signed functions θ_{ij}, Formula (7) can be expressed as:

$$\theta_{ij} \sum_{j=1}^{N^P} (CPU^P_{In^p_j} + CPU^P_{Sn^p_j} + CPU^S_{Tn^p_j}) \geq \sum_{i=1}^{N^V} CPU^V_{n^v_i} x. \tag{8}$$

2. Security level: When embedding a virtual node onto a physical node, it is essential to ensure that the security level of the physical node is equal to or greater than the level of security required for the virtual node. We can express this as:

$$if\ \theta_{ij} = 1,\ SR^V_{n^v_i} \leq SL^P_{n^p_j}. \tag{9}$$

In Formula (9), the security level of n^p_j, denoted as $SL^P_{n^p_j}$, must be higher than the security requirements of n^v_i denoted as $SR^V_{n^v_i}$.

3. Bandwidth: For the bandwidth, we can give similar constraints to those of computational capacity resources:

$$if\ \phi_{ij} = 1,\ BW^V_{l^v_i} \leq BW^P_{l^p_j}. \tag{10}$$

In Formula (10), $BW^V_{l^v_i}$ represents the bandwidth demand of l^v_i. $BW^P_{l^p_j}$ represents the bandwidth resources available for l^p_j. The set of links that were able to successfully connect and integrate nodes we define as L^P_{ij}. For the physical link L^P_{ij}, the sum of the bandwidth resource requirements for virtual links embedded in L^P_{ij} must not exceed the available bandwidth resource of L^P_{ij}.

$$\sum_{i=1}^{|VNR|} \sum_{l^v_i \to l^p_j} BW^V_{l^v_i} \leq BW^P_{l^p_j}. \tag{11}$$

Another way to express the total bandwidth is:

$$\begin{aligned} \phi_{ij} \sum_{j=1}^{L^P} (&BW^P_{Il^p_{ij}} + BW^P_{Sl_i j^p} + BW^S_{Tl^p_{ij}} + BW^P_{ISl^p_{ij}} \\ &+ BW^P_{STl^p_{ij}} + BW^P_{ITl^p_{ij}}) \geq \sum_{i=1}^{L^V} BW^V_{l^v_i}. \end{aligned} \tag{12}$$

4. Delay: If the link is successfully embedded, the delay constraint means that the delay of the link used for the VNR should not be less than the VNR's required delay. This can be expressed as:

$$if\ \phi_{ij} = 1,\ D^V_{l^v_i} \leq D^P_{l^p_j}. \tag{13}$$

4. Algorithm

4.1. Algorithm Description

For the multi-objective allocation of network resources in edge computing networks, we propose a policy network-based RL intelligent resource allocation algorithm. The algorithm mainly consists of two main components: security-based node mapping and security-based link mapping.

4.2. Node Mapping

During the virtual node mapping phase, the physical node with the greatest possibility of being selected can be the mapping solution for the virtual node. In order to obtain the selected probability of physical nodes, extracting the state features of the physical nodes in the current state can be necessary; then, obtain the state matrix and put the state matrix into the trained node mapping policy network to output the selected probability of the physical nodes.

4.2.1. Node Features

In order to describe the node status more accurately and describe the current physical network status more accurately, this paper extracts node-related features. The main extracted features include Node Computing Resources, Adjacent Link Bandwidth, Distance Correlation, Time Correlation and Node Security; this paper also connects all the features of a single node into a feature vector, and the feature vectors of nodes are combined into a node feature state matrix that represents the physical network state.

- Node Computing Resources ($CPU(n_k^p)$): the available CPU resources of n_k^p.
- Adjacent Link Bandwidth ($BW(n_k^p)$): the quantity of available link bandwidths connected to n_k^p.
- Distance Correlation ($DSC(n_k^p)$): the average of distances from n_k^p to mapped nodes.
- Time Correlation ($TC(n_k^p)$): the average value of the quantity of the delays used from n_k^p to all other nodes in the network.
- Node Security ($NSE(n_k^p)$): measures the security of N_k^p deployable virtual nodes.

After extracting the features of all the physical nodes, their values are adjusted to be between 0 and 1 by the min–max normalization method:

$$x' = (x - x_{min})/(x_{max} - x_{min}). \tag{14}$$

So, the eigenvector of the k-th physical node can be represented by M_k:

$$M_k = (CPU(n_k^p), BW(n_k^p), DSC(n_k^p), TC(n_k^p), NSE(n_k^p)). \tag{15}$$

Take the eigenvectors of each physical node as a row to construct the node eigenstate matrix.

$$Matrix_{node} = (m_1, m_2, \ldots, m_n)^T. \tag{16}$$

4.2.2. Node Policy Network

Due to the continuous assignment of the state space of the VNE problem, this paper uses a policy-based approach for RL. Policy-based methods are usually combined with deep learning to evolve into policy network-based methods. The most important step in the process of node mapping is to calculate the probability of all possible choices through the current state of the physical network. This step can be achieved through a policy network. When mapping available virtual nodes, the policy network is used to find the best physical node to host virtual nodes. The node strategy network constructed in this paper includes input layer, convolution layer, softmax layer, filter layer and output layer.

- Input layer: Obtain the physical node feature matrix of the physical network by reading Formula 15 in the current state.
- Convolution layer: Evaluate the resources of each physical node. In order to acquire the available resource vector of each physical node, this layer performs convolution calculation on the feature matrix of the input layer. Each output of a convolutional layer represents an available resource vector for a physical node. The specific calculation method of this layer is as follows:

$$h_k = ReLU(w_n \cdot m_k + b^n) = \begin{cases} 0, otherwise. \\ w_n \cdot M_k + b_n, w_n \cdot M_k + b_n \geq 0. \end{cases} \tag{17}$$

$ReLU(\cdot)$ is a piecewise linear function, which is used as the activation function.

- Filtering layer: Refer to the resource vector obtained in the previous layer to judge whether the physical node is available. An unavailable physical node means that the remaining CPU resources of this physical node cannot support current virtual node mapping. The filtering layer filters out unavailable nodes and outputs the available resource vectors of all candidate physical nodes.
- Softmax layer: Normalize the available resource vectors of each physical node obtained by the filtering layer. The larger the value, the greater the benefits obtained when mapping the virtual node to the physical node.

$$p_k = \frac{h_k}{\sum_i h_i}. \tag{18}$$

- Output layer: Output the probability of each physical node being selected as a hosting node.

$$P_{node} = (p_1, p_2, \ldots, p_n)^T. \tag{19}$$

Reward function: The agent learns about the working state of the model through rewards. The reward function is fed back to the agent, so that the agent understands whether the current action is correct or wrong, and the agent performs self-regulation according to the reward function. In this paper, we use the weights method (α, β) to face different QoS requirements and the reward function is defined as:

$$Reward(G^V) = \alpha \times \frac{1}{Delay(G^V)} + \beta \times \frac{Rev(G^V)}{Cost(G^V)}. \tag{20}$$

$Delay(G^V)$ represents the delay of G^V, defined as

$$Delay(G^V) = \sum_{l^v \in L^V} \sum_{l^s \in M(l^v)} D(l^s), \tag{21}$$

where $M(l^v)$ represents the set of physical links mapped by l^v and $D(l^s)$ represents the delay of l^s. When G^V mapping is successful, revenue and cost are mainly related to the amount of resources required by G^V, and can be defined as Formulas (22) and (23), respectively. Among them, $CPU(n^v)$ and $BW(l^v)$, respectively, represent the amount of CPU resources required by n^v and the amount of bandwidth resources required by l^v; Δt represents the duration of G^V; $Hops(l^v)$ represents the number of hops of the virtual link.

$$Rev(G^V) = \Delta t \times \sum_{n^v \in N^V} CPU(n^v) + \Delta t \times \sum_{l^v \in L^V} BW(l^v). \tag{22}$$

$$Cost(G^V) = \Delta t \times \sum_{n^v \in N^V} CPU(n^v) + \Delta t \times \sum_{l^v \in L^V} BW(l^v) \times Hops(l^v). \tag{23}$$

4.3. Link Mapping

When mapping a virtual link, checking whether the physical link has sufficient resources is necessary. In this paper, link mapping also introduces RL. Actions are output according to the current physical network state matrix.

4.3.1. Link Features

During this phase, a virtual link may correspond to multiple paths of a physical link, so the features extracted by link mapping should be the features of the connectable paths between the two physical nodes in the physical network. This paper uses bandwidth and link importance as features.

- Bandwidth ($BW(l_j^p)$): The bandwidth value of the link with the smallest available bandwidth resources on the mapping path.

- Link importance ($BE(l_j^p)$): The more shortest paths passed on the physical link, the more important the link is.

$$LI(l_j^p) = \sum \frac{S_l}{S_{all}}. \tag{24}$$

S_l represents the number of shortest paths passing through l and S_{all} represents the number of all shortest paths passing through the network.

These physical path features are normalized to form a feature vector.

$$lm_j = (BW(l_j^p), BE(l_j^p))^T. \tag{25}$$

The eigenvectors of all physical links form the state matrix of the link.

$$Matrix_{link} = (lm_1, lm_2, \ldots, lm_n)^T. \tag{26}$$

4.3.2. Link Policy Network

We take the link state matrix as input and the probability of the physical path being chosen as output. The policy network is divided into five layers like the node mapping stage.

- Convolution layer: Perform the convolution operation on the link feature matrix obtained by the input layer, so as to evaluate the resource situation of the physical path.

$$g_j = ReLU(w_l \cdot M_j + b_l) = \begin{cases} 0, otherwise. \\ w_l \cdot M_j + b_l, w_l \cdot M_j + b_l \geq 0. \end{cases} \tag{27}$$

- Softmax layer: The probability is calculated as follows:

$$lp_j = \frac{e^{g_j}}{\sum_i e^{g_j}}. \tag{28}$$

- Filtering layer: When physical paths' bandwidth resources do not meet the VNRs, we abandon such paths. And when the nodes at ends of the physical path are not mapped during mapping nodes, we abandon such paths.
- Output layer: Output the selected probability of each physical link.

$$P_{link} = (lp_1, lp_2, \ldots, lp_n)^T. \tag{29}$$

4.4. Mapping Algorithm

In this paper, policy gradients are used to train the node mapping policy network. As the initial parameters are randomly given, which may not lead to a good policy, we introduce the minibatch gradient descent to dynamically update parameters. The update of parameters is implemented in batches; we update the policy parameters when reaching the iteration threshold β. We adjust the gradient and the calculation speed of the training by introducing the learning rate α. A gradient that is too small will make it difficult for the model to converge and one that is too large will cause an unstable model.

The flowchart of node mapping model training is shown in Figure 2. We randomly set all the node policy network parameters and then select the most suitable physical node for virtual nodes in the VNR. We obtain the node feature matrix by extracting features on all physical nodes and normalize the matrix, and use the normalized results as the input of the node mapping policy network.

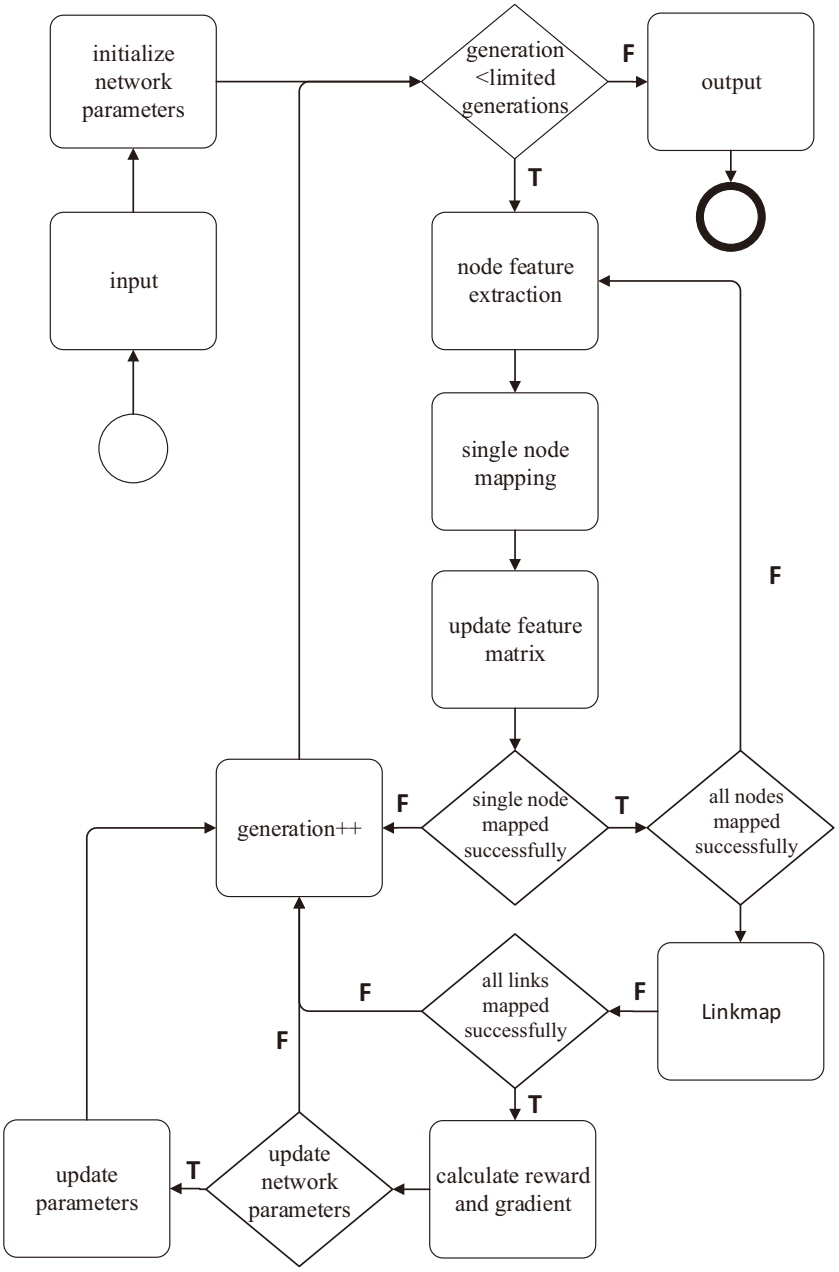

Figure 2. Node mapping algorithm model training flowchart.

After that, the probabilities of a set of candidate nodes are obtained. When selecting the physical nodes to be mapped, we choose the node with the highest probability from the candidate nodes. Current physical network resources are updated each time a virtual node is mapped. After all nodes in the VNR are successfully mapped, we map links using the shortest path algorithm. The reward function and gradient at this time are calculated after successfully mapping the link. The policy network parameters are updated in batches and the above process is repeated until the optimal node mapping policy network parameters are obtained when the VNR reaches the threshold. Algorithm 1 gives the pseudocode for the training of the node mapping model.

Algorithm 1: Training Node Mapping Model

input : Training Set VNRs; Physical Network Topology G^p; Iteration threshold β; *limited generations*;
output: Node Mapping Policy Network;

Initialization;
while *generation* $\leq$ *limited generations* **do**
  $Para_{num} = 0$;
  foreach *vnr in VNRs* **do**
    foreach n^v *in vnr* **do**
      $Matrix_{node} \leftarrow \varnothing$;
      foreach n_k^p *in* G^P **do**
        $N_k = (CPU(n_k^p), BW(n_k^p), DSC(n_k^p), TC(n_k^p), NSE(n_k^p))$
        $Matrix_{node} \leftarrow Matrix_{node} + N_k$;
      end
      // Policy network output mapping physical nodes;
      P_{node}=Policy network($Matrix_{node}$);
      update $Matrix_{node}$;
    end
    if $\forall\ n^v \in$ *vnr Mapped successfully* **then**
      // Link Mapping Using Shortest Path Algorithm;
      LinkMapping;
    end
    if *vnr Mapped successfully* **then**
      // Calculate rewards by Formula (20);
      getReward(vnr);
      // Calculate the gradient;
      getGradient(reward);
    end
    $Para_{num} + +$;;
    if $Para_{num} = \beta$ **then**
      $Para_{num} = 0$;
      update Policy Network Parameters;
    end
  end
  limited generations $+ +$;
end

Figure 3 shows the training flowchart of the link map model. The parameters in the policy network are randomly initialized above all while training the link model and, when processing the VNR, the node mapping plan is obtained by utilizing the trained node mapping policy network. Then, the link features of the physical network are extracted to obtain the link state matrix. The probability of physical path candidates is obtained by putting the state matrix into the link policy network. According to the probability distribution, the highest probability path is selected as the physical path for this mapping. Then, each time a virtual node is mapped, the current physical network resources are updated. Repeat the above steps until all the virtual links of the current VNR are mapped. If no physical link is available, the mapping fails and the next VNR is mapped.

To adjust the training direction and obtain a better solution, we use the reward function as the feedback of the mapping policy network. The policy network parameters are updated in batches and the above process is repeated until the optimal link mapping policy network parameters are obtained when the VNR reaches the threshold.

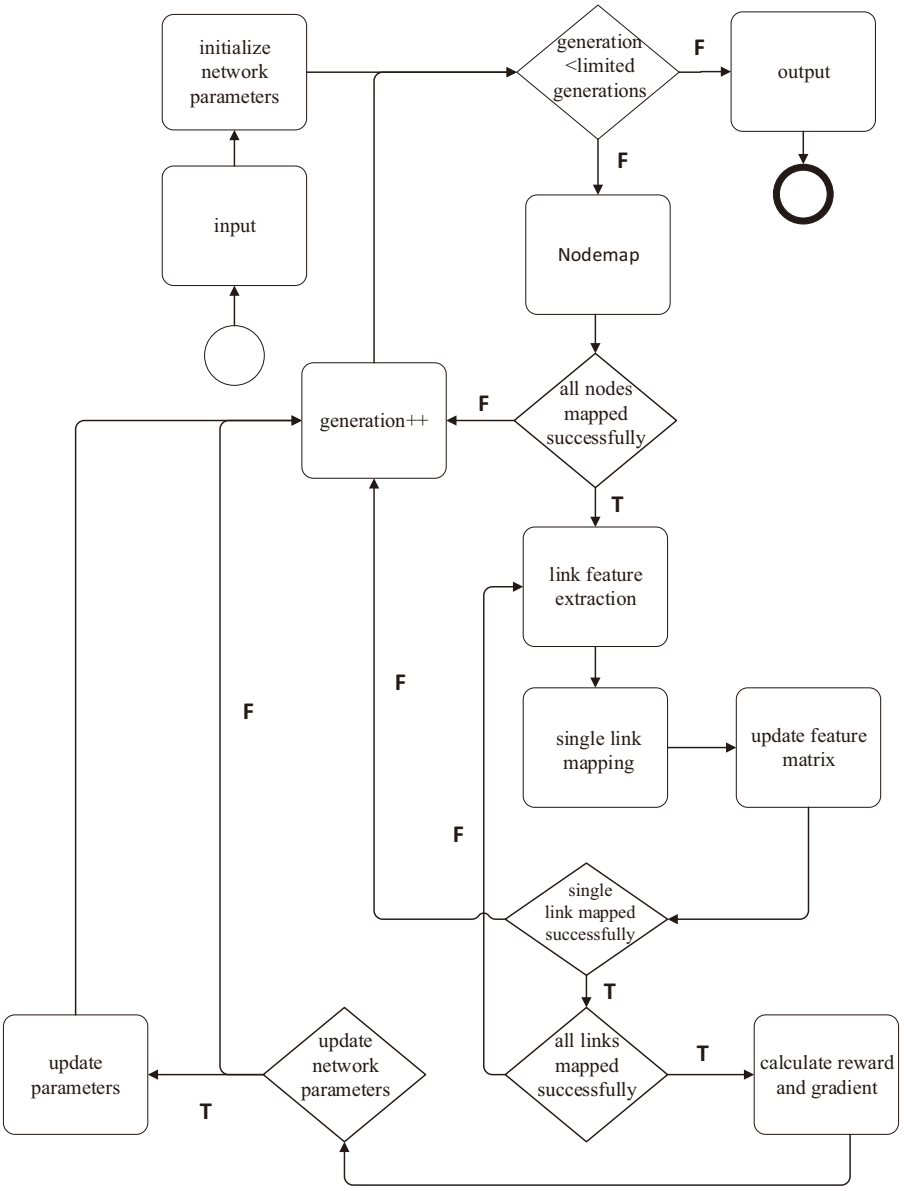

Figure 3. Link mapping algorithm model training flowchart.

When virtual node mapping is applied, the current node state matrix is sent to the trained node policy network. Obtain the selection probability of the available physical nodes. Then, take the highest probability physical node as the mapping of the current virtual node. After all nodes of the VNR are mapped, the link state matrix is sent to the trained link policy network. The selection probability of available physical links is obtained and the highest probability physical link is taken as the mapping of the current virtual link. When the nodes and links of VNR are mapped successfully, the output is that VNR mapping is successful; otherwise, output that VNR mapping is failing. The training process of the link mapping model is shown in Algorithm 2.

Algorithm 2: Training Link Mapping Model

input : Node Policy Network; Training Set VNRs; Physical Network G^p; *limited generations*; Iteration threshold β;
output: Link Mapping Policy Network;

initialization;
while *generation* $\leq$ *limited generations* **do**
  $Para_{num} = 0$;
  foreach *vnr in VNRs* **do**
    NodeMapping[**Algorithm** 1];
    if $\forall\ n^v \in vnr$ *is Mapped* **then**
      $Matrix_{link} \leftarrow \varnothing$;
      foreach l_k^p *in* G^P **do**
        $lm_k = (BW(n_k^p), BE(n_k^p))$;
        $Matrix_{link} \leftarrow Matrix_{link} + lm_k$;
      end
      P_{link} =Policy network ($Matrix_{link}$);
      update $Matrix_{link}$, $Matrix_{node}$;
    end
    if *vnr Mapped successfully* **then**
      // Calculate rewards by Formula (20);
      getReward(vnr);
      // Calculate the gradient;
      getGradient(reward);
    end
    $Para_{num} + +$;;
    if $Para_{num} = \beta$ **then**
      $Para_{num} = 0$;
      update Policy Network Parameters;
    end
  end
  limited generations $+ +$;
end

5. Experimental Results and Analysis

In this section, comparative experimental results will validate the proposed algorithm. Specifically, the experimental environment and parameters are first introduced, followed by a detailed description of the comparison algorithms and finally a theoretical analysis of the numerical results.

5.1. Experimental Environment and Parameters

In the experimental process of this paper, for the allocation of network resources, we choose appropriate physical nodes and physical links to map through the proposed algorithm, and rationally allocate CPU resources and bandwidth resources to improve the utilization of network resources. The topology structure of the physical network is generated by GT-ITM, which contains 100 nodes and 550 links, which is equivalent to a medium-scale network environment. The attribute information of nodes and links is generated programmatically and saved in the *.txt* file. The detailed attribute information of nodes and links is shown in Table 2. At the same time, we also generate 2000 VNRs, and use 1000 VNRs as training samples and another 1000 VNRs as testing samples. Each VNR has 2-10 nodes and the connection probability between nodes is 50%, so the VNR has $n(n-1)/4$ links. In addition, in order to simulate VNR more realistically, the arrival time and departure time of VNR conform to the mathematical distribution. The detailed parameters of VNR are also summarized in Table 2.

Table 2. Experimental environment parameters.

Network	Parameter	Value
Physical Network	Physical nodes	100
	Physical links	550
	CPU resources capacity	$U(50, 80)$
	Security level	$U(1, 3)$
	Bandwidth resource capacity	$U(50, 80)$
	Delay of link	$U(1, 50)$
VNRs	Number of VNRs	2000
	Training set	1000
	Testing set	1000
	Virtual nodes	$U(2, 10)$
	Node connection probability	0.5
	CPU resource requirement	$U(1, 30)$
	Bandwidth resource requirement	$U(1, 30)$
	Delay requirement of link	$U(1, 20)$
	Safety requirement level	$U(1, 3)$

5.2. Comparison Algorithms

We selected the following three representative algorithms as benchmark algorithms:

- RLVNE [31]: RLVNE is an RL-based VNE algorithm. Specifically, the author constructs a four-layer policy network, uses the policy gradient algorithm to update the policy network parameters and selects the mapping nodes according to the node mapping probabilities output by the policy network.
- NODERANK [35]: NODERANK is improved based on the PageRank algorithm [36]. The resource availability of nodes is taken as an important factor for node ordering and the nodes are mapped according to the ordering positions of nodes.
- VNE-HPSO [37]: VNE-HPSO is a heuristic algorithm that combines a particle swarm optimization algorithm and a simulated annealing method.

5.3. Performance Metrics

(1) Acceptance ratio

Define G_{acc}^{V} as the number of VNRs that have arrived and G_{arr}^{V} as the number of successfully embedded VNRs. The formula for the acceptance ratio is defined as follows:

$$ACC = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} G_{acc}^{V}}{\sum_{t=0}^{T} G_{arr}^{V}}. \tag{30}$$

(2) Long-term average revenue

Based on Formula (22), long-term average revenue is defined as

$$L\_REV = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} Rev(G^{V}, t)}{T}. \tag{31}$$

(3) Long-term average revenue–cost ratio

Long-term average revenue–cost ratio is defined as the ratio of revenue to cost over a period of time and the calculation formula is

$$L\_RC = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} Rev(G^{V}, t)}{\sum_{t=0}^{T} Cost(G^{V}, t)}. \tag{32}$$

(4) CPU resource utilization

In order to evaluate the resource utilization of the physical network, this paper selects the performance metrics of CPU resource utilization, which is defined as

$$U\_CPU = 1 - \frac{\sum_{n^p \in N^P} RC(n^p)}{\sum_{n^p \in N^P} AC(n^p)}, \tag{33}$$

where $RC(n^p)$ represents the remaining CPU resources of n^p and $RC(n^p)$ represents the total CPU resources of n^p.

5.4. Experimental Results and Analyses

5.4.1. Training Results

(1) Training result verification: To obtain the best performing policy network parameters, we first train a DRL agent. Specifically, we observe the agent's training results by setting different learning rates (0.01, 0.005, 0.001) and using three performance metrics: acceptance ratio, revenue and revenue-to-cost ratio. At the same time, we record the loss value during the iteration process, which is also an important indicator for evaluating the training effect.

(2) Discussion: Figure 4 shows the experimental values of the three performance indicators during the policy network training process. In the initial training stage, because the parameters of the network are random and the agent is not familiar with the network environment, the strategy at this time is not optimal and the values of the three indicators are relatively low. After a period of training, the agent becomes familiar with the network and the values of the three indicators continue to increase. At the same time, by setting different learning rates for training, we found that when the learning rate is 0.005, the training effect is the best. Table 3 shows the model training results under different learning rates. For best training performance, we fix the training learning rate at 0.005. Figure 5 shows the trend of the loss value during the training process and testing process. With the continuous optimization of parameters, the loss value gradually decreases and tends to be stable. It can be verified that our proposed algorithm has good convergence.

Table 3. Training result with different learning rates.

Learning Rate	Acceptance Ratio	Revenue	Revenue/Cost
0.001	0.625	1600	0.315
0.005	0.725	2050	0.405
0.01	0.450	1425	0.29

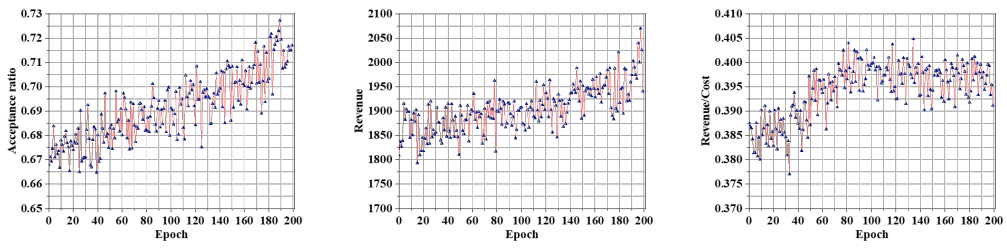

Figure 4. Policy network training process.

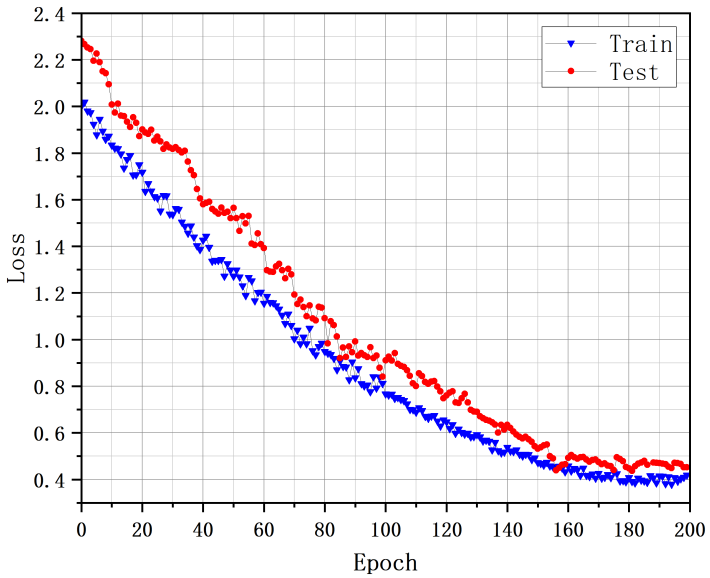

Figure 5. Loss on the training set.

5.4.2. Comparative Experimental Results

Experiment 1: Acceptance ratio. We first conducted a comparative experiment on the VNR acceptance ratio, which is a key indicator for evaluating the effectiveness of the algorithm, because one of the goals of the VNE problem is to serve as many VNRs as possible within limited resources.

Discussion: Figure 6 reflects the experimental results of the VNR acceptance ratio. It can be seen that the acceptance ratios of the four algorithms decrease significantly over time. Because the resources of the physical network are limited, when the VNR is successfully embedded, it will occupy node resources and link resources. For the newly arrived VNR, some nodes or links may not have enough resources to carry them, so the embedding fails. However, our algorithm outperforms the other three algorithms in any given time period. Finally, the VNR acceptance ratio of our algorithm is 6.34% higher than RLVNE, 8.06% higher than VNE-HPSO and 12.17% higher than NODERANK. Compared with RLVNE, our algorithm optimizes the two stages of node mapping and link mapping at the same time, which improves the VNR acceptance ratio. The initial value of the particles in the VNE-HPSO algorithm is randomly initialized, and the results of multiple optimizations do not always converge to the global or local optimal solution without changing any parameters, so the VNR acceptance ratio drops significantly. NodeRank adopts a manual node sorting method; the algorithm lacks flexibility and only considers the local importance of node attributes, so it has a low VNR acceptance ratio.

Experiment 2: Long-term average revenue. The revenue is usually related to the amount of VNR resources, which is usually positively correlated with the acceptance ratio but also related to whether the physical resources are used efficiently. Therefore, we conducted a comparative experiment of long-term average revenue.

Discussion: Figure 7 shows the experimental results for long-term average revenue. In the early stage of the test, the network resources such as nodes and links are relatively abundant and can carry most of the VNRs, so the long-term average revenue of the four algorithms are all relatively high. Due to the occupation of a large number of network resources and the failure of some VNR embeddings, the long-term average revenues of the four algorithms all declined. Through data analysis, our long-term average revenue is 7.39% higher than RLVNE, 17.98% higher than VNE-HPSO and 21.05% higher than NODERANK. Compared to the NODERANK, VNE-HPSO and RLVNE algorithms, the proposed method is able to continuously improve the model according to the rewards received by the agent. Therefore, our algorithm can more optimally utilize limited network resources to deploy VNR and obtain stable and higher revenue.

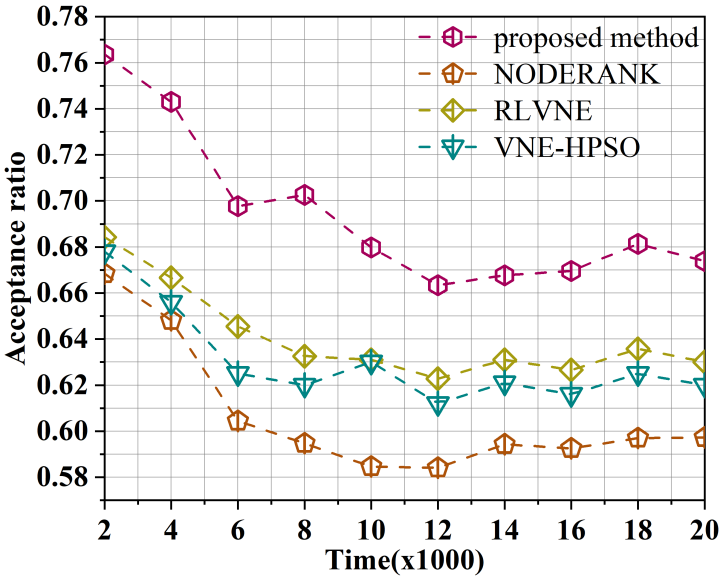

Figure 6. Acceptance ratio comparison.

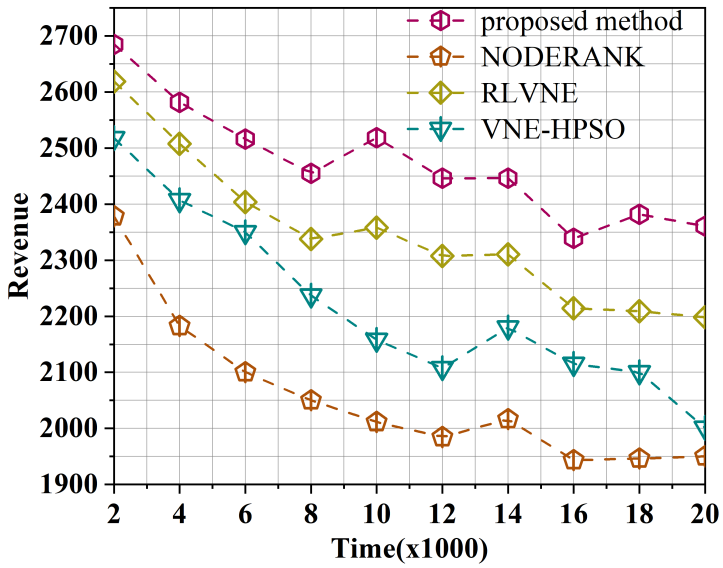

Figure 7. Long-term average revenue comparison.

Experiment 3: Long-term average revenue–cost ratio. The long-term average revenue–cost ratio considers revenues and costs at the same time, and can reflect the utilization efficiency of physical resources over a period of time. The higher the revenue–cost ratio, the higher the revenues that can be obtained even with the same physical resources.

Discussion: As can be seen from Figure 8, compared with the curve of the first two performance indicators, the long-term average revenue–cost ratio curve fluctuates less. Because the mapping cost decreases as the revenue obtained decreases, changes in the resources of the physical network do not significantly affect the change in the revenue–cost ratio. According to the experimental results, our long-term average revenue–cost ratio is 4.12%, 5.68% and 6.10% higher than RLVNE, VNE-HPSO and NODERANK, respectively. The link mapping cost is an important factor that affects the revenue–cost ratio, whereas the other three algorithms optimize only the node mapping and will have a higher revenue–cost ratio in the case where the link mapping uses only the shortest path algorithm. Our algorithm optimizes link mapping and is able to reduce the mapping cost in the link phase.

In addition, we consider this evaluation metric in the design of the reward function, which will guide the agent or strategies that reduce the revenue–cost ratio.

Experiment 4: CPU resource utilization. Finally, in order to more comprehensively evaluate the utilization efficiency of physical network resources, we conducted a comparative experiment of CPU resource utilization.

Discussion: It can be seen from Figure 9 that our algorithm has the highest CPU resource utilization efficiency, indicating that our node mapping network optimizes node mapping. Moreover, compared with RLVNE, the policy network designed by our algorithm can better reflect the network environment. NODERANK and VNE-HPSO are two non-machine learning algorithms; because they cannot respond to the dynamic network environment in time, the acceptance ratio is relatively low, and the CPU utilization rate will be affected and reduced. Finally, the CPU resource utilization of our algorithm is 22.67% higher than those of the three algorithms on average.

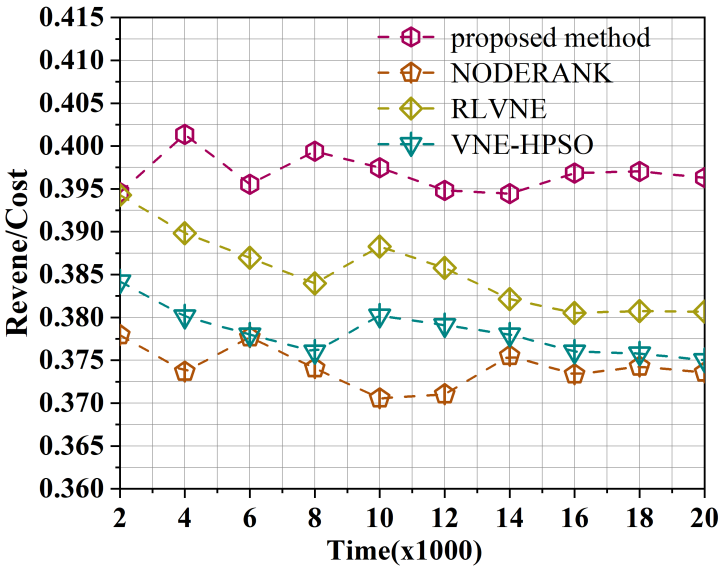

Figure 8. Long-term average revenue–cost ratio comparison.

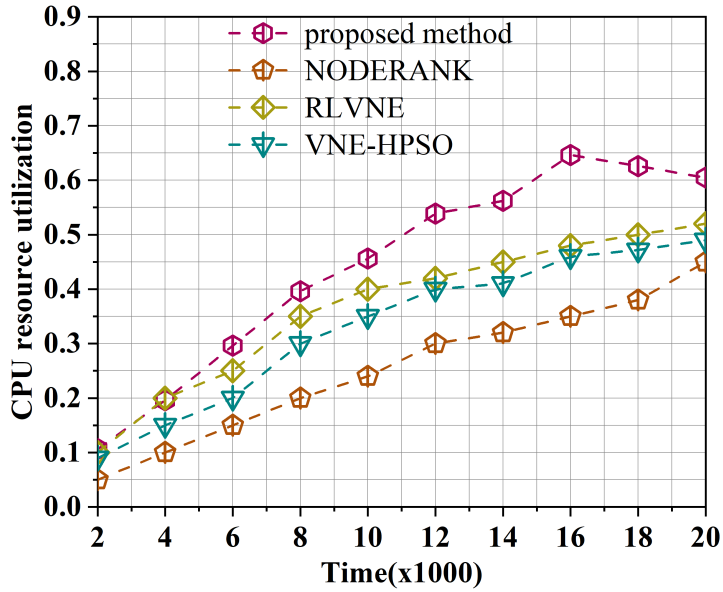

Figure 9. CPU resource utilization comparison.

6. Conclusions

This paper introduces edge computing into the smart grid, transforms resource allocation into a multi-domain virtual network embedding problem to improve the utilization of network resources and proposes a virtual network embedding algorithm based on double reinforcement learning. Unlike previous reinforcement learning-based studies, we simultaneously optimize node mapping and link mapping using reinforcement learning. Specifically, we extract the features of nodes and links, construct a policy network for node maps and link maps, respectively, and design reward functions with multi-objective optimization. Through training, our self-built neural network has good convergence and can make an optimal resource allocation strategy to improve the utilization of network resources. In addition, our algorithm can process dynamically arriving virtual network requests immediately, instead of processing a large number of requests at one time, which is in line with the actual network environment. Finally, we compared the proposed algorithm with the existing research on resource allocation algorithms. Overall, in this paper, we address the resource allocation to improve the utilization of network resources problem in edge computing-assisted smart grid scenarios from the perspective of virtual network embedding.

Although our algorithm improve the utilization of resources of the smart grid, it still has certain limitations. First of all, what we simulated in the experimental part is a medium-scale network. We should consider how to design a more complex policy network model and a more effective feature extraction method to deal with large-scale networks. Secondly, in the face of joint optimization of multi-grid resources, we should further consider the issue of data security, which is very important. In future work, we need to study how to improve data security with the help of blockchain and federated learning technology. Finally, we need to consider more network attributes, which are more in line with the real network environment.

Author Contributions: Conceptualization, Z.Z. and Y.H.; methodology, Y.C.; software, F.Y.; validation, W.C., H.Z. and Y.Z.; formal analysis, Z.Z.; investigation, P.Z.; resources, Y.Z. and P.Z.; data curation, Z.Z.; writing—original draft preparation, Z.Z. and Y.H.; writing—review and editing, Y.Z. and Y.C.; visualization, F.Y.; supervision, P.Z.; project administration, H.Z.; funding acquisition, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Scientific Research Programs for High-Level Talents of Beijing Smart-chip Microelectronics Technology Co., Ltd., and partially supported by the Academician Expert Open Fund of Beijing Smart-chip Microelectronics Technology Company Ltd.

Data Availability Statement: Data sharing not applicable.

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

Appendix A

Table A1. List of allocated resources.

Mapping Stage	Resource Name
Node Mapping	Physical nodes CPU resources
Link Mapping	Physical links Bandwidth resources

References

1. Islam, S.; Zografopoulos, I.; Hossain, M.T.; Badsha, S.; Konstantinou, C. A Resource Allocation Scheme for Energy Demand Management in 6G-enabled Smart Grid. In Proceedings of the 2023 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 16–19 January 2023 ; pp. 1–5. [CrossRef]
2. Fang, X.; Misra, S.; Xue, G.; Yang, D. Smart grid—The new and improved power grid: A survey. *IEEE Commun. Surv. Tutor.* **2011**, *14*, 944–980. [CrossRef]
3. Zhou, H.; Zhang, Z.; Li, D.; Su, Z. Joint Optimization of Computing Offloading and Service Caching in Edge Computing-Based Smart Grid. *IEEE Trans. Cloud Comput.* **2023**, *11*, 1122–1132. [CrossRef]
4. Ma, R.; Yi, Z.; Xiang, Y.; Shi, D.; Xu, C.; Wu, H. A Blockchain-Enabled Demand Management and Control Framework Driven by Deep Reinforcement Learning. *IEEE Trans. Ind. Electron.* **2023**, *70*, 430–440. [CrossRef]
5. Yang, C.; Chen, X.; Liu, Y.; Zhong, W.; Xie, S. Efficient task offloading and resource allocation for edge computing-based smart grid networks. In Proceedings of the ICC 2019–2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
6. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358.
7. Yang, X.; Yu, X.; Hou, H.; Tan, Z.; Wu, F. Smart grid edge fault detection architecture. In Proceedings of the 2023 IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 24–26 February 2023; Volume 6, pp. 692–700. [CrossRef]
8. Liao, Y.; He, J. Optimal Smart Grid Operation and Control Enhancement by Edge Computing. In Proceedings of the 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Tempe, AZ, USA, 11–13 November 2020; pp. 1–6.
9. Liu, R.; Yang, R.; Wang, Z.; Sun, X. Application of Edge Computing in Smart Grid. In Proceedings of the 2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Xi'an, China, 15–17 July 2022; pp. 62–65.
10. Xiao, Y.; Jia, Y.; Liu, C.; Cheng, X.; Yu, J.; Lv, W. Edge computing security: State of the art and challenges. *Proc. IEEE* **2019**, *107*, 1608–1631. [CrossRef]
11. Aloul, F.; Al-Ali, A.; Al-Dalky, R.; Al-Mardini, M.; El-Hajj, W. Smart grid security: Threats, vulnerabilities and solutions. *Int. J. Smart Grid Clean Energy* **2012**, *1*, 1–6. [CrossRef]
12. Bhamare, D.; Jain, R.; Samaka, M.; Erbad, A. A survey on service function chaining. *J. Netw. Comput. Appl.* **2016**, *75*, 138–155. [CrossRef]
13. Huang, L.H.; Hsu, H.C.; Shen, S.H.; Yang, D.N.; Chen, W.T. Multicast traffic engineering for software-defined networks. In Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
14. Wang, C.; Dong, T.; Duan, Y.; Sun, Q.; Zhang, P. Multi objective resource optimization of wireless network based on cross domain virtual network embedding. In Proceedings of the 2020 IEEE Computing, Communications and IoT Applications (ComComAp), Beijing, China, 20–22 December 2020; pp. 1–6.
15. Zhang, P.; Wang, C.; Qin, Z.; Cao, H. A multi-domain VNE algorithm based on multi-objective optimization for IoD architecture in Industry 4.0. *arXiv* **2022**, arXiv:2202.12830.
16. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
17. Ayoub, A.; Jia, Z.; Szepesvari, C.; Wang, M.; Yang, L. Model-based reinforcement learning with value-targeted regression. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 463–474.
18. Chen, N.; Zhang, P.; Kumar, N.; Hsu, C.H.; Abualigah, L.; Zhu, H. Spectral graph theory-based virtual network embedding for vehicular fog computing: A deep reinforcement learning architecture. *Knowl.-Based Syst.* **2022**, *257*, 109931.
19. Ren, J.; Guo, Y.; Zhang, D.; Liu, Q.; Zhang, Y. Distributed and efficient object detection in edge computing: Challenges and solutions. *IEEE Netw.* **2018**, *32*, 137–143. [CrossRef]
20. Li, L.; Ota, K.; Dong, M. Deep learning for smart industry: Efficient manufacture inspection system with fog computing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4665–4673. [CrossRef]
21. Deng, X.; Guan, P.; Wan, Z.; Liu, E.; Luo, J.; Zhao, Z.; Liu, Y.; Zhang, H. Integrated trust based resource cooperation in edge computing. *J. Comput. Res. Dev.* **2018**, *55*, 449–477.
22. Dolui, K.; Datta, S.K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
23. Lin, H.; Chen, Z.; Wang, L. Offloading for edge computing in low power wide area networks with energy harvesting. *IEEE Access* **2019**, *7*, 78919–78929. [CrossRef]
24. Su, X.; Sperlì, G.; Moscato, V.; Picariello, A.; Esposito, C.; Choi, C. An edge intelligence empowered recommender system enabling cultural heritage applications. *IEEE Trans. Ind. Inform.* **2019**, *15*, 4266–4275. [CrossRef]
25. Wang, X.; Han, Y.; Leung, V.C.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 869–904. [CrossRef]
26. Tan, K.; Wang, X.; Du, P. Research progress of the remote sensing classification combining deep learning and semi-supervised learning. *J. Image Graph.* **2019**, *24*, 1823–1841.

27. Wang, F.; Wen, H.; Cheng, S. Privacy data protection method for mobile intelligent terminal based on edge computing. *Cyberspace Secur.* **2018**, *9*, 47–50.
28. Deng, R.; Lu, R.; Lai, C.; Luan, T.H.; Liang, H. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet Things J.* **2016**, *3*, 1171–1181. [CrossRef]
29. Chowdhury, N.M.K.; Rahman, M.R.; Boutaba, R. Virtual network embedding with coordinated node and link mapping. In Proceedings of the IEEE INFOCOM 2009, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 783–791.
30. Dietrich, D.; Rizk, A.; Papadimitriou, P. Multi-domain virtual network embedding with limited information disclosure. In Proceedings of the 2013 IFIP Networking Conference, Brooklyn, NY, USA, 22–24 May 2013; pp. 1–9.
31. Yao, H.; Chen, X.; Li, M.; Zhang, P.; Wang, L. A novel reinforcement learning algorithm for virtual network embedding. *Neurocomputing* **2018**, *284*, 1–9. [CrossRef]
32. Zhang, P.; Wang, C.; Kumar, N.; Zhang, W.; Liu, L. Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning. *IEEE Internet Things J.* **2021**, *9*, 9389–9398. [CrossRef]
33. Zhang, P.; Gan, P.; Kumar, N.; Hsu, C.H.; Shen, S.; Li, S. RKD-VNE: Virtual network embedding algorithm assisted by resource knowledge description and deep reinforcement learning in IIoT scenario. *Future Gener. Comput. Syst.* **2022**, *135*, 426–437. [CrossRef]
34. Zhang, P.; Zhang, Y.; Kumar, N.; Hsu, C.H. Deep reinforcement learning algorithm for latency-oriented iiot resource orchestration. *IEEE Internet Things J.* **2022**, *10*, 7153–7163.
35. Cheng, X.; Su, S.; Zhang, Z.; Wang, H.; Yang, F.; Luo, Y.; Wang, J. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 38–47.
36. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*; Technical Report; Stanford InfoLab: Stanford, CA, USA, 1999.
37. Zhang, P.; Hong, Y.; Pang, X.; Jiang, C. VNE-HPSO: Virtual network embedding algorithm based on hybrid particle swarm optimization. *IEEE Access* **2020**, *8*, 213389–213400. [CrossRef]